Demand Estimation and Fleet Management for Autonomous Mobility on Demand Systems

by

Justin Lee Miller

B.S., Arizona State University (2009)
S.M., Massachusetts Institute of Technology (2011)

Submitted to the Department of Mechanical Engineering in partial fulfillment of the requirements for the degree of Doctor of Philosophy at the MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2017
© Massachusetts Institute of Technology 2017. All rights reserved.

Author

Signature redacted

Department of Mechanical Engineering

August 14, 2017

Certified by

Signature redacted

Richard Cockburn Maclaurin Professor of Aeronautics and Astronautics

Thesis Supervisor

Certified by

Signature redacted

Sanjay E. Sarma

Fred Fort Flowers and Daniel Fort Flowers

Professor of Mechanical Engineering Chairman, Thesis Committee

Certified by

Signature redacted

Stephen C. Graves

Abraham J. Siegel Professor of Management Science and Professor of Mechanical Engineering Member, Thesis Committee

Accepted by

Signature redacted

Rohan Abeyaratne

Chairman, Department Committee on Graduate Theses
Demand Estimation and Fleet Management for Autonomous Mobility on Demand Systems

by

Justin Lee Miller

Submitted to the Department of Mechanical Engineering on August 14, 2017, in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

Mobility On Demand (MOD) systems are creating a paradigm shift in transportation, where mobility is provided not through personally owned vehicles but rather through a fleet of shared vehicles. To maintain a high customer quality of service (QoS), MOD systems need to manage the distribution of vehicles under spatial and temporal fluctuations in customer demand.

A challenge for MOD systems is developing and informing a customer demand model. A new proactive demand model is presented which correlates real-time traffic data to predict customer demand on short timescales. Traditional traffic data collection approaches use pervasive fixed sensors which are costly for system-wide coverage. To address this, new frameworks are presented for measuring real-time traffic data using MOD vehicles as mobile sensors. The frameworks are evaluated using hardware and simulation implementations of a real-world MOD system developed for MIT campus. First, a mobile sensing framework is introduced that uses camera and Lidar sensors onboard MOD shuttles to observe system-wide traffic. Through a principled approach for decoupling dependencies between observation data and vehicle motion, the framework provides traffic rate estimates comparable to those of costly fixed sensors. Second, an active sensing framework is introduced which quantifies demand uncertainty with a Bayesian model and routes mobile sensors to reduce parameter uncertainty. The active sensing framework reduces error in demand estimates over both short and long timescales when compared to baseline approaches.

Given estimates of customer demand, the challenge for MOD systems is maintaining high customer QoS through fleet management. New automated fleet management planners are introduced for improving customer QoS in ride hailing, ride requesting, and ridesharing MOD operating frameworks. The planners are evaluated using data-driven simulation of the MIT MOD system. For ride hailing, to address the challenge of missed customers, a chance-constrained planner is introduced for positioning vehicles at likely customer hailing locations. The chance-constrained planner provides a significant improvement in the number of served hailing customers over a baseline exploration approach. For ride requesting, to address the challenge of high customer wait times, a
predictive positioning planner is introduced to position vehicles at key locations in the MOD system based on customer demand. The predictive positioning planner provides a reduction in service times for requesting customers compared to a baseline waiting approach. For ridesharing, incorrect assumptions on customer preference for transit delays can lead to poor realized customer QoS. A ridesharing planner is introduced for assigning customers to vehicles based on a trained ratings-based QoS model. The ridesharing planner provides robust performance over a range of unknown customer preferences compared to approaches with assumed customer preferences.

Thesis Supervisor: Jonathan P. How
Title: Richard Cockburn Maclaurin Professor of Aeronautics and Astronautics
Acknowledgments

The PhD experience for me has been more than just the process of obtaining a degree or a check-mark on my resume, it has been a phase of life that has provided me with personal and professional growth. There are many people that I’d like to thank for their support of me and this work along the way.

First, I’d like to thank my advisor, Professor Jonathan How. I joined Jon’s lab halfway through my PhD program, and at that time, I had no clue how I was ever going to make it out. Over these past few years, Jon has pushed me and taught me the proper way to go about research. His advice on every aspect of my thesis, from building hardware, to formulating problem statements, to submitting papers, is ultimately what got me through to the end.

I’d like to thank my committee chair Professor Sanjay Sarma. I’m grateful to Sanjay for mentoring me throughout my entire PhD process, providing advise on both research and life. I would also like to thank Professor Stephen Graves for joining my committee and providing valuable feedback and encouragement.

I’m grateful to the graduate students and postdoc that have shared their time with me in ACL. In particular, I would like to thank Mark Cutler and Shih-Yuan Liu for their invaluable support early on and passing down their knowledge of ROS, programming, and paper writing. I want to thank fellow Ford project team members, Andres Hasfura, Steven Chen, and Michael Everett, for their support with the project, and for all those late nights where we worked hard to create awesome demos and videos. I want to thank fellow building 41 members (including Brett Lopez) for helping to make working on robots fun. I also need to acknowledge the support of my small army of UROPs who have helped me to develop all of the hardware for this thesis: Jan Carius, Robert Tran, Alex Gilbert, Wally Wibowo, Victor Zhang, Angel Echevarria, Matt Deyo, Yun Chang, Ronnie Ogden, Rockey Hester, Nicolas Lanzetti, and Emmanuel Akinbo. I also want to acknowledge all the MIT MOD shuttle drivers who helped me collect data, Ian, Samantha, Ricky, Jolly, Anna, Hosea, and Kenneth, as well as all of the anonymous MIT students who used the MOD service in order to
provide me with the data that I needed. I am grateful to the ACL admins Quentin Alexander and Bryt Bradley who ordered and processed every piece of hardware that I ever needed for the project.

I would like to acknowledge the support of Ford Motor Company who sponsored my work through the Ford-MIT alliance, providing me with an interesting research project and the funding to accomplish it.

Finally, I would like to thank those who have supported me outside of the lab over all these years. I want to thank my friends for helping to make the grad school experience fun and I want to thank my family for their constant love and encouragement over all of these years. I am grateful to my loving girlfriend Samantha for her role in my life, for the fun times both home and abroad, and for being by my side through the toughest times. Lastly, I want to thank God for blessing me with the opportunity and abilities needed to study at the greatest engineering school in the world, I owe it all to Him.
# Contents

1 Introduction

1.1 Motivation .......................... 17
1.2 MOD Operating Frameworks .................. 18
1.3 Problem Statement ....................... 20
1.4 Literature Review ........................ 22
1.5 Summary of Contributions .................. 24

2 Background

2.1 Poisson Processes ...................... 27
2.2 Network Graphs ........................ 29
2.3 Parameter Estimation ..................... 32

3 The MIT MOD Testbed

3.1 Design Requirements .................... 36
3.2 Campus Operating Region ................ 38
   3.2.1 MIT Campus ......................... 38
   3.2.2 Golf Cart Shuttles .................. 39
   3.2.3 Network Graph ....................... 40
3.3 Ride Framework ......................... 41
3.4 Autonomous Driving ..................... 42
3.5 Advanced Sensing ....................... 43
   3.5.1 Hardware .......................... 43
   3.5.2 Localization ....................... 44

7
8 Conclusion

8.1 Future Work .......................................................... 129

References ....................................................................... 130

A MOD Sensing Hardware and Software Design .................... 139

A.1 Hardware .................................................................. 139
   A.1.1 Cameras ............................................................... 139
   A.1.2 2D Lidar .............................................................. 140
   A.1.3 3D Lidar .............................................................. 141
   A.1.4 Seat Sensors ....................................................... 142
   A.1.5 Computing .......................................................... 142
   A.1.6 Power and Communication .................................... 145

A.2 Software .................................................................. 147
   A.2.1 Robot Operating System (ROS) .............................. 147
   A.2.2 Hardware Drivers ............................................... 148
   A.2.3 Localization ....................................................... 150
   A.2.4 Pedestrian Tracking ........................................... 156
   A.2.5 Data collection procedure .................................... 160

B Marginalization of Arrival Probability .............................. 165
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Ridesharing example</td>
<td>23</td>
</tr>
<tr>
<td>1-2</td>
<td>Approach overview</td>
<td>24</td>
</tr>
<tr>
<td>2-1</td>
<td>Example traffic network with corresponding network graph</td>
<td>37</td>
</tr>
<tr>
<td>3-1</td>
<td>Map of MIT operating region showing demand locations</td>
<td>43</td>
</tr>
<tr>
<td>3-2</td>
<td>Stock 2015 GEM e4s</td>
<td>44</td>
</tr>
<tr>
<td>3-3</td>
<td>MIT MOD campus operating region with vehicle justification</td>
<td>45</td>
</tr>
<tr>
<td>3-4</td>
<td>Pedestrian network graph</td>
<td>46</td>
</tr>
<tr>
<td>3-5</td>
<td>Full network graph</td>
<td>47</td>
</tr>
<tr>
<td>3-6</td>
<td>App framework for the MIT MOD system</td>
<td>48</td>
</tr>
<tr>
<td>3-7</td>
<td>Driver routing app for MIT MOD system</td>
<td>49</td>
</tr>
<tr>
<td>3-8</td>
<td>GEM shuttle with sensors</td>
<td>51</td>
</tr>
<tr>
<td>3-9</td>
<td>Pedestrian sensing coverage of the MOD vehicles</td>
<td>52</td>
</tr>
<tr>
<td>3-10</td>
<td>Pedestrian tracking framework</td>
<td>53</td>
</tr>
<tr>
<td>3-11</td>
<td>Pedestrian tracking implementation</td>
<td>55</td>
</tr>
<tr>
<td>3-12</td>
<td>Pickup and dropoff locations for the campus MOD system</td>
<td>56</td>
</tr>
<tr>
<td>3-13</td>
<td>Pedestrian trajectories overlayed on the full network graph</td>
<td>57</td>
</tr>
<tr>
<td>3-14</td>
<td>Pedestrian velocities on MIT campus</td>
<td>59</td>
</tr>
<tr>
<td>3-15</td>
<td>Example simulation of the MIT MOD system</td>
<td>60</td>
</tr>
<tr>
<td>3-16</td>
<td>MIT MOD fleet</td>
<td>61</td>
</tr>
<tr>
<td>4-1</td>
<td>Moving observer diagram</td>
<td>65</td>
</tr>
<tr>
<td>4-2</td>
<td>Hardware setup for moving observer experiment</td>
<td>70</td>
</tr>
<tr>
<td>Section</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>----------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>4-3</td>
<td>Moving observer arrival rate estimate comparison</td>
<td>71</td>
</tr>
<tr>
<td>4-4</td>
<td>Arrival rate estimates for pedestrians traveling along additional links</td>
<td>72</td>
</tr>
<tr>
<td>4-5</td>
<td>Arrival rates on each link in the network</td>
<td>74</td>
</tr>
<tr>
<td>4-6</td>
<td>Full network simulation results</td>
<td>75</td>
</tr>
<tr>
<td>4-7</td>
<td>Arrival rate estimates for varied number of link traversals</td>
<td>76</td>
</tr>
<tr>
<td>4-8</td>
<td>Arrival rate estimates for varied true arrival rates</td>
<td>76</td>
</tr>
<tr>
<td>5-1</td>
<td>Two-factor arrival diagram</td>
<td>81</td>
</tr>
<tr>
<td>5-2</td>
<td>Route assignments from the exploration planner. Relative link uncertainities are shown in magenta. Three vehicles (red, blue, green) are each assigned a route to reduce uncertainty. The route origin for each vehicle is marked with a cross.</td>
<td>89</td>
</tr>
<tr>
<td>5-3</td>
<td>Customer arrival rate estimation error over time</td>
<td>90</td>
</tr>
<tr>
<td>6-1</td>
<td>Comparison of expected value and chance-constrained planners</td>
<td>100</td>
</tr>
<tr>
<td>6-2</td>
<td>Comparison of ride hailing planners</td>
<td>103</td>
</tr>
<tr>
<td>7-1</td>
<td>Ridesharing assignment diagram</td>
<td>116</td>
</tr>
<tr>
<td>7-2</td>
<td>Relative customer arrival rates and computed predictive nodes</td>
<td>120</td>
</tr>
<tr>
<td>7-3</td>
<td>Service time comparison of baseline and predictive positioning methods over a range of customer arrival rates</td>
<td>121</td>
</tr>
<tr>
<td>7-4</td>
<td>Service times for unmanaged and predictive positioning methods, with and without ridesharing, over a range of customer arrival rates</td>
<td>122</td>
</tr>
<tr>
<td>7-5</td>
<td>Average customer rating for several fleet management strategies</td>
<td>125</td>
</tr>
<tr>
<td>8-1</td>
<td>MIT MOD vehicle in pedestrian traffic</td>
<td>130</td>
</tr>
<tr>
<td>A-1</td>
<td>Logitech C920 camera</td>
<td>140</td>
</tr>
<tr>
<td>A-2</td>
<td>Sick LMS 151 Lidar</td>
<td>141</td>
</tr>
<tr>
<td>A-3</td>
<td>Velodyne VLP-16 Lidar</td>
<td>142</td>
</tr>
<tr>
<td>A-4</td>
<td>Seat sensor</td>
<td>143</td>
</tr>
<tr>
<td>A-5</td>
<td>Computing and battery</td>
<td>144</td>
</tr>
<tr>
<td>A-6</td>
<td>Laptop location</td>
<td>144</td>
</tr>
</tbody>
</table>
Hardware wiring diagram ........................................ 145
Hardware wiring components .................................. 146
Software overview diagram .................................... 147
Sensor data comparison ......................................... 162
Occupancy grid map for MIT campus ......................... 163
Localization with AMCL ......................................... 163
List of Tables

1.1 Summary of MOD operating frameworks. ........................................ 20

7.1 MOD fleet management performance as measured by average customer rating. ......................................................... 124
Chapter 1

Introduction

Mobility On Demand (MOD) systems are revolutionizing transportation, especially in urban environments. In MOD systems, a fleet of shared vehicles continually serves multiple customers by transporting them from desired pickup locations to desired destinations. By utilizing shared resources, MOD systems promote higher vehicle utilization rates and more sustainable urban land use through reduced parking spaces [65].

MOD usage is growing quickly with the introduction of new app-based ride request systems such as Uber and Lyft. Both companies are seeing increases in requests, with Lyft recently reporting a rise in gross bookings by as much as 25% quarter over quarter [5]. It is estimated that by 2030, as much as 26% of all global miles traveled will be from customers using shared vehicles [4].

Additionally, the introduction of new autonomous MOD systems is expected, where vehicle fleets are composed of self-driving, autonomous vehicles. Autonomous vehicles in general promise to save lives and reduce costs with fewer accidents [8], but their use in MOD systems is also expected to reduce labor costs by removing the need for a driver. Existing MOD providers such as Uber are in active development of autonomous vehicles in order to augment their existing fleets [55]. Furthermore, many vehicle manufactures currently in active development of autonomous vehicles are planning deployment within MOD systems. For example, Ford Motor Company is targeting high volume production of autonomous MOD fleets as early as 2021 [2].
1.1 Motivation

A fundamental goal for MOD systems is providing a high customer quality of service (QoS). Compared to the private automobile ownership experience, MOD customers may experience inconveniences that are introduced from relying on a shared resource. For example, MOD customers can experience transit delays in the form of additional wait time as they wait for a vehicle to pick them up, or additional ride time if the vehicle makes additional stops before dropping them off. Along with the price of the service, the MOD system transit delays are a key factor in the success of MOD systems [44]. The culmination of these factors is the customer QoS, a qualitative factor that specifies how customers perceive their MOD transportation experience. Through proper management of MOD systems, the QoS for MOD customers can be made more attractive than private ownership alternatives [44].

Success for MOD systems depends on the ability to manage the distribution of MOD vehicles under spatial and temporal fluctuations in customer demand [44]. Therefore, MOD system must leverage powerful and accurate demand prediction models to efficiently manage the spatio-temporal distribution of the fleet [44]. There are many modeling paradigms that can be considered for demand prediction, each operating on different timescales.

Reactive models operate on instantaneous timescales by using current ride request information as a measure of customer demand. Example reactive models include real-time rebalancing policies that model customer demand using queue sizes of customers waiting for rides [50, 66]. The advantage of reactive models is that there is little uncertainty in demand because it is already made known. The disadvantage is that reactive models inherently lag behind demand, that is, demand is only made known once customers have arrived and are waiting to be served.

Predictive models operate on long timescales by using past ride request information to extract temporal patterns in customer demand. An example predictive model is the Poisson-spectral model for extracting temporal patterns in [37]. The advantage of predictive models is that they can extract any temporal patterns that exist in the
system and provide predictions based only on the time-of-day, allowing for long-term predictions. A challenge for predictive models is that in order to extract any meaningful patterns, large datasets are required which can difficult to obtain, especially in new environments. One disadvantage is that the models are sensitive to deviations from historical data and can leave un-met demand if the system doesn’t have records of customer arrivals in system locations.

Proactive models operate on short timescales by using real-time data in order to anticipate future customer arrivals. These models attempt to use real-time measurements of data that is reflective of demand to identify demand hotspots before customers arrive. An example proactive model is the demand-supply level model in [56] that correlates real-time taxi turnover times with customer demand in an attempt to reduce customer wait times. The advantage of proactive models is that they can quickly respond to sudden changes in demand and can be used in new environments. A challenge for these models is that they require real-time data acquisition that often requires system-wide sensing for system-wide demand estimation. They also have the disadvantage that customer demand estimates are only valid for the short timescales for which the collected real-time data is valid, requiring that data be collected quickly and continually.

This thesis is motivated by the rise of advanced sensing capabilities of modern vehicles, particularly autonomous vehicles, that can be used to address the challenges associated with proactive customer demand models. By using autonomous vehicles as MOD shuttles, new sensing capabilities are introduced into the MOD system, where the advanced sensors used to automate each vehicle can now double as part of a mobile sensor network. Through a principled approach, sensor data from the mobile vehicles can be used to estimate real-time data in the form of traffic flows on short timescales. A proactive model then correlates real-time traffic data with customer demand to provide a belief in customer demand.

By utilizing the customer demand model, MOD fleets can be managed such that customer QoS is improved. Many modern MOD systems utilize automated fleet management for vehicle repositioning and fleet-wide coordination, especially MOD
Table 1.1: Summary of MOD operating frameworks.

<table>
<thead>
<tr>
<th></th>
<th>Car sharing</th>
<th>Ride hailing</th>
<th>Ride requesting</th>
<th>Ridesharing</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Design</strong></td>
<td>Vehicles wait at hubs</td>
<td>Vehicles hailed by passengers</td>
<td>Rides requested via an app</td>
<td>Customers ride together</td>
</tr>
<tr>
<td><strong>Ideal Customer QoS</strong></td>
<td>Vehicle available at customer's hub</td>
<td>Vehicle in proximity of arrival location</td>
<td>Low pickup wait time</td>
<td>Minimal overall transit delays</td>
</tr>
<tr>
<td><strong>MOD Fleet Management Goal</strong></td>
<td>Rebalance fleet with respect to demand [50,58,64,66]</td>
<td>Assign vehicles to expected arrival locations [9,63]</td>
<td>Minimize wait time for customers [13,27]</td>
<td>Assign customers to vehicles based on QoS [42,47]</td>
</tr>
</tbody>
</table>

fleets with autonomous vehicles capable of operation without a driver [49]. While the fundamental approach is to manage vehicles relative to customer demand, the exact means for which customer QoS can be improved through automated fleet management depends on the underlying MOD operating framework.

### 1.2 MOD Operating Frameworks

MOD in general refers to any framework in which customers do not own vehicles for transportation, but rather have on-demand access to rides whenever needed. However, the means for which customers request and receive rides can vary depending on the specific type of MOD framework. The distinction for how customers access an MOD system is important for understanding how customer QoS is affected by system operation. The following provides an overview of the details for several MOD frameworks with specific consideration on factors that affect customer QoS and the goals for effective MOD fleet management.

MOD operation can be classified into car sharing, ride hailing, ride request, and ridesharing frameworks, a summary for which is provided in Table 1.1.

**Car Sharing** In car sharing frameworks, customers walk to an origin hub to pick up a vehicle and then drive to a destination hub to drop it off. Examples of car sharing
frameworks include one-way car rental services such as Zipcar and city bike sharing services such as Hubway. Typically, car sharing frameworks will utilize large vehicle fleets that are distributed across a relatively few number of hubs. The benefit of car sharing is that vehicles only need to be stored and routed between hubs, circumventing the need to have vehicles at every exact customer arrival location. The drawback is that customers must first travel to a hub to receive a ride, introducing an additional transit delay. The customer QoS metrics for car sharing frameworks typically focus on the availability of vehicles at hubs [50, 58, 64, 66]. A binary measure of QoS can be used, where an arriving customer either does or does not find access to a vehicle. Therefore, the challenge for MOD fleet management approaches is to ensure that there is always a vehicle available for an arriving customer. As vehicles service customers, vehicles will be moved across the network and may be far from the next customer arrival location, causing an imbalance between future customer demand and fleet locations. The goal is then to design a car sharing planner that rebalances vehicles across hubs based on expected customer arrivals.

**Ride Hailing** In ride hailing frameworks, customers do not travel to a hub but are rather picked up at their exact arrival location from a nearby waiting or passing vehicle. A request for a ride is made simply by hailing (waving down) a vehicle as it passes, requiring that the vehicle and arrived customer be within line-of-site proximity to one another. The most common example of ride hailing frameworks are taxi services that allow for street hailing. Like car sharing, ride hailing customer QoS is a binary measure that depends on whether or not the customer successfully hailed a ride. The challenge for MOD fleet management is ensuring that an MOD vehicle is within proximity of a customer arrival location, where the number of possible arrival locations often exceeds the number of vehicles in the fleet. As a result, the goal of a ride hailing planner is to selectively place vehicles in sufficient proximity of expected customer “hotspots”, as identified by a customer demand model.
**Ride Requesting** In ride request frameworks, customers provide requests in the form of a pickup and dropoff location and a vehicle is routed directly to their arrival and destination locations. Modern ride requesting examples, such as Uber and Lyft, utilize a smartphone app capable of specifying precise address locations to a centralized server. Basic ride request frameworks provide a one-to-one service, where vehicles serve only one customer at a time. Because the customer’s request is pushed to the MOD system, customer QoS no longer depends on whether a customer will receive a ride, but rather on how long they wait to receive the ride. A customer may have to wait for a vehicle to become available if all vehicles are currently in use. Even if a vehicle is available, the customer may have to wait for the vehicle to travel from its current location to the customer’s pickup location. The challenge for MOD fleet management is that many or unexpected customer arrivals will lead to large customer wait times. One goal for a ride request planner could be to ensure that there are enough vehicles available in the system to always accommodate customer demand. However, given a fixed number of available vehicles, the goal is instead to minimize the wait time customers experience by placing vehicles close to expected customer arrival locations.

**Ridesharing** In ride request frameworks, if demand is high, it may not be possible to ensure there are enough vehicles to serve every customer on a one-to-one basis due to limited fleet sizes. Ridesharing frameworks are an extension of ride request frameworks where multiple customers may be serviced simultaneously by sharing a ride. Examples of ridesharing frameworks include airport shuttle services such as the SuperShuttle and app-based services such as Uber Pool and Lyft Line. Ridesharing reduces customer wait times by allowing new customers to be picked up before previous passengers have been dropped off. However, a trade off is introduced between reduced wait times for new customers and increased ride times for current customers. Customer QoS now depends on several factors such as wait time, ride time, the number of stops the vehicle makes, etc., and the reduction of one factor typically leads to the increase of another. Figure 1-1 shows an illustration of how a ridesharing customer may
1.3 Problem Statement

The goal of this work is to develop algorithms for demand estimation and MOD fleet management in order to improve customer QoS. The focus for demand estimation is on the development of a proactive customer demand model that uses real-time traffic data for short-term predictions of customer arrival locations. A challenge for the proactive model is measuring real-time traffic data on a system-wide scale on short timescales. To address this, first real-time traffic arrival rate data is estimated using sensors onboard MOD vehicles. Traffic arrival data is then correlated with ride requests using
a proactive model to estimate expected demand hotspots in the form of customer arrival locations. Through automated fleet management, vehicles as active sensors can be specifically routed to perform active sensing in order to further improve customer demand estimates. In addition to active sensing, MOD fleet management can be used to ensure that customer’s experience a high QoS. This work focuses on the challenges associated with ride hailing and ride requesting frameworks for finding customers and mitigating transit delays. To address those challenges, fleet management planners utilize customer demand estimates from the proactive model to ultimately improve customer QoS. An overview of the approach for this thesis is illustrated in Figure 1-2.

**Real-time Traffic Estimation** Traffic data is quantified in terms of arrivals (flows) in a network graph representation of an MOD operating environment. In particular, the work in this thesis focuses on campus MOD systems composed primarily of pedestrian traffic. The traffic estimation problem considers how data collected from autonomous vehicles can be used to estimate Poisson arrival rates of pedestrian traffic. In autonomous MOD systems, each vehicle is equipped with a suite of real-time sensors in the form of cameras, Lidar, and GPS that enable it to drive autonomously. The first objective is to build and evaluate a physical framework for recording pedestrian movement using these onboard sensors. Pedestrian observations in the form of
trajectory data is used to estimate arrival rates along links in a network graph. For static sensors, arrival rate estimation could be trivially performed by counting the number of passing pedestrians in a given time. For mobile sensors, however, pedestrian counts and observation times are dependent on the vehicle motion. For example, a vehicle traveling alongside traffic will have different pedestrian counts if it travels with or against the pedestrian flow. The objective is to decouple the vehicle motion from the pedestrian observations in order to accurately measure pedestrian arrival rates. The goal is to demonstrate that real-time arrival rate estimation from already available mobile sensors on MOD shuttles is comparable to that of costly static sensors.

**Customer Demand Estimation**  
Customer demand is quantified as the number of MOD customers that will arrive at each location within an MOD system. Prediction of customer demand is achieved by estimating the Poisson arrival rates of customers at each location. Arrival rate estimation is performed through a proactive model that correlates real-time traffic data with customer arrivals. The first challenge is therefore determining the proper correlation between traffic and customer arrivals, ideally with quantification of uncertainty. The second challenge is obtaining system-wide observability. Fine-grained MOD network graphs typically have a large number of possible arrival locations with relatively few vehicles available for observing traffic data at each one. A trial-and-error approach to assigning vehicles to park and wait sequentially at every potential arrival location would be slow and result in missed customers. The challenge is that of performing active sensing within the MOD network, where sensing vehicles are specifically routed to provide more accurate customer demand estimates.

**MOD Fleet Management**  
Given accurate predictions of customer demand, the next challenge is that of managing MOD fleets to best improve customer QoS, the requirements for which will vary depending on the underlying MOD framework. Of the many MOD frameworks, autonomous car sharing is potentially the least likely to be used in practice, as automatic positioning capabilities will likely obviate the
need for customers to travel to a hub. This thesis instead focuses on autonomous fleet management for ride hailing, ride requesting, and ridesharing frameworks. In ride hailing environments, customers do not specify pickup locations but rather hail a nearby passing vehicle. Unless a vehicle is in proximity of a customer, the customer will not be served. The challenge becomes that of estimating customer arrival locations and coordinating the vehicle fleet in the presence of estimation uncertainty in order to minimize lost customers. In ride request environments, as vehicles service customers, vehicles will be moved across the network and may be far from the next customer arrival location, resulting in large customer wait times. The challenge becomes that of using estimated customer demand to position unallocated vehicles in the MOD fleet such that customer wait times are minimized. In ridesharing environments, vehicle capacities are increased and different customers may be serviced simultaneously. The first challenge is that of understanding how tradeoffs in customer transportation metrics, such as reduced wait times for new customers and increased ride times for current customers, affect overall customer QoS. Naive assumptions in customer preference can lead to poor QoS if customers are improperly allocated to vehicles. Once a model of customer QoS is obtained, the challenge becomes that of efficiently assigning customers to vehicles to ensure that a high overall QoS is maintained.

1.4 Literature Review

This section highlights previous work in the domains of traffic sensing, customer demand estimation, and autonomous MOD fleet management.

Traffic Sensing

Automatic traffic monitoring techniques traditionally utilize stationary sensors such as cameras, pneumatic tubes, and magnetic loop counters [3]. For pedestrian networks, traffic flow rates are obtained through similar means such as cameras and turnstiles. Stationary sensors can measure temporal variations in traffic, but only at the locations where they are installed which can make network-wide coverage cost prohibitive for
large networks.

Mobile sensor methods have been proposed using vehicles for traffic monitoring under two main variations: probe vehicle methods and moving observer methods. Probe vehicle methods measure traffic stream characteristics by placing a probe vehicle in the flow of traffic [17,19,61]. Only statistics of the probe vehicle are actually measured, which limits the types of traffic data that can be inferred and makes the method unsuitable for measuring arrival rates. Probe methods perform poorly for vehicles operating in pedestrian traffic networks because vehicle movement is not necessarily reflective of pedestrians traffic movement.

Moving observer methods can measure arrival rates by making manual counts of other vehicles or pedestrians as the observer vehicle drives along a link in the network graph. Previous work on counting methods use manual counts of other vehicles, typically the number of vehicles that are overtaking or are being overtaken by the moving observer, as the vehicle drives a link in either both directions [59] or a single direction [45]. The methods are performed manually and are limited in the amount of data that can be collected because the algorithms require vehicles to traverse entire links to obtain arrival rate estimates. The idea of using autonomous vehicle sensors to estimate arrival rates is presented in [54], but the focus of the work was on developing the detection capabilities of the vehicle and arrival rate estimation methods were not addressed.

**Customer Demand Estimation**

Many arrival process are modeled using Poisson process, which is a prominent customer arrival modeling approach used in MOD systems. Many autonomous car sharing approaches [50,58,64,66] use Poisson arrival rates to quantify customer arrivals at hubs within a network. These methods focus on the fleet management approaches and assume accurate knowledge of customer arrival rates based on historical taxi log data. In [43], customer arrivals are not modeled through a prescribed distribution but rather through generalized uncertainty sets. This approach does not require assumptions about Poisson distributed arrivals, but the bounds on the uncertainty sets will be
dependent on choice of confidence level and data availability, where arrival data is again obtained from historical taxi logs.

Non-autonomous ride hailing approaches study customer demand estimation for taxi MOD systems. In [18, 23, 40, 51], temporal demand estimation methods are proposed based on historical data. A benefit to non-autonomous MOD systems is that human taxi-drivers provide initial knowledge on customer demand locations, which can then be studied to estimate future demand. In autonomous MOD systems operating in ride hailing frameworks, if there are no knowledgeable human-taxi drivers to find customers in the first place, then historical databases may be data deficient, causing demand to remain unknown. In [9, 56], customer demand hotspots are estimated in real-time using large-scale taxi fleets that determine demand based on the likelihood of finding a customer. Demand is specified either for city-scale areas [56] or for customers waiting at taxi stands [9], where it is assumed that customers will be willing to wait for a taxi to pass by. If customers are not willing to wait for vehicles, but rather choose alternative transportation if no vehicle is waiting, then data composed only of customer arrivals will not be a sufficient measure of true customer demand.

MOD Fleet Management

MOD fleet management has been traditionally studied for non-autonomous MOD systems through taxi dispatch and recommendation systems, with recent research interests focusing more on planners for autonomous MOD vehicles. Most of the literature on autonomous fleet management has focused on car sharing frameworks. Literature regarding other MOD frameworks (ride hailing and ride requesting) remains largely in the domain of non-autonomous MOD fleet planners.

Autonomous Fleet Management  Previous works on autonomous MOD fleet management have focused on the car sharing challenge of determining how many vehicles are needed and how to rebalance (reposition) vehicles to ensure arriving customers always have access to a vehicle. In [50], rebalancing flow rates are determined and used to find the minimum number of rebalancing vehicles to bound the number
of waiting customers. A similar approach in [58] determines rebalancing Markov transition probabilities to find the minimum number of vehicles to ensure that all arrivals can be met while minimizing wait times. In [66], a queuing theoretic approach is used to find rebalancing rates in the form of virtual passengers arrival rates. The method finds both the policy and minimum number of rebalancing vehicles needed to ensure that vehicles are available uniformly throughout the network. In [64], a rebalancing policy uses Model Predictive Control to optimize over several objective functions, such as the number of waiting customers, the number of rebalancing vehicles, and the state of charge of the vehicles. These approaches assume a large fleet size is available in order to cover each station and do not extend well to more complex MOD frameworks where the number of possible nodes can quickly outgrow the number of vehicles.

**Ride Hailing**  Ride hailing planners focus on ensuring that vehicles are in proximity of possible customer arrival locations, where there are typically more arrival locations than available vehicles. Previous work on ride hailing planners do not consider autonomous MOD systems but rather take the form of taxi recommendation systems. In [28], a taxi recommendation system provides a sequence of potential pick-up points to drivers in order to minimize the distance that is likely to be driven before finding a customer. In [62,63], behaviors discovered from high-profit taxi drivers are used to make recommendations that maximize the profit of recommended taxi drivers. In [9], a recommendation smartphone application is used to direct taxi drivers to nearby hotspots in an attempt to balance fleet-wide supply and demand. Taxi-based planners typically focus on recommendations for individual taxi drivers and do not consider fleet-wide coordination that is guaranteed in autonomous systems.

**Predictive Positioning for Ride Requesting**  Ride requesting planners with one-to-one assignments (without ridesharing) consider customer wait times when planning how vehicles should be repositioned throughout the network. Real-time predictive positioning of autonomous vehicles to minimize wait time for ride requests is not
common in literature. However, the concept is very similar to the ambulance location and relocation problem that positions ambulances with respect to expected emergency demand. A survey of ambulance location models is available in [13]. The ambulance location problem is different from MOD positioning due to its focus on ensuring demand coverage, where the radius of coverage for each ambulance is prescribed by government standards. While coverage standards can be specified for MOD systems (minimum wait times), meeting those standards may require excessively large fleets. In MOD predictive positing, the problem is inverted: given a set of available vehicles, what is the best obtainable coverage.

**QoS in Ridesharing**  Ride request planners with ridesharing must include additional considerations on how individual customers should be assigned to each vehicle. There are many factors that compose customer QoS that need to be considered by the ridesharing planner. MOD ridesharing is traditionally formulated as a Dial-A-Ride Problem (DARP), which is a specialization of the Vehicle Routing Problem, formulated specifically for allocating customers to vehicles. The DARP has traditionally been applied to door-to-door transportation of elderly or disabled people, where the static DARP is used to make passenger assignments well in advance to vehicle operation [21]. The DARP is NP-Hard and exact mathematical formulations, especially those with customer QoS criteria included, can be complex and difficult to solve exactly [42]. Many approaches focus more on techniques for exact DARP solutions, with less focus on proper consideration for customer QoS. For example, in [7] an exact solution to the DARP is used by formulating a mixed integer linear program with customer QoS included in the cost function simply as a piecewise-linear customer impatience function. To reduce computation time and provide more generalization in QoS constraints, heuristic methods have been proposed taking the form of either integer program formulations or scheduling problem formulations. Integer program formulations using a weighted sum of QoS metrics as the cost function and a large number of feasibility constraints have been solved using genetic algorithms [36], simulated annealing [42], and tabu search [48] heuristic approaches. In each approach, the cost function
weights and constraint formulations are predefined according to assumed customer preference. Scheduling problem formulations enumerate the possible ways of inserting new passengers into the customer schedule of existing vehicles and evaluate each new schedule according to a cost function and a set of constraints. In [26] customers are assigned to the nearest vehicle for which a set of predefined customer QoS constraints are satisfied, and the fleet is sized to ensure all customers can be assigned. In [22, 35], passengers are required to specify either pickup or drop off time windows that serve as additional constraints when evaluating a customer QoS based cost function. In addition to specifying any cost function weights, these methods must also specify customer QoS thresholds for the constraints. The main drawback with all of these approaches is that the formulation that encodes the QoS metrics could be wrong or the choice of weights and constraint thresholds that define customer preference may be chosen incorrectly. If customers perceive a different QoS preference than was assumed by the assignment algorithm, true customer QoS will suffer.

1.5 Summary of Contributions

The contributions of this thesis address challenges in the field of MOD systems with respect to estimating customer demand and performing automated fleet management. The research contributions of this thesis are:

1. Introduction of a mobile-sensing framework for estimating network-wide traffic arrival rates; Chapter 4. Traffic arrival rates are estimated using camera and Lidar sensors onboard MOD vehicles. Challenges associated with coupling between vehicle motion and traffic observations are addressed through a novel moving observer method. Experimental testing demonstrates that the moving observer method achieves comparable arrival rate accuracy to that of pervasive stationary counters, allowing for network-wide sensing with mobile sensors.

2. Development of a customer demand estimation framework that incorporates real-time traffic sensing; Chapter 5. Estimates of customer arrivals are made
using a two-parameter approach that combines traffic arrival rates with observed customer arrival fractions. A Bayesian framework utilizes the data available from the moving observer method and customer arrivals to make online, recursive updates to the demand model. An active sensing planner is introduced for utilizing the customer demand model to specifically route vehicles in an MOD network graph in order to reduce uncertainty. Experimental testing shows that the modeling and active sensing approaches successfully improve customer arrival rate estimation accuracy on short timescales when compared against baseline approaches that focus either on waiting for customers or continual exploration.

3. Formulation of MOD fleet management planners for improving customer QoS in ride hailing, ride request, and ridesharing operating frameworks; Chapters 6 and 7.

For ride hailing, an MOD fleet management framework is introduced that addresses the challenges of utilizing customer demand in order to maximize the number of served customers. Uncertainty in future customer arrival locations is addressed through formulation of a chance-constrained planner. Experimental testing of the chance-constrained ride hailing planner demonstrates a significant improvement in the number of served customers in the MOD system over a baseline exploration approach.

For ride requesting, predictive positioning and ride hailing MOD fleet management frameworks address the challenges of minimizing customer wait time and maximizing overall customer QoS. A predictive positioning planner is introduced that uses customer arrival rate information to position vehicles at key nodes in the MOD network graph that minimize the expected customer wait time. Through experimental testing, the predictive positioning approach is shown to effectively reduce customer service times when compared against a baseline planner that does not reposition vehicles after serving customers. A ridesharing planner is introduced for assigning customers to vehicles such that the ride metrics customers experience will result in a high perceived customer
QoS as indicated through rating feedback. Rather than assume a customers’ QoS preference, a customer ratings model trained on 5-star ratings feedback is used to predict the mapping between customer ride metrics and perceived QoS. An insertion-based ridesharing planner utilizing the novel customer ratings model is shown to provide more robust customer rating performance over a range of unknown customer preferences when compared against ridesharing planners that assume a pre-defined customer preference.

4. Design and implementation of an experimental campus MOD test-bed framework; chapter 3. A hardware implementation of a real-world MOD system is developed for the MIT campus. Three manually-driven, four-passenger golf cart shuttles are equipped with a set of sensors that are most commonly found on autonomous vehicles and operate within ride hailing and app-based ride request frameworks. The advanced capabilities of the MIT MOD framework are used for the purposes of developing MOD insights, collecting real-world data, and performing experimental evaluation of demand modeling and fleet management algorithms.
Chapter 2

Background

This chapter presents background material for the work in this thesis. Specifically, it covers Poisson process probability distributions, network graphs for transportation systems, associated network graph parameters, and parameter estimation techniques. Poisson arrival rate parameters are used in Chapters 4 and 5 to quantify and estimate both traffic and customer arrivals in terms of locations and rates. Network graphs and associated parameters are used throughout the thesis to provide a foundation for developing fleet management strategies.

2.1 Poisson Processes

Many random events occurring within a transportation system can be modeled using counting processes. A counting process captures the random number of occurrences $N(t)$ that have occurred at time $t \geq 0$. $N(t)$ is a nonnegative integer, is nondecreasing in $t$, and is right-continuous, i.e. the number of events in the interval $\tau = (s, t]$ is given by $N(\tau) = N(t) - N(s)$.

A Poisson process is a counting process where the the number of events in $\tau$ is Poisson disturbed, i.e. $N(\tau) \sim \text{Poisson}(\lambda \tau)$ and

$$P(N(\tau) = k) = e^{-\lambda \tau} \frac{(\lambda \tau)^k}{k!},$$  \hspace{1cm} (2.1)
where $\lambda$ is the rate parameter that describes the process.

Poisson processes have many useful properties. First, independent Poisson processes can be merged into a single Poisson process through the superposition property. That is, given two independent Poisson processes, $N_1(t)$ and $N_2(t)$ with rate parameters $\lambda_1$ and $\lambda_2$, their sum $N(t) = N_1(t) + N_2(t)$ will also be a Poisson process with rate parameter $\lambda_1 + \lambda_2$. Similarly, a single Poisson process can be split into independent Poisson processes through the thinning property. That is, given a splitting fraction $p$, a Poisson process $N(t)$ with rate parameter $\lambda$ can be split into independent Poisson processes $N_1(t)$ and $N_2(t)$ with rate parameters $\lambda_1 = p\lambda$ and $\lambda_2 = (1-p)\lambda$.

The thinning and superposition processes can be further applied to determine the order of events in independent Poisson processes [31]. Given two independent Poisson processes $N_1(t)$ and $N_2(t)$ with rate parameters $\lambda_1$ and $\lambda_2$, the probability that $N_1(t) = n$ occurs before $N_2(t) = m$ occurs is given by

$$
\sum_{k=n}^{n+m-1} \binom{n+m-1}{k} \left( \frac{\lambda_1}{\lambda_1 + \lambda_2} \right)^k \left( \frac{\lambda_2}{\lambda_1 + \lambda_2} \right)^{n+m-1-k}.
$$

### 2.2 Network Graphs

A network graph can be used to represent the connectivity of a transportation system in order to describe how traffic (pedestrians, bikers, cars, etc.) moves throughout the system. Figure 2-1 shows a an example section of a road traffic network with a corresponding network graph.

**Nomenclature**

A directed network graph $\mathcal{G}$ is composed of a set of nodes $\mathcal{N}$ and a set of links $\mathcal{L}$. The set of nodes $\mathcal{N} = \{n_1, \ldots, n_{N_n}\}$ contains $N_n$ nodes that represent the regions where traffic can enter, leave, or transition through the traffic network. The set of links $\mathcal{L} = \{l_1, \ldots, l_{N_L}\}$ contains $N_l$ directed link edges, each taking the form of an ordered pair of neighbor nodes, $l = (n_{o(l)}, n_{d(l)}) \in \mathcal{N}^2$, where $o(l)$ and $d(l)$ represent the respective origin and destination node indexes of each link. The links represent
Figure 2-1: Example traffic network with corresponding network graph. The 6 shaded, numbered circles represent nodes where traffic may enter, exit, or transition through the network and the 6 bi-directed arrows represent the 12 directional links that indicate how traffic moves between the nodes.

how traffic moves between node regions, with the constraint that once traffic begins traveling along a link, it must travel to the destination node. If there are regions along a link for which that constraint does not hold true, i.e. traffic could exit the link at that location, then the link would instead be split into an additional set of nodes and links with an additional node at the exit location.

It is also useful to consider routes, that represent the complete travel path from origin to destination within the network graph. A route $r(n_{o(r)}, n_{d(r)})$ is defined as a sequence of directed links $L_r \subseteq L$ that corresponds to a unique minimum-travel-time path between origin node $n_{o(r)}$ and destination node $n_{d(r)}$, where $o(r)$ and $d(r)$ are indices corresponding to the origin and destination of $r$.

**Network Arrivals**

Arrivals within a network graph are defined in terms of node arrivals, link arrivals, or route arrivals, the distinction of which is important for parameter estimation discussed in Chapters 4 and 5. A node arrival occurs for a specific node whenever traffic appears within the node region, whether due to transitioning from another node or entering the network for the first time. A link arrival occurs for a specific link whenever traffic leaves the origin node of that link and begins to travel along the link towards the destination node of the link. A route arrival occurs for a specific route when traffic arrives in the network for the first time at the origin node of the route, travels along each link in the route, and leaves the network at the destination node of the route.
Arrivals in the network graph are modeled as route arrivals that occur according to Poisson processes. Poisson route arrivals will induce both link and node arrivals as traffic moves along the routes. Due to the superposition and decomposition properties of Poisson processes, nodes and link arrivals will also occur according to Poisson processes. For a route \( r \) that experiences arrivals according to rate \( \lambda_r \), the arrivals for link \( l \) along that route will occur with arrival rate parameter \( \lambda_l \) such that

\[
\lambda_l = \sum_{r: \ l \in \mathcal{L}_r} \lambda_r,
\]

where the summation is over all routes that include \( l \). Similarly, node arrivals will be Poisson distributed as a superposition of all link arrivals originating from that node. The arrivals for node \( n \) will occur with arrival rate parameter \( \lambda_n \) such that

\[
\lambda_n = \sum_{l: \ n = n_{o(l)}} \lambda_l,
\]

where the summation is over all links that have \( n \) as the origin node.

### 2.3 Parameter Estimation

This section covers maximum likelihood and Bayesian parameter estimation for Poisson and Bernoulli distributions that will be used in Chapters 4 and 5.

Arrivals within the transportation network follow Poisson processes that are defined by an arrival rate parameter \( \lambda \). Observation data for Poisson processes takes the form of the number of arrivals, \( m \), observed over a time period of \( \tau \). Given a sample of \( N_o \) independent arrival counts and corresponding observation time periods, the goal is to estimate \( \lambda \) of the Poisson process from which the samples were drawn.

One approach is to use the maximum likelihood estimator (MLE) which is an efficient, unbiased estimator [25]. First, the sample data can be aggregated into sufficient statistics for total arrival counts, \( N_e = \sum_{i=1}^{N_o} m_i \), and total count observation
time $T_c = \sum_{i=1}^{N_c} \tau_i$. The MLE is,
\[
\hat{\lambda}_{\text{MLE}} = \frac{N_c}{T_c}.
\] (2.5)

Confidence interval bounds can be used to provide a measure of estimate accuracy. Lower and upper confidence interval bounds, $\lambda_L$ and $\lambda_U$, specify a $100(1 - \alpha)$ confidence level that the true value is within the bounds, where $\alpha$ is chosen based on the desired confidence level. The bounds are,
\[
\lambda_L = \frac{X^2_{\alpha/2}(2N_c)}{2T_c},
\] (2.6)
\[
\lambda_U = \frac{X^2_{1-\alpha/2}(2N_c + 2)}{2T_c},
\] (2.7)

where $X^2_p(\nu)$ is the $p$-th quantile of the $\chi^2$ distribution with $\nu$ degrees of freedom.

An alternative approach is to use a Bayesian estimator which uses prior belief to provide a full probability distribution on the parameter itself. A common approach is to model the parameter using a distribution that is conjugate to the likelihood distribution. For data that is Poisson distributed, a conjugate prior for the rate parameter is the Gamma distribution, i.e.
\[
\lambda \sim \text{Gamma}(\alpha; \alpha, \beta)
\]
and
\[
P(\lambda = x) = \frac{\beta^\alpha}{\Gamma(\alpha)} x^{\alpha-1} e^{-\beta x},
\] (2.8)

where $\Gamma(\cdot)$ is the Gamma function and $\alpha$ and $\beta$ are hyperparameters that describe the distribution of $\lambda$. Prior belief on the distribution of $\lambda$ is specified using values $a_0$ and $\beta_0$. Estimation of the arrival rate parameter is performed by using the number of arrivals $m$ observed over a time period of $\tau$ to update the hyperparameters. Because the Gamma prior distribution is conjugate to the Poisson, the posterior distribution with updated hyperparameters will also be Gamma distributed, with parameters $\alpha = a_0 + m$ and $\beta = \beta_0 + \tau$. The updated Gamma distribution completely describes the parameter, where the mean or median can be used as a point estimate, and the variance can be used as a confidence measure.

This thesis also utilizes Bernoulli distributed random variables. A Bernoulli
distribution models the probability that a binary quantity \( x \) takes value either 1 or 0. The probability is determined using a fraction parameter \( p \), that is \( x \sim \text{Bernoulli}(p) \) such that \( P(X = 1) = p \) and \( P(X = 0) = 1 - p \). Estimation of the fraction parameter uses a sample of \( N_o \) observations of \( x \) in the form of the number of observed “successes”, \( s = \sum_{i=1}^{N_o} x_i \), and the number of observed “failures”, \( N_o - s \). A Bayesian estimator can be used to estimate \( p \) with a Beta distribution that is a conjugate prior for a Bernoulli likelihood model. The fraction parameter is modeled as Beta distributed with hyperparameters \( a \) and \( b \) such that
\[
P( p = x ) = \frac{x^{a-1}(1 - x)^{b-1}}{B(a, b)},
\]
where \( B(\cdot, \cdot) \) is the Beta function. Prior beliefs for the distribution of \( p \) are specified using prior values \( a_0 \) and \( b_0 \). The posterior distribution for \( p \) will be Beta distributed with updated hyperparameters \( a = a_0 + s \) and \( b = b_0 + N_o - s \).
Chapter 3

The MIT MOD Testbed

This chapter describes the design and development of an experimental MOD testbed framework to be used on MIT campus. First, the research objectives of the thesis are used to generate a list of design requirements for the MOD framework. The implementation of the physical MOD system is then presented with regard to each of the design requirements including design choices and justification. Finally, the capabilities of the MIT MOD system as a whole are quantified and further used to develop a data-driven simulation that allows for experimental testing of the algorithms presented in Chapters 4 to 7.

3.1 Design Requirements

The goals of this thesis is to improve customer quality of service through demand estimation and fleet management of autonomous MOD systems. Future autonomous MOD systems will be composed of large fleets of fully autonomous vehicles operating over city-wide scales. However, achieving this level of capability and coverage requires significant technological development beyond the scope of this thesis. Instead, a smaller-scale MOD system is proposed that serves as a microcosm of larger transportation systems, where the fundamental properties and capabilities of autonomous MOD operation are maintained. The properties and capabilities that are the focus of this thesis are listed as follows:
1. Operating Region: The MOD operating region is required to contain areas for which there is sufficient demand for operation. The region should include popular origin and destinations as well as the ability to function as a first/last mile service by including public transportation stops.

2. Ride Request Framework: The MOD system requires that customers have on-demand access to rides, that is, there is a means for requesting and receiving rides directly from the MOD system.

3. Autonomous Driving: The MOD system is required to have an automated planner for routing vehicles. The key aspect is that the routing of autonomous vehicles is managed through automated planning algorithms instead of driver decisions.

4. Advanced Sensing: The MOD system is required to provide traffic tracking using advanced sensors such as those commonly found on autonomous vehicles. Typically, autonomous vehicle sensors are used for obstacle tracking for collision avoidance. This work instead uses obstacle tracking for modeling traffic flows through the MOD system.

Requirements 1 and 2 are standard for any MOD system, while requirements 3 and 4 are more unique to the study of autonomous MOD systems. Each of these four requirements are satisfied by the MIT MOD system.

3.2 Campus Operating Region

3.2.1 MIT Campus

The MOD research framework was developed on MIT campus with the goal of transporting MIT students, faculty, and staff. MIT campus provides several high-demand locations consisting of university offices and classrooms as well as nearby businesses and restaurants. The campus contains public transportation hubs in the form of subway and bus stops, allowing the MOD system to serve as a first/last
Figure 3-1: Map of MIT operating region showing demand locations. MIT's campus contains several high-demand locations including classrooms, public transportation stops, and restaurants. Some (but not all) of these locations are labeled on the map.

mile service. Figure 3-1 shows the operating region and demand locations for the MIT MOD system. Many of these demand locations are internal to the campus along pedestrian walkways that are generally closed to public vehicles. As such, a distinguishing feature of this campus MOD system is the choice of vehicles that allow for travel along pedestrian pathways, allowing for faster routes and better access to buildings.

### 3.2.2 Golf Cart Shuttles

Due to the requirement for MOD vehicles to drive on pedestrian walkways, the form factor of the shuttle was chosen to conform to what would be most acceptable to the MIT community. The shuttles are based on the golf carts that are already widely used on MIT campus by MIT facilities, allowing for the shuttles to move throughout campus with as much maneuverability and minimal disruption as MIT facilities. The MOD
fleet is composed of three 2015 Polaris GEM e4s electric vehicles that have a top speed of 25 mph and can carry 4 passengers including the driver. Figure 3-2 shows a stock 2015 GEM e4s vehicle. The vehicles are registered as low-speed vehicles and can legally drive on public roadways with speed limits up to 35 mph. Additionally, the vehicles are authorized by MIT to drive on pedestrian walkways that are wide enough to accommodate both the shuttle and pedestrians. Figure 3-3 shows the public roadways and internal pedestrian walkways for the MOD system and demonstrates that the golf cart shuttles allow for simultaneous usage of walkways alongside pedestrians.

### 3.2.3 Network Graph

Two network graphs were created for the campus operating region. The first network graph focuses only on the subset of the campus consisting of pedestrian walkways (walkways 1-7 of Figure 3-3a). Figure 3-4 shows the pedestrian network graph that was constructed using knowledge of the campus including doorways, walkways, and intersections as well as empirical observations of how pedestrians move through campus. Nodes are placed at locations where pedestrians enter or exit the region either through a building or through transitioning to bordering regions. Nodes are also placed at intersection locations where pedestrians change walking direction. Directed links are then assigned between each pair of nodes corresponding to a valid walkway, with separate links for each direction of travel.

The second network graph focuses on the full operating region of campus. Figure 3-
Figure 3-3: (a) shows the MIT MOD campus operating region that is composed of both pedestrian walkways (orange) and vehicle roadways (blue). (b) demonstrates that golf cart shuttles can justifiably operate alongside pedestrians along each of the numbered walkways in (a).

5 shows the *full network graph* that was constructed similarly to the first but with additional consideration for the connectivity of vehicle roadways. The broader full
network graph consolidates nodes from the finer pedestrian network graph to reduce complexity. For example, the graph shown in Figure 3-4 models pedestrians as entering the network at node 15 and immediately traveling to node 14 before transitioning. The graph shown in Figure 3-5 instead combines those two nodes into a single node at 13, where pedestrians can enter and immediately transition to several other nodes. Likewise, some of the nodes that receive little traffic (i.e. nodes 9 and 10 of Figure 3-4) are removed from the full network graph, where the negligible difference in modeling accuracy allows savings on model complexity.
Figure 3-5: Full network graph. The network graph was created for the full operating region on MIT campus. The network graph is composed of 33 nodes and 106 directed links.

### 3.3 Ride Framework

A ride request framework was created to allow for “customers” (MIT students and faculty) to request rides from the shuttles. The MOD service is provided free of charge. Customers can request rides either by hailing a vehicle or by sending a ride request using a custom app interface. Customers access the ride request app by visiting a website and then use the interface to select a pickup and dropoff location along nodes in the network graph. Figure 3-6 shows the app interface used for customer requests.

The ride request framework consists of 3 major components: the user interfaces, the backend server, and the database. The user interfaces consist of passenger and driver web applications. The use of web apps allows for access from most major devices including smartphones and PCs. The apps communicate with a python-based backend server to perform more complicated computations such as routing vehicles to
passenger requests as well as handling more secure transactions such as usernames and passwords. Finally, the apps communicate with a database that stores data regarding the locations of the vehicles and passenger-vehicle assignments. A graphic summarizing the app framework components is shown in Figure 3-6

3.4 Autonomous Driving

At this time, there are no commercially available autonomous vehicles for use as transportation shuttles. Therefore, the MIT MOD system requires adding autonomous capabilities to stock golf cart shuttles. To bring the automated driving capabilities to the golf carts, there are two options: human actuation or robotic actuation. Human actuation uses human drivers for manually actuating the vehicles in terms of control and path following, but uses automated algorithms for routing drivers. Robotic actuation uses electro-mechanical motors, micro-controllers, and sensors to automate all aspects of driving. There are many open challenges for creating a fully autonomous vehicle that can navigate reliably in complex campus environments without human intervention, many of which are fundamental challenges in the field of robotics. However, the design requirement for the experimental autonomous MOD system is that vehicles need only be routed according to an automated planner. To avoid the open challenges
associated with low-level control, navigation, and passenger interactions required for robotic actuation, the MIT MOD system uses human actuation in conjunction with a centralized automated planner.

Automated driving is accomplished by having an automated planner within the ride request framework provide routing information to human drivers through a tablet mounted in the vehicle. The backend server takes in ride request information and computes routes for each of the vehicles in the network which are pushed to the database. A driver app pulls the routing information from the database and presents the information to the driver. The driver interface is shown in Figure 3-7. In addition to driving the routes prescribed by the app, drivers handle passenger interaction tasks such as responding to directly hailed rides (walk-on riders) and logging pickup and dropoff events.

3.5 Advanced Sensing

Autonomous vehicles have advanced sensing capabilities to allow for obstacle detection, path planning, and localization when operating without a driver. When the MIT MOD
system operates under human actuation, advanced sensing capabilities are not needed for safe driving. However, the sensing capabilities of autonomous vehicles extend beyond enabling navigation to also provide scene parsing and contextual labeling. For example, autonomous vehicles do not simply plan around obstacles using raw sensor data directly, but rather classify that data into pedestrians, bikes, and cars in order to model the behaviors of those obstacles. Instead of using classified obstacle data for path planning, the MIT MOD system seeks to build a model for how classified traffic moves through a network graph for the campus. Given that the MIT MOD system operates in a campus environment, the primary source of traffic data is provided by pedestrians. Therefore, a means for tracking pedestrians is needed to satisfy the sensing requirement for the MIT MOD system.

This section presents an overview of the novel pedestrian detection and tracking algorithm that was developed for the MOD system. Pedestrian detection and tracking requires identifying pedestrians within proximity of the vehicle and recording their positions. For some static environments, both detection and tracking could be performed using a single camera, but no single sensor solution exists for moving sensors attached to vehicles. Instead, pedestrian detections from Lidar and camera data are fused to determine the trajectories of pedestrians relative to the vehicle. Through localization of the vehicle, the trajectory of the pedestrians in a global map frame can be recorded. The hardware setup, localization method, and pedestrian tracking methods are discussed. For a more detailed discussion on each topic, see Appendix A.

3.5.1 Hardware

The sensing suite was developed in accordance with sensors that are commonly found on autonomous vehicles. Each of the three GEM vehicles is equipped with an identical suite of sensors consisting of camera and Lidar. Figure 3-8 shows a GEM vehicle with the sensing suite. Three Logitech C920 cameras are mounted on each vehicle, providing a front-facing 160° coverage. Two SICK LMS151 2D Lidars are mounted on the hood and roof of the vehicle, providing a front-facing 50 m, 270° coverage at
varied heights. A single Velodyne VL-P16 3D Lidar sensor is mounted on the roof of the vehicle, providing a 100 m, 360° coverage. Localization is performed using only the upper 2D Lidar. Pedestrian tracking is performed using only the cameras and lower 2D Lidar, the coverage of which is shown in Figure 3-9.

3.5.2 Localization

Collecting trajectories of pedestrians in a global coordinate frame requires that the MOD vehicle also be localized within a global frame. To accomplish this, a global occupancy grid map is created for the MOD service area that is used for vehicle localization through onboard sensing. The map generation and localization is performed using laser scans from the roof-mounted 2D Lidar. The Robot Operating System (ROS) [53] is used for managing sensor data and perception algorithms. Odometry for the vehicle is estimated using the Hector SLAM ROS package which performs laser scan registration on multi-resolution grid maps [38]. An occupancy grid map of the environment is obtained using the SLAM Karto ROS package which uses a graph SLAM approach with loop closures [30]. Real-time localization of the vehicle is
Figure 3-9: Pedestrian sensing coverage of the MOD vehicles. Pedestrian sensing is performed using the lower 2D Lidar (orange) with 50 m, 270° coverage as well as 3 cameras that have a combined 160° coverage (purple).
Pedestrian tracking framework. The camera and Lidar are individually processed and then combined to generate pedestrian trajectory data.

achieved using the AMCL ROS package which uses a particle filter to track the pose of the vehicle with respect to the generated map [57].

### 3.5.3 Pedestrian Tracking

Pedestrian tracking is performed in three phases: Lidar clustering, pedestrian detection, and cluster classification. An overview of the system architecture is shown in Figure 3-10.

Lidar clustering dynamically clusters raw laser scan data from the lower Lidar in order to track obstacles in front of the vehicle. Laser scans are first transformed into the map frame using vehicle localization and then processed to filter out background data corresponding to static buildings from the occupancy grid map. Sequential measurements from the filtered laser scans are then clustered together using Dynamic Means [14], which is a fast, general purpose clustering algorithm designed to capture the temporal and spatial evolution of clusters. Obstacle tracking is available inherently using dynamic means because cluster centers corresponding to dynamic obstacles will
be updated to reflect the motion while retaining the same cluster id. The result of Lidar clustering is a set of cluster trajectories corresponding to all obstacles in the sensing range of the vehicle.

Pedestrian detection is used to identify the presence of pedestrians in front of the MOD vehicle using an off-the-shelf vision processing algorithm. Image data from the three cameras on the vehicle are all streamed through a VeryFast pedestrian detector [11], which produces bounding boxes indicating the locations of pedestrians in the images. The SVM-based VeryFast detector is able to achieve high frame rate detections on the order of 90 fps, which makes it well suited for real time applications. The bounding boxes provide classification information within the camera reference frame that needs to be transformed into the map frame common to the Lidar clusters. The left, middle, and right edges of the pedestrian’s bounding box are converted into a set of detection vectors that are projected into the map frame using extrinsic calibration data.

Cluster classification uses the pedestrian detections from the camera to classify which tracked obstacle clusters correspond to pedestrians. A fusion method [32] is used to assign pedestrian probabilities to each cluster based on the relative alignment of clusters and detection vectors. Over time, clusters continually close to detection vectors will receive an increasingly larger pedestrian likelihood, while those farther away will also receive a reduced likelihood. The final output of the framework is the global trajectories of all clusters that have exceeded a pedestrian likelihood threshold. Figure 3-11 shows a visualization of the pedestrian tracking implementation running on an MOD vehicle.

3.6 MIT MOD Evaluation

The capabilities of the MIT MOD system were evaluated through experimental testing. One goal is to demonstrate the passenger operation of the MOD system and determine specific parameters of the MOD environment such as customer arrival and destination locations and frequencies. Another goal is to demonstrate the pedestrian tracking
Figure 3-11: Pedestrian tracking implementation on an MOD vehicle. (Top) The white box represents the vehicle, green arrows represent the detection vectors of pedestrian detections projected into the map frame, cylinders represent detected pedestrians with paths trailing. (Bottom) The images from the three cameras show bounding boxes of the detected pedestrians outlined in green (with faces covered for privacy reasons).

capabilities of the MOD shuttles and record traffic data for the MOD environment such as pedestrian trajectories.

3.6.1 Passenger Request Testing

Passenger request testing of the MOD system was performed over the course of a Fall semester. Testing of the MOD system was limited in terms of time and participation size. The service operated during lunch hours between 11am and 1pm every weekday. Participation was limited to 71 students and faculty who used the service only at their convenience, often for round trip lunch runs or as a connection from the MBTA subway station. The system operated with as few as one vehicle or as many as all three vehicles. In total, there were 36 requested rides and 22 hailed rides occurring over 38 separate days. The average arrival rate was 0.0125 customers/min for requested rides and 0.0096 customers/min for hailed rides. As expected, different locations on campus experience different frequencies for pickup or dropoffs. The relative number of pickups and dropoffs at each node in the network graph is shown in Figure 3-12.
Figure 3-12: Pickup and dropoff locations for the campus MOD system. The blue circles show pickup locations, where larger circle thickness indicates larger number of pickups. The red circles show the same for dropoff locations.

The locations with the largest number of pickups and dropoffs correspond to campus building entrances and the MBTA subway stop. The number of pickups and dropoffs is not identical for each node because demand can be non-symmetric during the given 2 hour observation period (i.e. students may ride to a class that does not end until after the observation period). The average wait time for requesting customers was 2.4 min and the average ride time for any customer was 3.5 min. The average route distance traveled was 404.8 m.

### 3.6.2 Pedestrian Trajectory Collection

Pedestrian trajectory data was collected over the course of a Spring semester. Between 1 and 3 vehicles were driven in circuits through the campus, specifically to collect...
Figure 3-13: Pedestrian trajectories overlayed on the full network graph. Trajectories are indicated by a colored line and circle, where the circle indicates the final pose of the trajectory. Each trajectory was collected using sensors onboard MOD shuttles.

pedestrian data. As the vehicles drive, the Lidar and camera data are fused online and pedestrian trajectories are recorded. According to [32], the vehicles are capable of capturing over 90% of pedestrians paths with only 1.5 false detections per minute when driving in the MOD environment. The vehicles were driven a total of 94 days collecting a total of 64,487 pedestrian trajectories. Figure 3-13 shows the trajectory data on the full network graph. The trajectories for the pedestrian traffic are well aligned with the links of the network graph. Because the data is collected with moving vehicles that can spend more time on one link than another, it is difficult to make claims as to which links are more used. To obtain temporal information such as pedestrian arrival rates, the state of the vehicle will need to be decoupled from the collected data, a challenge that is addressed in Chapter 4.
3.7 High-Fidelity MIT MOD Simulation

The physical MOD system that was developed on MIT campus is used to inform a high-fidelity simulation. The goal of the simulation is to create a test framework that provides more flexibility for parameters that can be difficult to adjust in the physical system. For example, the simulator can more easily vary the number of vehicles without requiring additional physical hardware development or vary customer arrival rates without having to recruit more real-life users to opt-in to the service. The simulator can then evaluate the performance of several MOD fleet management planners in terms of pedestrian traffic sensing and customer QoS metrics. The simulator is termed high-fidelity due to the small, one-second time discretization between the simulated actions of pedestrians and vehicles.

The simulator operates with respect to the network graphs that were developed for the campus. Pedestrians arrive according to a Poisson process with fixed arrival rates for each predetermined route in the system. They move at a constant velocity based on recorded pedestrian trajectory data. A histogram of measured pedestrian velocities from over 16000 pedestrian trajectories is shown in Figure 3-14. Based on this, simulated pedestrian speeds are sampled from a normal distribution with mean speed of 1.5 m/s and standard deviation of 0.4 m/s. At each time step, pedestrians move according to their velocity in the direction of the links in their route, changing direction at each transition node. Pedestrian’s move with some additional random noise such that the movement is not perfectly aligned with the links, as was observed from the collected pedestrian trajectory data. When a pedestrian reaches the destination node, they are removed from the simulation. Customers are sampled from among the pedestrian arrivals such that a fraction of route arrivals will be customers. Customers wait for a fixed time period for a vehicle to arrive and then begin to walk if one does not arrive, where the time period could be set to be infinite. Customers can either perform ride requesting where a request is made known to the vehicle planner immediately upon arrival, or they can perform ride hailing where the request is made to the planner only if a vehicle comes within proximity of the customer.
Simulated vehicles also travel within the network graph. Vehicles move between or wait at nodes according to a centralized planner. Vehicles travel between nodes according to a set of vehicle routes, that are predetermined based on the set of vehicle-traversable links in the network graph. Vehicles move at each time step according to their velocity in the direction of the links in their route, instantaneously changing direction at each transition node. Each vehicle moves with a uniform velocity according to the link that it is traveling along. Links corresponding to pedestrian walkways limit vehicle speeds to 4 m/s, while links corresponding to roadways allow vehicles speeds of 11 m/s. Customers assigned to a vehicle will immediately be picked up or dropped off when a vehicle reaches the customer's arrival or destination node. Potential delays at transition nodes such as stop lights or when picking up or dropping off customers are neglected in this model.

The simulated vehicles are also provided with the same sensing capabilities of their physical counterparts. Each vehicle is given a 160°, 20 m sensing field of view for measuring pedestrian trajectories. Whenever a walking pedestrian enters into the sensing field of view of a vehicle, the pedestrian’s trajectory is recorded. Figure 3-15 shows an example simulation with three shuttles operating for one hour in the campus environment and recording simulated pedestrian trajectories while serving customers.
Figure 3-15: Example simulation of the MIT MOD system. The vehicles are indicated with a colored sensing field of view and an indicator showing the number of passengers on board. Pedestrians moving along links in the network graph are indicated by black squares. Collected pedestrian trajectory data is shown with colored circles, where each trajectory was recorded only when the sensing field of view of a vehicle and position of a pedestrian overlapped.

3.8 Summary

This chapter presents the development of a physical MOD test system and corresponding simulation test environment. The MOD system was developed for operation in the campus environment for MIT. Figure 3-16 shows the complete MIT MOD vehicle fleet. The MOD capabilities of the system are added through the use of an app-based ride request framework in which users request rides via an app and human drivers are automatically routed to serve them. Advanced sensing capabilities were added in the form of mobile pedestrian traffic tracking using onboard camera and Lidar sensors. These capabilities were evaluated through real-world experiments and used to quantify properties of the created system. A high-fidelity simulation was developed
Figure 3-16: MIT MOD fleet. The fleet is made up of three four-passenger GEM e4s vehicles equipped with the sensing capabilities of autonomous vehicles.

to reflect the real-world MIT MOD system in terms of request functionality and pedestrian trajectory sensing. The MIT MOD system, either through experimental testing or simulation, is used to motivate and evaluate the demand estimation and fleet management frameworks that compose the contributions of this thesis.
Chapter 4

Moving Observer Method

One of the capabilities added by autonomous vehicles in MOD systems is the ability to observe traffic data. While Chapter 3 demonstrated how vehicles can collect trajectory data, extracting useful traffic arrival rate information from trajectory data is non-trivial. This chapter presents a moving observer method that estimates the arrival rates of traffic within an MOD network graph using trajectories observed from moving observers, i.e. autonomous shuttles. Because this work focuses on pedestrian traffic, the following analysis is worded for pedestrian sensing, but it should be noted that the same analysis can be applied to non-pedestrian traffic as well, given adequate sensing capabilities.

The first main challenge for traffic arrival rate estimation is the partial observability of pedestrian arrivals. Each pedestrian arriving in the network graph will walk a specific route, however, it is generally not possible to continuously observe pedestrians traveling along an entire route since an observer would have to travel the same route at the same time for every pedestrian. Instead, observations are made only at the link level where an observed pedestrian is assumed to travel the whole length of the link. For this reason, the aim of the moving observer method is to estimate link-level arrival rates for each link in the network graph. Link arrival rates can later be used to estimate node arrival rates, the process for which is presented in Chapter 5. Additionally, link arrival rates could be used to estimate route arrival rates, but this is a separate active research field known as Origin-Destination Matrix Estimation, and
the interested reader is referred to [6].

The second challenge for traffic rate estimation is the undesired correlation between pedestrian observations and moving sensors. As discussed in Section 2.3, observation data for Poisson processes takes the form of the number of arrivals, \( m \), observed over a time period of \( \tau \). For static sensors, link arrival rate estimation is trivially performed by counting the number of passing pedestrians in a given time. For mobile sensors, however, pedestrian counts and observation times are dependent on the vehicle motion. For example, a vehicle traveling along a link will have different pedestrian counts if it travels with or against the pedestrian flow, despite having the same traversal time. Therefore, the moving observer method aims to decouple the vehicle motion from the pedestrian observations in order to accurately measure pedestrian arrival counts and observation time periods.

4.1 Moving Observer Method

As described in Section 3.5, MOD shuttles equipped with the sensing capabilities of autonomous vehicles can be used to measure pedestrian trajectories throughout an MOD network graph. A moving observer method is proposed for estimating arrival rates by using the pedestrian trajectory data along with pedestrian velocities and the vehicle’s known sensing field of view. Pedestrian velocities are estimated directly from the trajectory data by differentiating pedestrian positions with respect to time. The known sensing field of view is estimated through sensor calibration and testing.

Consider a link in the network for which pedestrians arrive at its origin node according to a Poisson process with link arrival rate \( \lambda_l \) and then travel along that link. At time \( t_p \), a pedestrian \( p \) arrives at the link’s origin node and travels along the link with constant velocity \( v_p \). At time \( t \geq t_p \), the position \( x_p \) of the pedestrian relative to the link’s origin node will be \( x_p = v_p(t - t_p) \).

An MOD vehicle in proximity of the link at \( t \) will observe any pedestrians that fall within its sensing field of view, along with their velocity \( v_p \). Figure 4-1 illustrates how the moving vehicle observers pedestrians along the link of a network graph. The
Figure 4-1: Moving observer diagram. Pedestrians (orange and green) leave the origin node of a link according to arrival rate parameter $\lambda$. The moving observer (blue box) will observe any pedestrians that are within the sensing field of view (blue wedge).

Portion of the link within the sensing field of view has length $d_{obs}$ and is bounded by locations $\{x_1, x_2\}$, where $x_1 = x_2 + d_{obs}$. For a pedestrian to be within the sensing region at time $t$, it must hold that $x_p \in [x_2, x_1]$. Using the pedestrian's velocity, the bound on the pedestrian's arrival time at the link's origin node $t_p \in [t_1, t_2]$ can be determined as follows:

$$x_2 \leq x_p \leq x_1$$

$$x_2 \leq v_p (t - t_p) \leq x_1$$

$$t - \frac{x_2}{v_p} \geq t_p \geq t - \frac{x_1}{v_p}$$

$$t_1 \geq t_p \geq t_2,$$

where $t_1 = t - \frac{x_1}{v_p}$ and $t_2 = t - \frac{x_2}{v_p}$.

By extension, the vehicle would observe $m$ pedestrians at time $t$ if each pedestrian left the link's origin node between times $t_1$ and $t_2$, traveling at speed $v_p$. Therefore, knowledge of the sensing region bounds $\{x_1, x_2\}$ and pedestrian velocity information $v_p$ can be used to estimate a window of projected observation times $\{t_1, t_2\}$ within which $m$ pedestrians are estimated to have arrived in the network. At each point in time, an observation is made on the count of $m$ pedestrians arriving in estimated time period $\tau$ defined as $\tau = t_2 - t_1 = \frac{d_{obs}}{v_p}$.

65
In practice, not all pedestrians will travel at the same speed in the network. In the case where \( m \) pedestrians are observed in the sensing region with different velocities \( \{v_1 \ldots v_m\} \), an approximation of \( v_p \) is made using an average speed. In this case, it is more appropriate to use the space mean speed, as done in [45]. Unlike the arithmetic mean speed that averages over time, the space mean speed averages over distance and is defined as

\[
\bar{v}_p = \frac{m}{\sum_{i=1}^{m} v_i^{-1}}.
\] (4.1)

In the \( m = 0 \) case, an expected pedestrian speed \( \bar{v}_p \) is used based on an assumed average pedestrian velocity.

### 4.2 Poisson Arrival Rate Estimation

Using the moving observer method, a shuttle driving in the MOD network graph will continually collect data in the form of counts and time periods for each link. The Poisson rate parameter \( \lambda_i \) can be estimated using the set of \( N_o \) independent observations consisting of counts \( [m_1 \ldots m_{N_o}] \) and estimated arrival time periods \( [\tau_1 \ldots \tau_{N_o}] \). Arrival rates can either be estimated using the maximum likelihood estimator (MLE) or a Bayesian estimator as discussed in Section 2.3. For this analysis, the MLE with upper and lower confidence bounds are used. Sufficient statistics \( N_c = \sum_{i=1}^{N_o} m_i \) and \( T_c = \sum_{i=1}^{N_o} \tau_i \) are computed for the set of observations for each link, the MLE is computed according to Equation (2.5), and the confidence bounds are computed according to Equations (2.6) and (2.7).

A criteria imposed on the data for estimating the MLE is that each observation must be independent. In the case of the moving observer method, the key to maintaining independent observations is to ensure that the recorded projected observation times do not overlap. As the observer moves, the projected observation times may overlap based on the relative speeds of the vehicle, pedestrians, and sampling rate of the data. For example, consider the limiting case where the vehicle is moving with the traffic flow and at the same speed as pedestrians. In that case, the projected observation
times of the data will not change because each observation will be of the same counted pedestrians leaving their origin node during the same projected time window. If the vehicle is making observations at a fixed sampling rate, it will generate multiple dependent observations all with overlapping intervals. To ensure that the data is independent and eliminate sampling rate factors from affecting the estimation results, all new measurements with projected observation times overlapping with previous times are discarded from the data set.

4.3 Comparison To Previous Moving Observer Methods

The novelty and benefit for this new proposed moving observer method comes from the use of onboard sensors with a known sensing region, allowing for arrival rate estimation on any portion of links within the vehicle’s sensing region, whether the vehicle is traveling with, against, or adjacent to the traffic. Previously, a one-way moving observer method was proposed in [45] that uses manually observed traffic counts from human drivers. This method was an improvement on the original method in [59], but it still has a limitation that the observer vehicle is required to be driven along the whole length of a link, opposite to the direction of traffic. While the newly presented moving observer does not require driving the whole length of a link, if it were to do so, the resulting arrival rate estimation would be equivalent to the previous one-way moving observer method, as demonstrated through the following analysis.

Consider a time $t$, when the vehicle is observing pedestrians between link positions $x_1$ and $x_2$, each traveling towards the vehicle with velocity $v_p$. After time $\Delta t$, the vehicle moves forward with velocity $v_v$ and the new observation region will be $x'_1 = x_1 - v_v \Delta t$ and $x'_2 = x_2 - v_v \Delta t$. The projected observation times for each location $x_i$ is $t_i = t - \frac{x_i}{v_p}$ and for each $x'_i$ is $t'_i = t + \Delta t - \frac{x'_i}{v_p}$. The projected arrival time periods are both $\tau = \frac{d_{obs}}{v_p}$, where $d_{obs} = x_1 - x_2$. As mentioned, in order ensure that observations are independent, only samples with non-overlapping projected times will be included; that
is, only samples with the property $t'_2 < t_1$ or $t'_1 > t_2$. Using either of these criteria, the bound on the minimum time between making observations, $\Delta t_{\text{min}}$, is found to be $\Delta t_{\text{min}} > \frac{d_{\text{obs}}}{v_v + v_p}$ as follows:

\[
\begin{align*}
t'_1 > t_2 \\
t + \Delta t - \frac{x'_1}{v_p} > t - \frac{x_2}{v_p} \\
v_p \Delta t > x'_1 - x_2 \\
v_p \Delta t > x_1 - v_v \Delta t - x_2 \\
\Delta t > \frac{x_1 - x_2}{v_v + v_p} \\
\Delta t > \frac{d_{\text{obs}}}{v_v + v_p}.
\end{align*}
\]

The total time the vehicle spends on link $l$ is denoted as $T_l = \frac{d_l}{v_v}$ where $d_l$ is the length of the link. The total number of independent time period observations that will be made, $N_o$, is

\[
N_o = \frac{T_l}{\Delta t} = \frac{d_l}{v_v} \cdot \frac{(v_v + v_p)}{d_{\text{obs}}}.\]

After traversing the link, the sufficient statistic for the total amount of observed time $T_c$ can be computed as

\[
T_c = N_o \tau = \frac{d_l}{v_v} \frac{(v_v + v_p)}{d_{\text{obs}}} \cdot \frac{d_{\text{obs}}}{v_p} = d_l \left( \frac{1}{v_v} + \frac{1}{v_p} \right).
\]

During each independent observation time window, a count of pedestrians will be made. Let $N_c$ be the sufficient statistic for the sum over all of these pedestrian counts. The MLE arrival rate estimate is

\[
\lambda = \frac{N_c}{T_c} = \frac{N_c}{d_l \left( \frac{1}{v_v} + \frac{1}{v_p} \right)}.
\]

Equation (4.2) is equivalent to the result found in [45], indicating that the methods would perform equivalently when performing a one-way complete link traversal. The significance is that there is no loss in performance with the proposed moving observer.
method over previous methods. The added benefits are that observations of $m$ and $\tau$ can be made at any time for any link within the vehicle's sensing region. This results in generating a much larger set of observations as vehicles drive throughout an MOD network.

4.4 Experimental Testing

The accuracy and confidence of the arrival rate estimates generated using the moving observer method are evaluated in hardware and simulation experiments.

4.4.1 Hardware Experiments

A hardware experiment was performed using one of the MOD vehicles presented in Chapter 3 that is equipped with camera and Lidar that provide a sensing region of 20 m range and 160° field of view. While the moving vehicle drives, pedestrian tracking is used to obtain pedestrian trajectories and the moving observer method is used to estimate the pedestrian arrival rates in the network. The testing was performed using the pedestrian network graph shown in Figure 3-4. The moving observer vehicle was driven such that it attempted to cover all links equally. The vehicle was driven at speeds between 2.5 and 4.5 m/s and the pose of the vehicle was always known through localization. To establish ground truth, an equally equipped MOD vehicle served as a stationary link counter and was parked along the link between nodes 20 and 22. Figure 4-2 shows the experimental setup.

The two vehicles collected data for a time period of 1.5 hours. Figure 4-3a shows the estimated arrival rates from both the moving observer and the stationary counter for pedestrians traveling from node 20 to 22. Similarly, Figure 4-3b shows the estimated arrival rates in the opposite direction, from node 22 to 20. The time profile of the arrival rates is generated using a moving average filter, where arrival rates and confidence intervals are computed from data within a 10 minute interval using Equations (2.5) to (2.7). The time profiles indicates that the arrival rates generally vary over time, with a large increase around 12 PM, which is known to be common for these two
Figure 4-2: Hardware setup for moving observer experiment. The white shuttle is stationary and serves as a ground truth link counter. The blue vehicle is a moving observer that occasionally passes the same link as the stationary sensor. Both vehicles are equipped with the same hardware and perform pedestrian detection and tracking.

links due to the change of classes that occurs then. The moving observer only makes sparse measurements of the link, but the mean values of the estimated arrival rate at each measurement are generally consistent with those from the stationary counter. The confidence intervals for the moving observer method estimates are much larger, which is expected due to the smaller amount of effective viewing time in the 10 minute window. During that 10 minute window, the moving observer traversed other areas of the network graph. Unlike the stationary counter, the moving observer was able to estimate similar arrival rate profiles for every other link in the network. As an example, similar arrival rate profiles on four other links in the network are shown in Figure 4-4.

In an MOD setting, less confident estimates of the relative arrival rates among all links is arguably more useful than very confident estimates on just a few links because it provides insights into which locations in the network graph have the most arrivals at any given time. More confident estimates are achievable with the moving observers, however, if multiple MOD vehicles are traversing the network graph at the same time and their arrival rate observations are combined. Similarly, through management of an MOD fleet, more confident estimates can be obtained by having vehicles spend more time on certain links than others. In the limiting case where a vehicle is parked,
(a) Pedestrians traveling from node 20 to node 22.

(b) Pedestrians traveling from node 22 to node 20.

Figure 4-3: Moving observer arrival rate estimate comparison. The stationary counter (red) collects data continuously while the moving observer (blue) collects data only at certain intervals (square markers). Estimates are shown with a moving average filter applied using a 10 minute interval centered at each point in time. The shaded regions indicates the 90% confidence interval.
Figure 4-4: Arrival rate estimates for pedestrians traveling along additional links in the network. Estimates are shown with a moving average filter applied using a 10 minute interval centered at each point in time. The shaded regions indicate the 90% confidence interval.
the estimates would be equivalent to a stationary link counter.

4.4.2 Simulation Experiments

Simulation Setup

A similar experiment is conducted in simulation, where ground truth is available along every link of the pedestrian network graph. The simulator presented in Section 3.7 is used for the experiment, where vehicles and pedestrians move along links in the pedestrian network graph. In this test, vehicles do not serve customers but rather drive for even coverage of the network because the focus is on measuring pedestrian arrival rates using a moving observer. In addition to the baseline simulation parameters, the simulated pedestrian arrival rate parameters are also chosen to match real-world data. The simulated arrival rates in the network are determined by applying the moving observer method to data collected from the physical MOD vehicle. Figure 4-5 shows 20 day averaged arrival rates for each link. Based on the data a nominal arrival rate of 1.62 ped/min was chosen for use in simulation. The simulator was run on the pedestrian network graph with half of the pedestrian links each having the same constant arrival rate and half having no arrivals. In order to ensure that the vehicle can make a significant number of observations of each link, 1 hour of driving was simulated.

Performance across full MOD area

The arrival rate estimates and corresponding confidence intervals from the moving observer and stationary counters are shown in Figure 4-6 alongside the ground truth. The results show that a single moving observer is able to generate reasonable arrival rate estimates for most links in the network. The estimates and confidence interval from the stationary counter indicate the best confidence bound that could be achieved by the moving observer method, which would occur if it was parked on the link. To achieve this, however, there would need to be a stationary counter on every pair of directed links in the network, in this case 37 stationary counters were used. In
Figure 4-5: Arrival rates on each link in the network were estimated from an MOD vehicle using the moving observer method. Data was collected during the same 2 hour period between 11 AM and 1 PM and averaged over the course of 20 days.

In contrast, the moving observer method is able to generate an estimate for all of the links using only a single vehicle. This, however, comes at the cost of less certainty in some estimates as indicated by the larger confidence intervals. Higher uncertainty is inherent to the method because the vehicle spends less time on each link compared to the stationary counters and the total observation time will be less.

Despite each active link having the same arrival rate, the results show that both the estimated parameters and their confidence intervals resulting from the moving observer method tend to vary across links. Those links with highest uncertainty correspond to the shortest links in the networks. While the vehicle attempts to make an equal number of trips to each link, the effective observation time will be dependent on the length of the link and the vehicle's velocity. This indicates that methods for actively sensing along links should consider spending more time on shorter links by either visiting them more often or by reducing the vehicle's velocity while traversing them. An active sensing approach is presented in Chapter 5.
Figure 4-6: Full network simulation results. The results from a full network simulation compare the moving observer method (blue) with a stationary counter (red) and ground truth (black). The results are averaged over 100 simulated runs and 90% confidence intervals are shown.

Parameter Sensitivity

To understand the effect of observation time on moving observer estimates, a simulation was run in which the number of times the vehicle traverses a link is varied. Here the focus is on a single network link of length 100 m for which the vehicle only observes the link for one-third of the total simulation run time. Figure 4-7 shows how the estimates vary as a function of number of visits across the link. For the 100 m link, the vehicle is able to correctly estimate after relatively few visits, but with low confidence in the estimates. With more visits to the link, the confidence interval decreases as is expected.

To understand the effect of the magnitude of arrival rates on moving observer estimates, the same simulation is run over a range of true arrival rates with a nominal
Figure 4-7: Arrival rate estimates for varied number of link traversals. The results are averaged over 100 simulated runs and the 90% confidence intervals are shown.

Figure 4-8: Arrival rate estimates for varied true arrival rates. The results are averaged over 100 simulated runs and the 90% confidence intervals are shown.
number of visits of 10. Figure 4-8 shows the estimated arrival rates from the moving observer method given different constant true arrival rates. The results show that the arrival rate estimates from the moving observer method are equally valid over the range of arrival rates that are expected in the network.

4.5 Summary

This chapter introduced a new moving observer method for measuring traffic arrival rates within an MOD network graph. The method focuses on decoupling the motion of mobile sensors from the trajectory data they collect. The output of the algorithm is a set of accurate, independent observations in the form of pedestrian counts and observation time periods. The accuracy of these observations is tested by comparing maximum likelihood link arrival rate estimates made using the moving observer with those from stationary counters. Experimental testing demonstrates that the moving observer method achieves comparable arrival rate accuracy to that of pervasive stationary counters, allowing for network-wide sensing with mobile sensors.

While traffic estimation has many uses in its own right, the focus of this thesis is on using traffic arrival rates to improve estimates of customer arrival rates. Chapter 5 builds on the work of this chapter by utilizing the pedestrian counts and observation time periods from the moving observer method to model customer arrival rates.
Chapter 5

Customer Demand Estimation

This chapter introduces modeling and active sensing approaches for measuring and predicting customer demand within an MOD system. A customer demand model is critical for improved performance of MOD systems. In order for MOD fleet management planners to move vehicles in anticipation of customer arrivals, a demand model must be consulted. The fleet management planners presented in Chapters 6 and 7 utilize the customer demand model presented in this chapter for positioning vehicles within the MOD network graph. A key element to the proposed customer demand model is the correlation of traffic arrival rates with MOD customer arrivals, allowing inferences to be made regarding expected customer arrivals based on current traffic patterns. As presented in Chapter 4, MOD shuttles can directly estimate traffic arrival rates using onboard sensors. This chapter extends the utility of traffic arrival rate measurement for MOD systems by incorporating pedestrian link arrivals into a novel customer arrival model. Additionally, an exploration planning algorithm is presented for performing active sensing with MIT MOD shuttles, so that customer arrival rate estimation is improved as shuttles explore the MOD system. The model, active sensing approach, and experimental testing results are presented.
5.1 Customer Arrival Model

Customer demand is quantified as the number of MOD customers that will arrive at specific locations within an MOD system over a prediction time horizon. This section presents a customer arrival model based on a network graph for an MOD system composed of pedestrian traffic. The network graph contains a set of \( N_n \) nodes, a set of \( N_l \) directed link edges, and a set of \( N_r \) routes. Customer arrivals are modeled with a Poisson process at each node in a network graph, that are each parametrized by node arrival rates. Customers are modeled as arriving in the system with a predetermined route \( r \) according to a discrete-time Poisson process with the time-varying arrival rate parameter \( \lambda_r^c(t) \). The time discretization is assumed sufficiently large such that arrival rates are assumed constant within the considered operating regime, that is \( \lambda_r^c(t) = \lambda_r^c \). The arrival rate of customers at each node, \( \lambda_n^c \), is given by \( \lambda_n^c = \sum_{r \in \{r(n, \cdot)\}} \lambda_r^c \), where \( \{r(n, \cdot)\} \) represents the set of routes with \( n \) as the origin node. The number of customer node arrivals, \( c_n \), over a prediction time horizon, \( t_{pred} \), is modeled as \( c_n \sim \text{Pois}(c_n; \lambda_n^c t) \). Predicting the expected customer arrivals in the system is achieved through estimation of the customer arrival rate parameters, \( \lambda_n^c \), for each node in the network graph.

5.2 Two-factor Demand Model

The key challenge for estimating customer arrivals is the sparsity of observable data. Direct estimation of the customer arrival rates would require observing the number of customer arrivals at each node over a given period of time. For ride hailing, this would require vehicles to be placed at each node in order to fully observe the network. For ride requesting, this requires building up historical databases of arrivals at each node. The fundamental problem is that customer arrivals are relatively low frequency events that require significant observation time to build a reliable estimate. In MOD systems, observation time can be even more limited as arrival rates could be temporally varying. For example, by the time enough customer arrivals have been observed, a lunch rush
Figure 5-1: Two-factor arrival diagram. Pedestrians (green and orange) arrive according to Poisson arrival rates for each node. A fraction of pedestrian arrivals are customers (purple). The customer arrival rate is the product of the pedestrian arrival rate and the customer fraction.

could be over and the arrival rate would be different. Instead, low frequency customer arrivals are correlated with higher frequency traffic arrivals to more quickly learn demand patterns.

Customer arrivals are estimated through a two-factor model that splits customer arrivals at each node into pedestrian arrivals and customer fractions. The customer and pedestrian arrivals are modeled as Poisson processes with respective arrival rates, $\lambda^c$ and $\lambda$. The customer probabilities at each node are represented using a Bernoulli distribution with customer fraction parameter $p_n$. Due to the decomposition property of Poisson processes, the relation between parameters is $\hat{\lambda}^c_n = p_n \lambda_n$. Figure 5-1 illustrates the two factor arrival model, showing how the customer arrivals are modeled as a fraction of total pedestrian arrivals. The benefit to this decomposition is that either large customer fractions or large pedestrian arrival rates can be used as indicators of nodes with large customer arrival rates. While customer fractions still require making observations at each node, the pedestrian arrival rates can be estimated using the moving observer method and serve as indicators as to which nodes are worth observing. With two parameters, uncertainty in the estimates of the customer arrival rates will be dependent on any uncertainties in either pedestrian arrival rates or customer fractions. To address this, a Bayesian framework is used to capture the uncertainty in customer arrival probabilities, and Bayesian estimators for both pedestrian arrival rates and
customer fraction parameters are developed.

For pedestrian arrival rate estimation, observations of pedestrian counts and observation times are available using the moving observer method presented in Chapter 4. The method provides data in the form of a pedestrian link count \( m_l \) with corresponding observation time window \( \tau_l \) for a particular link. For Bayesian estimation, link arrival rates, \( \lambda_l \), are modeled and updated using Gamma distributions with hyperparameters, \( \hat{\alpha}_l \) and \( \hat{\beta}_l \), that is \( \lambda_l \sim \text{Gamma}(\lambda_l; \hat{\alpha}_l, \hat{\beta}_l) \) While the moving observer method provides pedestrian link arrival rate estimates, the customer arrival model requires pedestrian node arrival rate estimates. As presented in Section 2.3, node arrival rates, \( \lambda_n \), are related to link arrival rates through

\[
\lambda_n = \sum_{l: n = n_{o(l)}} \lambda_l. \quad (5.1)
\]

Therefore, the node arrival rate parameters are distributed according to a sum of Gamma distributed link arrival rates. The node arrival rate parameters are chosen to be modeled using Gamma distributions with hyperparameters, \( \alpha_n \) and \( \beta_n \), that is \( \lambda_n \sim \text{Gamma}(\lambda_n; \hat{\alpha}_n, \hat{\beta}_n) \). In order for Equation (5.1) to hold, node hyperparameters are computed from the link hyperparameters using the Welch-Satterthwaite approximation for the sum of Gamma distributions provided in [41]. That is,

\[
\hat{\alpha}_n = \frac{\sum_{l: n = n_{o(l)}} \left( \frac{\hat{\alpha}_l}{\hat{\beta}_l} \right)^2}{\sum_{l: n = n_{o(l)}} \left( \frac{\hat{\alpha}_l}{\hat{\beta}_l} \right)^2} \quad (5.2)
\]

\[
\hat{\beta}_n = \frac{\sum_{l: n = n_{o(l)}} \frac{\hat{\alpha}_l}{\hat{\beta}_l}}{\sum_{l: n = n_{o(l)}} \left( \frac{\hat{\alpha}_l}{\hat{\beta}_l} \right)^2}. \quad (5.3)
\]

To summarize, as vehicles observe pedestrian trajectories, the moving observer method extracts link observations, that update link hyperparameters, that update node hyperparameters, that provide the belief in the pedestrian node arrival rate estimates.
For customer fraction estimation, observations of customer and non-customer arrivals are available through both pedestrian trajectory data and the MOD ride hailing and ride requesting frameworks. Observations include the number of customers that were picked up at a node, \( c_n \), and the number of non-customer pedestrians observed at the node using vehicle sensors, \( -c_n \). The observations occur any time a customer requests a ride from a node (either through an app or by hailing) or any time a pedestrian’s observed trajectory originates from a node. For Bayesian estimation, the customer fraction parameters are modeled and updated using Beta distributions with hyperparameters \( a_n \) and \( b_n \), that is, \( p_n \sim \text{Beta}(p_n; a_n, b_n) \). Whenever a customer requests a ride or a pedestrian is observed at a node, the hyperparameters that quantify the belief in the node customer fractions are updated.

Given the pedestrian arrival rates and customer fractions, the likelihood for customer arrivals over a time period \( t_{\text{pred}} \) is 
\[
P(\hat{c}_n; \lambda_n, p_n, t_{\text{pred}}) = \text{Pois}(\hat{c}_n; p_n \cdot \lambda_n \cdot t_{\text{pred}}).
\]
Because parameters \( p_n \) and \( \lambda_n \) are themselves modeled as probabilistic distributions that depend on hyperparameter values, computing the probability of customer arrivals with respect to the hyperparameters requires marginalization over the parameters. The likelihood for customer arrivals over a time period \( t_{\text{pred}} \) with respect to pedestrian arrival rate hyperparameters \( \alpha_n \) and \( \beta_n \) and customer fraction hyperparameters \( a_n \) and \( b_n \) is determined through marginalization of the pedestrian arrival rate and customer fraction parameters. The marginalization can be simplified into an analytical expression, the derivation of which is provided in Appendix B. The analytical expression is given as

\[
P(\hat{c}_n; t_{\text{pred}}, \alpha_n, \beta_n, a_n, b_n, ) = \frac{\Gamma(\alpha_n + \hat{c}_n) \cdot \Gamma(\alpha_n + \hat{c}_n) \cdot \Gamma(\alpha_n + b_n) \cdot (t_{\text{pred}}/\beta_n)^{\hat{c}_n} \cdot \frac{\Gamma(\alpha_n + \hat{c}_n) \cdot \Gamma(\alpha_n + b_n + \hat{c}_n) \cdot (t_{\text{pred}}/\beta_n)^{\hat{c}_n}}{\hat{c}_n! \Gamma(\alpha_n)} \cdot \frac{\Gamma(\alpha_n + \hat{c}_n) \cdot \Gamma(\alpha_n + b_n + \hat{c}_n) \cdot (t_{\text{pred}}/\beta_n)^{\hat{c}_n}}{\hat{c}_n! \Gamma(\alpha_n)} \cdot 2F_1 \left( \alpha_n + \hat{c}_n, a_n + b_n + \hat{c}_n; \frac{t_{\text{pred}}}{\beta_n} \right),
\]

where \( \Gamma(\cdot) \) represents the Gamma function and \( 2F_1(\cdot, \cdot, \cdot) \) represents the generalized hypergeometric function. The benefit to the customer arrival likelihood model in
Equation (5.4) is that it incorporates both the natural uncertainty in the Poisson arrival process as well as the parameter uncertainty from online estimation of pedestrian arrival rates and customer fractions. Because the full distribution is available, the model can provide prediction for customer demand in the form of estimated number of customer arrivals along with a measure of uncertainty.

5.3 Active Sensing

An MOD system composed of autonomous vehicles has the ability to perform active sensing through combined onboard sensing and automated routing. By routing vehicles to explore the MOD network graph, parameter estimation for pedestrian arrival rates is performed. In Section 4.4 it was shown that uniform coverage of the network graph does not necessarily provide accurate estimates for all links. Particularly, shorter links will have few observations from a single pass of a moving observer. Rather than strive for uniform coverage, an exploration planner is developed for routing vehicles to sense along links in the network graph based on relative parameter uncertainty for each link. With each pass of a network link from a moving observer, the uncertainty is reduced and the customer arrival likelihood model is improved. However, the customer arrival model also requires estimation of customer fractions as well, which occurs when vehicles observe customer and pedestrian arrivals at nodes. Therefore, the complete active sensing approach combines assigning vehicles to explore to estimate pedestrian arrival rates as well as assigning vehicles to wait at nodes to estimate customer fractions. Specific fleet management strategies for assigning vehicles to wait at nodes are presented in Chapters 6 and 7, this section focuses on the exploration algorithm and overall active sensing performance.

5.3.1 Arrival Rate Variance

This exploration planner routes vehicles along links based on the variance in pedestrian arrival rates. The belief of the link arrival rate, $\lambda_l$, is expressed through the Gamma distribution hyperparameters $\alpha_l$ and $\beta_l$. The mean and variance for a Gamma
distribution are known to be $\mathbb{E} [\lambda_i] = \frac{\hat{\alpha}_l}{\hat{\beta}_l}$ and $\text{Var} [\lambda_i] = \frac{\hat{\alpha}_l}{\hat{\beta}_l^2}$, respectively. Following Equation (4.2), the number of pedestrians that are expected to be observed during a traversal of a link is $\hat{m}_l = \mathbb{E} [\lambda_i] \hat{\tau}_l$, where $\hat{\tau}_l$ is the effective observation time given by $\hat{\tau}_l = \frac{d_l}{v_o} + \frac{d_l}{v_p}$, $d_l$ is the length of the link, and $v_o$ and $v_p$ are the expected vehicle and pedestrian speeds along the link, respectively. Using the Bayesian update for link arrival rates, the expected variance in the link after a traversal is given by

$$\text{Var} (\hat{\lambda}_i) = \frac{\hat{\alpha}_l + \hat{m}_l}{(\hat{\beta}_l + \hat{\tau}_l)^2} = \frac{\hat{\alpha}_l}{\hat{\beta}_l (\hat{\beta}_l + \hat{\tau}_l)}. \quad (5.5)$$

### 5.3.2 Exploration Planner Formulation

The objective of the exploration planner is to assign vehicles to links such that the reduction in link arrival rate variance is maximized. Vehicles are assigned a route composed of links, where $I$ is the total number of assigned links and $i$ represents the link index within the route. The problem can be formed as a non-linear integer problem with binary decision variables $x^v_i \in \{0, 1\}$ equal to 1 if vehicle $v$ is assigned to link $l$ for the index $i$ within the route; and zero otherwise. The exploration problem formulation is,

$$\text{argmax}_{x^v_i} \sum_{i=1}^{N_l} \frac{\hat{\alpha}_l}{\hat{\beta}_l^2} - \frac{\hat{\alpha}_l}{\hat{\beta}_l (\hat{\beta}_l + \hat{\tau}_l) \left( \hat{\beta}_l + \hat{\tau}_l \sum_{v=1}^{V} \sum_{i=1}^{I} x^v_i \right)} \quad (5.6)$$

s.t. \[ \sum_{i=1}^{N_l} x^v_i = 1 \quad \forall \, v, i \quad (5.7) \]

\[ \sum_{i=1}^{N_l} x^v_{i-1} = x^v_i \quad \forall \, v, l, i, \quad (5.8) \]

where $o(l)$ and $d(l)$ are the known origin and destination node indexes of link $l$, respectively; and $x^v_i$ for $i = 0$ is known using each vehicle’s current link. The term $\sum_{v=1}^{V} \sum_{i=1}^{I} x^v_i$ in Equation (5.6) is the total number of visits for link $l$ across all time periods and vehicles. The objective is a measure of the expected reduction in uncertainty across all links. Equation (5.7) ensures that each vehicle is assigned one and only
one link for each index in its route. Equation (5.8) ensures that route continuity is maintained by ensuring that the origin node of the link at \( i \) is the same as the destination node of the previous link at \( i - 1 \). Equation (5.8) can formulated as a linear constraint using a precomputed connectivity matrix \( A \) composed of elements \( a_{jk} \) taking value 1 if \( d(j) = o(k) \) and zero otherwise.

### 5.3.3 Online Implementation

The exploration problem can be solved to assign vehicle routes; however, the objective function in Equation (5.6) is nonlinear and challenging to solve exactly. To address this, an online implementation uses a greedy assignment approximation where route assignments are determined sequentially for each individual vehicle. The feasibility constraints in Equation (5.8) are ensured by assigning routes chosen from a set of pre-computed minimum time routes between any two nodes. The set does not contain inefficient routes between nodes such as those with cycles, although such routes could be added if desired. Each route is evaluated under the nonlinear objective in Equation (5.6). The process is repeated for each unassigned vehicle with knowledge of the previous vehicle’s route so that future link visits are accounted. A greedy approach is used so that vehicles are continually assigned new routes whenever their previous route is completed. Otherwise, routes would have to be recomputed for all vehicles whenever one vehicle completes its route, or vehicles would have to wait until all routes are completed.

### 5.4 Experimental Testing

The simulator presented in Section 3.7 is used to test the customer demand modeling and active sensing approaches. Testing in the simulation environment is ideal because ground truth for customer demand can be easily established. The two-parameter node arrival customer demand model is tested by sampling pedestrian and customer arrivals through a route arrival process to see if customer demand can be successfully captured. Pedestrian and customer arrivals are sampled from a Poisson process with
route arrival rates, while the customer demand model estimates node arrival rates. The distinction is important because pedestrians arriving with routes will generate node arrivals that do not correspond to their true path origin. In the midst of many pedestrian arrival nodes, the goal for the two-parameter customer demand model is to determine which nodes will actually experience customer arrivals. The active sensing framework is tested to demonstrate the effectiveness of estimating customer arrival rates. As vehicles continue to explore the network, the goal is that estimated customer arrival rates converge to ground truth.

5.4.1 Simulation Setup

Pedestrians and vehicles move within the full network graph for the MIT campus. Pedestrian route arrival rates are set to reflect the true arrival rates shown in Figure 4-5. In each round of simulation, ten randomly chosen routes with separate origin nodes are assigned pedestrian arrival rates of 1 ped/min. Three of the ten routes will be customer arrival routes, where customers wait at their origin node for a vehicle. Here customers perform ride hailing, where they will successfully hail a ride if, within 30 seconds of their arrival, a vehicle comes within a 20 meter proximity of the customer, otherwise the customer will instead walk their route. The simulation considers 5 vehicles in the MIT MOD system. A period of 1 hour is simulated.

5.4.2 Active Sensing Approaches

The active sensing planner operates on a 5 minute planning horizon. At the start of the planning horizon, the number of vehicles to assign to each node is first determined from a ride hailing planner presented in Chapter 6. If fewer vehicles are needed than there are vehicles in the system, the remaining vehicles are assigned to drive routes according to the exploration planner, where assigned routes have a length of 5 links. In either case, if a vehicle encounters a hailing customer, the vehicle immediately serves the customer.

Four different fleet management planners are analyzed. The first is the online
implementation of the active sensing planner presented in Section 5.3.3. The second is a pure sensing planner that represents a baseline approach where vehicles continually patrol the network in order to happen upon a hailing customer. The planner is implemented by never assigning vehicles to wait at nodes so that they are always assigned to traverse links through the exploration planner. The third is an oracle planner designed to represent the upper bound on performance for serving customers. The planner is implemented by providing the true number of customer arrivals over the planning horizon to the expected value planner, causing vehicles to often be assigned to wait at nodes. The oracle is provided only with the actual number of arrivals, and not the true arrival rate parameters, so that estimation performance can still be tested. Finally, a fourth approach is included that places stationary counters on every node to fully observe pedestrian and customer arrivals. This strategy would require as many vehicles as nodes and is only included as a lower-bound error metric. Each planner utilizes the customer demand model and performs parameter estimation through Bayesian updates.

5.4.3 Exploration Example

The exploration planner is illustrated through an example simulation shown in Figure 5-2. Here, pedestrians and vehicles have been moving through the full network graph and three vehicles have been assigned to explore. The variance in pedestrian arrival rate estimates are computed and shown for each link in the network graph. For added demonstration in this example, vehicles select from routes up to 10 links in length, and the chosen route for each vehicle is shown. Given the starting position of each vehicle and the relative arrival rate variances in each link, the routes that are chosen from the set of routes will best reduce the overall arrival rate variance across all links in the network.
Figure 5-2: Route assignments from the exploration planner. Relative link uncertainties are shown in magenta. Three vehicles (red, blue, green) are each assigned a route to reduce uncertainty. The route origin for each vehicle is marked with a cross.

5.4.4 Active Sensing Results

In campus MOD systems, arrival rates can vary throughout the day, so it is important to determine the timescales for which arrival rates can be estimated using the active sensing planner. The simulation considers all planners initially having a uniform uninformative prior belief for each node, and evaluates the mean squared error between the true and estimated customer arrival rates over the course of 1 hour. Figure 5-3 shows the performance of each strategy for the one hour operation. Initially, the prior beliefs result in high starting error. The active sensing and pure sensing planners initially assign all vehicles to explore the network. During the first 10 minutes, the arrival rate error is reduced from exploration. After that time, customer fraction estimates are improved by the active sensing planner that has begun to assign vehicles to nodes. The pure sensing planner rarely improves customer fraction estimates as vehicles are never assigned to wait at nodes, thus the error converges to a higher
value. The oracle planner immediately assigns vehicles to known arrivals and is slower to reduce error as vehicles only traverse the network graph when serving customers. The results demonstrate the effectiveness of the active sensing planner for improving customer arrival rate estimates over time. For estimation of arrival rates that would change on short time scales (10 minutes), an exploration planner approach performs better than waiting. For changes over larger time scales (1 hour), the estimation accuracy approaches that of the lower-bound stationary counter. The results also indicate the effectiveness of the two-parameter model, where customer node arrival rate error converges to zero despite the fact that customer arrivals are occurring along routes. This shows that the two-parameter model is capable of capturing the dynamics of customer arrivals within the MOD system.

5.5 Summary

This chapter introduces a customer demand model for estimating future customer arrivals within an MOD system. The key to the model is a two-parameter estimation
approach that allows for incorporation of traffic information to improve estimates of customer arrivals. A Bayesian framework is presented that utilizes the data available from the moving observer method presented in Chapter 4 and the MOD framework presented in Chapter 3 in order to make online, recursive updates to the demand model. An exploration planner is presented that purposefully routes vehicles in an MOD network graph in order to reduce uncertainty in pedestrian arrival rates. This is used within an active sensing planner that combines exploration and node assignment in order to improve customer arrival rate estimates. Simulation results show that the active sensing approach successfully reduces arrival rate estimation error across the network graph over time.

The customer demand model presented in this chapter can be used to predict future customer arrivals. Knowledge of future customer arrival locations allows MOD fleet management planners to improve customer quality of service by positioning the vehicle fleet with respect to anticipated demand. Chapters 6 and 7 present fleet management strategies for ride hailing and ride requesting MOD environments that are based on the customer demand model presented in this chapter.
Chapter 6

Fleet Management for Ride Hailing

This chapter addresses the challenge of managing an autonomous vehicle fleet in order to improve the number of served customers in an MOD system that features ride hailing. This work is motivated by the MIT MOD system that operates in a campus environment and allows customers to both make app requests and hail rides directly from a fleet of electric shuttles. Experience has shown that the hassle of requesting a ride via an app often causes many customers to default to walking. Ride hailing can lower the barrier to entry by allowing customers who plan to walk, to instead hail a ride if they encounter an MOD vehicle. The challenge for ride hailing is that unless a vehicle is in proximity of a customer’s arrival location, that customer will not be served. Furthermore, there are often more potential customer arrival locations than MOD vehicles available to monitor them. The ride hailing fleet management approach makes use of the customer demand model presented in Chapter 5 that estimates customer arrivals based on past arrivals and pedestrian traffic. A key aspect of that model is the focus on quantifying the uncertainty in arrival rate estimates using a Bayesian framework. In this chapter, customer arrival rate probabilities are incorporated into robust ride hailing planners that plan under uncertainty. Expected value and chance-constrained MOD fleet planners are developed and evaluated through simulation of the campus MOD framework.
6.1 Hailing Model

This section presents the model that captures how ride hailing customers arrive and travel within the MOD system. Customer arrivals in the network graph are modeled as node arrivals that occur according to Poisson processes. The two-parameter model presented in Section 5.2 is used to split customer arrival rate parameters into pedestrian arrival rates and customer fractions. Upon arrival, customers look to hail a ride from one of the $N_v$ MOD vehicles in the system. If an unoccupied vehicle comes within proximity of a waiting customer, the customer becomes “served” and will receive a ride. If a customer does not receive a ride after waiting for a time of $t_{wait}$, they will walk to their destination. While it is possible that a customer may be interested in receiving rides even after beginning to walk, it is likely that their interest drops as the customer gets closer to their destination. Here, the limiting case is assumed, where customers immediately lose interest in receiving a ride after waiting a time of $t_{wait}$. Therefore, once customers begin walking, they become a “missed” customer within the MOD system.

To maximize the customer QoS in ride hailing environments, the goal is to maximize the number of served customers. If there are $c_n$ customer arrivals at node $n$, an unoccupied vehicle will need to come within proximity of each customer within a time of $t_{wait}$. Here, proximity is defined according to the sensing range of the autonomous vehicles, that is, a vehicle would need to see the customer with onboard camera and Lidar in order to serve them. Also, the wait time for each customer can vary. To account for this, the limiting case where $t_{wait} = 0$ is used. Therefore, for each customer node arrival, a vehicle will need to already be waiting at that node before the arrival occurs. For a given time period, $t_{pred}$, the number of vehicles $v_n$ that need to be waiting at node $n$ will be equal to the number of customer arrivals, $v_n = c_n$, in order to serve all customers. In many MOD systems such as the MIT campus MOD system, the number of possible customer arrival locations is larger than the number of available vehicles. Therefore, vehicles will need to be selectively placed only at certain customer arrival locations. The goal of the ride hailing planner is to determine how to
selectively place vehicles at nodes within the network graph while taking into account the likelihood and uncertainty in customer arrival estimates.

6.2 Ride Hailing Planner

This section presents two ride hailing planners for assigning vehicles to nodes in order to maximize the number of served arriving customer. Because the number of customer arrivals at each node is uncertain, robust assignments will need to incorporate the probability distribution for the customer arrivals presented in Equation (5.4). To capture the customer arrival uncertainty, expected value and chance constrained problem formulations are used to generate ride hailing planners.

6.2.1 Problem Formulation

Let \( c_n \) be the number of customers arriving at node \( n \) in a time period of \( t_{\text{pred}} \). Let \( v_n \) be the number of vehicles assigned to \( n \), where \( N_v \) is the total number of available vehicles. For a given vehicle assignment, the node cost, \( C_n \), is chosen to be

\[
C_n = (c_n - v_n)^2. \tag{6.1}
\]

This quadratic cost function is chosen so that 1) in the case where there are fewer customer arrivals than vehicles in the system, excess vehicles at a node are penalized in order to have more vehicles available for exploration; and 2) in the case where there are more customer arrivals than vehicles, vehicles are assigned proportionally to the number of customer arrivals. The problem can be formulated as an integer quadratic program with \( v_n \in \{1, \ldots, N_v\} \) as the decision variables. The ride hailing problem formulation is,

\[
\arg\min_{v_n} \sum_{n=1}^{N_n} (c_n - v_n)^2 \quad \text{s.t.} \quad \sum_{n=1}^{N_n} v_n \leq N_v. \tag{6.2}\]

95
Equation (6.3) ensures that vehicle assignments do not exceed the fleet size. The challenge for the problem formulation stems from the uncertainty in the number of customer arrivals at each node. The formulation is adapted to address the uncertainty by using expected value and chance-constrained approaches.

The expected value formulation assigns vehicles based on the expected number of customers and is given by

$$\arg\min_{v_n} E \left[ \sum_{n=1}^{N_n} (c_n - v_n)^2 \right]$$

subject to

$$\sum_{n=1}^{N_n} v_n \leq N_v.$$  \hspace{1cm} (6.5)

The expectation is taken over all customer arrivals. The objective in Equation (6.4) can be rewritten as

$$\arg\min_{v_n} \sum_{n=1}^{N_n} (E[c_n] - v_n)^2,$$  \hspace{1cm} (6.6)

where $E[c_n]$ is the expectation according to Equation (5.4). Equation (6.6) is derived from Equation (6.4) by adding a constant term $\text{Var}[c_n]$ to the objective and using the fact that the probability of customer arrivals at each node is independent.

The chance-constrained formulation bounds the total vehicle assignment cost to be within a threshold. The chance-constrained formulation is

$$\arg\min_{v_n} X$$

subject to

$$\Pr \left( \sum_{n=1}^{N_n} (c_n - v_n)^2 \leq X \right) \geq \eta,$$  \hspace{1cm} (6.8)

$$\sum_{n=1}^{N_n} v_n \leq N_v.$$  \hspace{1cm} (6.9)

where $X$ is a decision variable representing the total cost and $\eta$ is a predetermined risk tolerance threshold. The constraint in Equation (6.8) uses the joint probability of the sum of random customer arrivals, which is difficult to compute. Instead, a more constrained version of the problem is formulated, where the bound is made for each
node cost. The node chance-constrained formulation is

\[
\arg\min_{v_n} \sum_{n=1}^{N_n} X_n \quad \text{(6.10)}
\]

\[
s.t. \quad P \left( (c_n - v_n)^2 \leq X_n \right) \geq \eta_n \quad \text{(6.11)}
\]

\[
\sum_{n=1}^{N_n} v_n \leq N_v, \quad \text{(6.12)}
\]

where \( X_n \) are decision variables representing the cost incurred at each node and \( \eta_n \) are risk tolerance thresholds for each node. Equation (6.11) constrains the cost at each node according to the customer arrival probability. The constraint can be rewritten as,

\[
P \left( (c_n - v_n)^2 \leq X_n \right) \geq \eta_n
\]

\[
P \left( v_n - \sqrt{X_n} \leq c_n \leq v_n + \sqrt{X_n} \right) \geq \eta_n
\]

\[
F \left( v_n + \sqrt{X_n} \right) - F \left( v_n - \sqrt{X_n} \right) \geq \eta_n, \quad \text{(6.13)}
\]

where \( F(\cdot) \) represents the cumulative distribution function (CDF) for the customer arrivals, computed using Equation (5.4).

6.2.2 Online Approach

The ride hailing planners operate on a fixed planning horizon of length \( t_{pred} \). At the start of the planning horizon, the number of vehicles to assign to each node is determined. After determining the number of vehicles to assign, actual vehicle assignments are made using greedy assignment. Vehicles are continually assigned over the course of the time horizon to satisfy the determined node assignment numbers. For example, if a vehicle encounters a customer and leaves an assigned node to serve them, any unassigned vehicles in the system will be assigned to take its place. The main challenge for the online approach is to determine the number of vehicles to assign.

For the expected value ride hailing planner, the expected number of customers arrivals at each node for \( t_{pred} \) is first computed. Rather than compute the expectation
from Equation (5.4), an iterated expectation is computed over the parameters. The expectation is given as,

$$\mathbb{E}[c_n] = \mathbb{E}[p_n] \mathbb{E}[\lambda_n] = \frac{a_n \alpha_n}{a_n + b_n \beta_n}, \quad (6.14)$$

where $\mathbb{E}[p_n]$ and $\mathbb{E}[\lambda_n]$ are known for Beta and Gamma distributions, respectively. Equations (6.5) and (6.6) are then solved using integer quadratic programming.

For the chance-constrained ride hailing planner, the problem is not as easily solved because Equation (6.13) introduces a nonlinear constraint. Instead, the problem is decomposed into two sub-problems. The first problem determines the minimum cost for each vehicle at each node, represented by the cost matrix $K \in R^{N_v \times N_n}$. The problem is formulated as,

$$K(v_n, n) = \arg\min_{X_n} X_n \quad (6.15)$$

$$s.t. \quad F(v_n + \sqrt{X_n}) - F(v_n - \sqrt{X_n}) \geq \eta_n. \quad (6.16)$$

$K$ is computed through enumeration over all nodes and number of vehicles. The second problem determines the optimal number of vehicles to assign to each node to minimize the total cost. The problem is formulated as,

$$\arg\min_{v_n} \sum_{n=1}^{N} K(v_n, n) \quad (6.17)$$

$$s.t. \quad \sum_{n=1}^{N} v_n \leq N_v, \quad (6.18)$$

which is solved using integer linear programming.

One challenge for the chance-constrained planner is that of determining the appropriate risk allocation thresholds for each node. External knowledge of the MOD network could be used to allocate more risk to certain nodes than others. In this work, a uniform risk allocation is used where each node is assigned the same risk tolerance value.
6.2.3 Planner Comparison

The expected value and chance-constrained formulations both utilize the belief in the number of customer arrivals. The expected value formulation assigns vehicles based on the expectation of the belief, and is robust only in the sense that the planner should perform well on average. The chance-constrained formulation uses the full posterior information of the belief through the CDF and allows the cost at each node to vary according to the risk threshold for that node. To develop a general understanding of the difference between the two planners, each is tested in simulation. Poisson customer arrivals for a 33 node network graph are generated from a negative binomial approximation, where each node has mean arrival rate of 1 ped/min but with a varied variance. Each planner can assign any number of vehicles to each node. The chance-constrained risk tolerance is set to $\eta_n = 0.99$ to better emphasize the difference. Two performance metrics are studied based on the node cost metric from Equation (6.1), where $c_n$ is the actual number of customer arrivals and $v_n$ is the number of vehicles that are chosen to be assigned. The first metric is the mean cost over all nodes, that is $\bar{C} = \frac{1}{N_n} \sum_{n=1}^{N_n} C_n$. The second is the maximum cost for any single node, that is $C_{max} = \max(C_1, \ldots, C_{N_n})$. Figure 6-1 shows comparisons of expected value and chance-constrained planners under increasing amounts of variance. The expected value planner has lower mean cost because it assigns the correct number of vehicles on average, with no consideration towards potentially high node costs. The chance-constrained planner has lower maximum cost because it assigns more vehicles in order to bound any potentially high individual node cost, which comes at the price of higher mean cost. The choice of planner ultimately comes down to the goal of the MOD system, whether it is more important to perform well on average or to bound performance on any individual node.

6.3 Experimental Testing

The ride hailing planners are tested using simulation of the MIT MOD system. The goal of the testing is to evaluate how well the planners can increase the number of
Figure 6-1: Comparison of expected value and chance-constrained planners. Arrivals are generated from a negative binomial distribution, with mean of 1 ped/min at each node and varying standard deviation. Cost is measured as (a) mean cost or (b) maximum cost across all nodes, lower is better. Note that maximum costs are an order of magnitude higher than the mean costs. Mean and standard deviation are shown for 1000 simulated arrivals.
served hailing customers.

6.3.1 Simulation Setup

Testing is performed using the simulator presented in Section 5.4.1. Pedestrians and vehicles move within the full network graph for the MIT campus. Pedestrian route arrival rates are set to reflect the true arrival rates shown in Figure 4-5. In each round of simulation, ten randomly chosen routes with separate origin nodes are assigned pedestrian arrival rates of 1 ped/min. Three of the ten routes will be customer arrival routes, where customers wait at their origin node for a vehicle. Customers perform ride hailing with a wait time of $t_{\text{wait}} = 30$ seconds. Customers are served if a vehicle comes within a 20 meter proximity, otherwise the customer will instead walk their route. The simulation considers 5 vehicles in the MIT MOD system. A period of 1 hour is simulated.

6.3.2 Ride Hailing Approaches

The ride hailing planner operates on a 5 minute planning horizon. At the start of the planning horizon, the number of vehicles to assign to each node is first determined from the ride hailing planner. If fewer vehicles are needed than there are vehicles in the system, the remaining vehicles are assigned to drive routes according to the exploration planner, where assigned routes have a length of 5 links. In either case, if a vehicle encounters a hailing customer, the vehicle immediately serves the customer.

Four different ride hailing planners are analyzed. The first two are the expected value and chance-constrained planners. The third is a sensing planner that represents a baseline approach where vehicles continually patrol the network in order to happen upon a hailing customer. The planner is implemented by never assigning vehicles to wait at nodes so that they are always assigned to traverse links through the exploration planner. Finally, the fourth is an oracle planner designed to represent the upper bound on performance for serving customers. The planner is implemented by providing the true number of customer arrivals over the planning horizon to the expected value
planner, causing vehicles to often be assigned to wait at nodes.

6.3.3 Ride Hailing Evaluation

The performance of the ride hailing planners is evaluated in terms of two performance metrics. The first is the average fraction of customers served across the whole time period. The second is the average highest cost for any individual planning horizon given by

\[
\text{Cost} = \frac{1}{12} \sum_{i=1}^{12} \max(C_{1,i}, \ldots, C_{N_n,i}),
\]

(6.19)

where \( C_{n,i} \) is Equation (6.1) evaluated over the \( i \)th 5 minute planning horizon. With the chance-constrained formulation, uncertainty is managed according to a risk tolerance parameter for each node \( \eta_n \), the effect of which is studied using several chance-constrained planners with varying risk tolerance values. Figure 6-2 show the two performance metrics for each of the planners. First, the results indicate that a higher risk tolerance generally leads to better performance with \( \eta_n = 0.9 \) performing best. Second, the performance of the oracle planner demonstrates the difficulty of the problem, in that a cost is still incurred due to there being more customer arrivals than vehicles available. Third, the chance-constrained planner demonstrates an advantage over the expected value planner with a 4% improvement in the fraction of customers served and 7% reduction in maximum cost. Finally, the chance-constrained planner significantly outperforms the baseline sensing planner with a 73% improvement in the fraction of customers served and a 30% reduction in maximum cost.

6.4 Summary

This chapter presents an MOD fleet management framework that addresses the challenges of utilizing customer demand for ride hailing environments in order to maximize the number of served customers. Two ride hailing planners were introduced that assign vehicles to wait at customer arrival locations that are subject to arrival rate uncertainty. Data-driven simulation of the MIT MOD system was used to validate
Figure 6-2: Comparison of sensing, expected value, oracle, and chance-constrained ride hailing planners. The chance-constrained strategy is compared with several risk allocation values as indicated by the x-axis. (a) Performance is measured as the fraction of customer population that is served, higher is better. (b) Cost is measured as the highest cost of any time horizon, lower is better. Mean and standard deviation shown for 100 simulated runs.
the performance of the planners. The chance-constrained ride hailing planner was shown to improve the fraction of served customers in the system by 73% over a baseline exploration approach. Together with the customer demand model presented in Chapter 5, the ride hailing fleet management planner is able to successfully learn and utilizes customer demand to improve the number of customers that are served in an MOD system.
Chapter 7

Fleet Management for Ride Requesting

This chapter addresses the challenge of managing an autonomous vehicle fleet in order to maximize the QoS of customers in an MOD system that features ride requesting. A ride requesting MOD framework allows for customers to submit ride requests to a centralized planner via a smartphone app. This allows customers to be served by routing vehicles to the customer’s arrival location. This additional MOD capability changes the QoS measure from that of the ride hailing framework presented in Chapter 6. Rather than a simple binary metric, QoS is instead defined as the amount of time customers have to wait after requesting a ride. For example, customers who are served by the MOD system but experience a long wait time may be dissatisfied with the service. Much like with the ride hailing planner, a customer demand model can be used to estimate likely customer arrival locations. A predictive positioning planner is presented that utilizes the demand model to minimize the expected wait time for ride request customers.

In addition to overcoming the need for direct hailing, a ride request framework allows for ridesharing, where multiple customers may share a ride in an MOD vehicle at the same time. Customer requests provide both arrival and destination location information, which MOD vehicles use to generate routes in order to serve customers. However, this additional capability also creates even more complexity in QoS measures.
To minimize wait times, newly arrived passengers can be picked up before onboard passengers have been dropped off, allowing for more customers to be serviced with fewer vehicles. However, the reduced wait time for requesting passengers comes at the expense of increased ride time of onboard passengers. With ridesharing, perceived QoS becomes dependent on customer preference (i.e. how much customers prefer one service metric over another). A ridesharing planner is presented that learns customer preference in order to route vehicles to customer arrival and destination locations so that overall customer QoS is maximized.

7.1 Customer QoS Model

Customers are assumed to arrive at nodes in the network graph according to a Poisson process with arrival rate parameter $\lambda_c$. A discrete-time approximation is used where Poisson arrival rates are static for the operating duration, with short-term fluctuations averaged out across the time period. Upon arrival, customers send ride requests consisting of a pickup node $p \in \mathcal{N}$ and a drop off node $e \in \mathcal{N}$.

7.1.1 Customer QoS without ridesharing

With ride requesting without ridesharing, vehicles are assigned to pick up customers when a request is received. A vehicle assigned to a customer is routed first to the customer’s pickup node and then to the customer’s dropoff node. If the vehicle is not at the arrival node of the customer when the request is made, the customer will experience an adverse wait time delay. However, because vehicles only serve one customer at a time, once a customer is picked up, driving them directly to their destination is assumed to create ideal the QoS from that point forward. Customers who experience a large wait time will perceive a poor QoS, and if the wait time is large enough, the preference would be to use another form of transportation, such as walking. A centralized ride request planner can actually provide wait time estimates to customers who have requested a ride, and allow them to cancel the request if the wait time is too high. As a result, the ride hailing QoS measure of maximizing the
number of served customers becomes correlated with minimizing the wait time for newly arriving customers. Therefore, in ride request environments without ridesharing, customer QoS is maximized by minimizing the wait time for customers.

The following provides detail for computing the wait time for ride request customers. Customer \( c \) makes a request with pickup node \( p_c \) at the point in time \( t^\text{request}_c \). The customer is assigned to an available vehicle \( v \) currently waiting at node \( n_v \). The vehicle is assigned route \( r(n_v, p_c) \) in order to pickup the customer. It is assumed that the travel time, \( t(n_i, n_j) \), between any two nodes \( n_i \) and \( n_j \) is known for all routes in the network graph. The wait time, \( t^\text{wait}_c \), is simply \( t^\text{wait}_c = t(n_v, p_c) \). In order to reduce the wait time for an expected customer arrival, the vehicle’s waiting location \( n_v \) should be as close as possible as the customer’s expected arrival location, \( p_c \). However, there are many possible arrival locations and relatively few MOD vehicles. To address this, a predictive positioning planner is developed that utilizes a customer demand model in order to assign available MOD vehicles to wait at nodes that minimize expected customer wait time.

### 7.1.2 Customer QoS with ridesharing

With ridesharing, vehicles can stop to pick up another customer before dropping off onboard customers. Rather than being assigned to pickup a single passenger, vehicles are assigned a routing schedule to pickup or dropoff multiple passengers. A centralized ridesharing planner takes a set customer requests that are split up and assigned to a set of vehicles, where each vehicle has a maximum capacity of \( Q \). Let \( C \) of size \( N_c \) be the set of customers who have requested rides, let \( O \) of size \( N_o = 2N_c \) be the set of requested customer pickup and drop off nodes, and let \( V \) of size \( N_v \) be the set of vehicles in the MOD fleet. Vehicle \( v \) will service a subset of the customers \( C_v \subseteq C \) by visiting their pickup and drop off nodes. All customer nodes are inserted into a schedule \( s_v = \{s_1, \ldots s_{N_o}\} \), \( s_i \in \{\emptyset, \mathcal{N}\} \), that the vehicle traverses in order using a sequence of routes. The metrics (wait time, ride time, etc.) for a customer will depend on their relative position within the schedule.

Customer QoS is quantified in terms of a set of transportation metrics, similar to
those in [47]. Let \( \mathbf{m}_c \in \mathbb{R}^{11} \) be the set of QoS transportation for customer \( c \), where the elements of \( \mathbf{m}_c \) are: ride time \( t^{\text{ride}} \), wait time \( t^{\text{wait}} \), service time \( t^{\text{service}} \), ratio of ride time to direct time \( t^{\text{ratio}} \), excess ride time \( t^{\text{excess-ride}} \), number of stops while user is onboard \( N^{\text{stops}} \), the notification time \( t^{\text{notify}} \), the total traveled distance while onboard \( d^{\text{traveled}} \), the time it would have taken the customer to walk \( t^{\text{walk}} \), and service time in excess of walk time \( t^{\text{excess-walk}} \). The following provides detail for computing the set of transportation metrics for ridesharing customers. Customer \( c \) makes a request at the point in time \( \hat{t}_c^{\text{request}} \). The customer request is either rejected or accepted by the MOD system, where metric \( I^{\text{rej}}i \) is a rejection indicator variable that takes value 1 if the customer is rejected and 0 otherwise. If rejected, the customers will walk. If accepted, the customer is assigned to vehicle \( v \) currently located at node \( n_v \) and the pickup \( p_c \) and drop off \( e_c \) nodes for \( c \) are inserted into respective nodes \( s_i \) and \( s_j \) in the vehicle schedule \( s_v \). The vehicle travels between any adjacent nodes \( s_k \) and \( s_{k+1} \) in its schedule using route \( r(s_k, s_{k+1}) \). The travel distance and travel time between the nodes are

\[
\begin{align*}
    d(s_k, s_{k+1}) &= \sum_{l \in L(r(s_k, s_{k+1}))} d_l, \quad (7.1) \\
    t(s_k, s_{k+1}) &= \sum_{l \in L(r(s_k, s_{k+1}))} \frac{d_l}{u_l}, \quad (7.2)
\end{align*}
\]

where \( d_l \) and \( u_l \) are the length and average travel speed of link \( l \), respectively. The set of metrics are computed as follows:

\[
\begin{align*}
    \hat{t}_c^{\text{pickup}} &= t(n_v, s_1) + \sum_{k=1}^{i-1} t(s_k, s_{k+1}), \quad (7.3) \\
    \hat{t}_c^{\text{dropoff}} &= t(n_v, s_1) + \sum_{k=1}^{j-1} t(k_k, s_{k+1}), \quad (7.4) \\
    t_c^{\text{direct}} &= t(p_c, e_c), \quad (7.5) \\
    d_c^{\text{direct}} &= d(p_c, e_c), \quad (7.6) \\
    t_c^{\text{walk}} &= \tilde{t}(p_c, e_c), \quad (7.7)
\end{align*}
\]
\[ t_{\text{ride}}^c = \hat{t}_{\text{dropoff}}^c - \hat{t}_{\text{pickup}}^c, \quad (7.8) \]
\[ t_{\text{wait}}^c = \hat{t}_{\text{pickup}}^c - \hat{t}_{\text{request}}^c, \quad (7.9) \]
\[ t_{\text{service}}^c = t_{\text{wait}}^c + t_{\text{ride}}^c, \quad (7.10) \]
\[ t_{\text{ratio}}^c = t_{\text{ride}}^c / t_{\text{direct}}^c, \quad (7.11) \]
\[ t_{\text{excess-ride}}^c = t_{\text{ride}}^c - t_{\text{direct}}^c, \quad (7.12) \]
\[ N_{\text{stops}}^c = k - j, \quad (7.13) \]
\[ d_{\text{traveled}}^c = \sum_{k=i}^{j-1} d(s_k, s_{k+1}), \quad (7.14) \]
\[ t_{\text{excess-walk}}^c = t_{\text{service}}^c - t_{\text{walk}}^c, \quad (7.15) \]

where \( \hat{t}_{\text{pickup}}^c \) is the point in time \( c \) is picked up, \( \hat{t}_{\text{dropoff}}^c \) is the point in time \( c \) is dropped off, \( t_{\text{direct}}^c \) is the time it would take to drive directly from \( p_c \) to \( e_c \), \( d_{\text{direct}}^c \) is the direct route distance between \( p_c \) and \( e_c \), \( t_{\text{walk}}^c \) is the time it would take to walk from \( p_c \) to \( e_c \), \( \hat{t}(n_i, n_j) \) is Equation (7.2) evaluated with \( r(n_i, n_j) \) and \( v_l \), as the respective route and speed of the pedestrian instead of a vehicle, and \( \hat{t}_{\text{assigned}}^c \) is the point in time when the ridesharing algorithm assigns the customer to the vehicle. Note, if a customer is rejected \( (1_{\text{rej}}^c = 1) \), then many of the metrics do not apply and the customer metrics are set to be \( m_{\text{rej}}^c = \{1_{\text{rej}}^c, t_{\text{notify}}^c, t_{\text{walk}}^c\} \).

While these metrics describe the objective experience that a customer receives, the actual perceived quality will be more subjective. To quantify QoS, the vector of service metrics are mapped to an overall QoS measure, \( g \in \mathbb{R} \). To maximize true customer QoS, a ridesharing planner is developed that learns and utilizes the mapping between customer experience metrics and perceived customer QoS.

### 7.2 Predictive Positioning for Minimizing Wait Time

This section introduces the predictive positioning approach for minimizing expected customer wait time. The approach manages vehicles in the absence of ride requests.
by using a predictive positioning algorithm to identify predictive nodes within the network graph to place unassigned vehicles such that the expected wait time of new customers is minimized.

7.2.1 Predictive Positioning Formulation

The goal of the predictive positioning algorithm is to determine the set of nodes for which to place unoccupied vehicles such that the wait times for a set of newly arriving customers will be minimized under expectation. Let the vector \( \mathbf{a} \in \{0, 1, \ldots, N_a\}^{N_a} \) be an arrival set with elements corresponding to the number of customer arrivals on each node for the next \( N_a \) total arrivals. Let the vector \( \mathbf{k} \in \{0, 1, \ldots, N_v\}^{N_v} \) denote an assignment set with elements corresponding to the number of vehicles assigned to wait at each node, where \( N_v \) is the number of available vehicles in the MOD system. If the number of considered customers is set to be the number of vehicles, \( N_a = N_v \), then the wait time would be minimized by assigning vehicles to the same customer arrival nodes, that is \( \mathbf{k} = \mathbf{a} \).

However, there is a large set of possible arrival sequences and only one choice for vehicle assignments. Let \( \mathcal{A} = \{\mathbf{a} \mid \sum_{i=1}^{N_a} a_i = N_a\} \) be the set of all combinations of possible arrivals and let \( \mathcal{K} = \{\mathbf{k} \mid \sum_{i=1}^{N_v} k_i = N_v\} \) be the set of all possible vehicle assignment options. Unlike with ride hailing, vehicles do not need to be exactly at customer arrival nodes. Given a vehicle assignment set, customers from each possible customer arrival sequence can still be served, albeit with induced wait times. For a given customer arrival set \( \mathbf{a} \) and vehicle assignment set \( \mathbf{v} \), the wait time cost \( w_{\mathbf{k}, \mathbf{a}} \) can be determined. First, an assignment approach is used to assign each customer to each vehicle based on route travel times. The total wait time is the sum of the route travel times, that is

\[
w_{\mathbf{k}, \mathbf{a}} = \sum_{v=1}^{N_v} \sum_{a=1}^{N_a} t(n_v, n_a) 1(a, v),
\]

where \( n_v \) is the waiting node of vehicle \( v \), \( n_a \) is the arrival node of customer \( a \), \( t(n_v, n_a) \) is the route travel time between \( n_v \) and \( n_a \), and \( 1(a, v) \) is an indicator function that is 1 if \( a \) is assigned to \( v \) and 0 otherwise.
Using precomputed wait times, one possibility would be to assign vehicles to nodes that minimize over all possible arrivals. However, not all arrival sequences are equally likely. Using the parameter estimation approach presented in Chapter 5, it is assumed that knowledge of customer arrival rates \( \{ \lambda_{c}^{1}, \ldots, \lambda_{c}^{N_{n}} \} \) can be obtained. These arrival rates can be used to determine the likelihood of each arrival sequence. Because arrivals are modeled using Poisson processes, the probability of a given arrival set \( P(a) \) can be determined using the decomposition property of a merged total network arrival Poisson process. The total number of arrivals in the network graph follows a Poisson process with rate parameter \( \Lambda = \sum_{n=1}^{N_{n}} \lambda_{c}^{n} \). Given that an arrival occurs, the probability of that arrival occurring at a node \( n \) is given by \( P(a_{n} = 1 \mid N_{a} = 1) = \frac{\lambda_{c}^{n}}{\Lambda} \). Following the probability of the order of events for a sequence of Poisson arrivals presented in Section 2.1, the probability of a set of arrivals follows a multinomial distribution, that is

\[
P(a) = \frac{N_{a}!}{a_{1}! \cdots a_{N_{n}}!} \left( \frac{\lambda_{c}^{1}}{\Lambda} \right)^{a_{1}} \cdots \left( \frac{\lambda_{N_{n}}^{n}}{\Lambda} \right)^{a_{N_{n}}}
\] (7.17)

Rather than minimize over all possible arrivals, the predictive positioning planner minimizes customer wait time based on expected arrival sequences. The predictive node locations \( k^{*} \) for which to place vehicles is determined by

\[
k^{*} = \arg\min_{k \in K} \mathbb{E}_{a \in A} \left[ w_{k,a} \right]
\] (7.18)

\[
k^{*} = \arg\min_{k \in K} \sum_{a \in A} w_{k,a} P(a).
\] (7.19)

### 7.2.2 Online Implementation

The predictive node locations \( k^{*} \) for which to place vehicles is determined using Algorithm 1. The algorithm takes as input the customer arrival rates and determines the optimal vehicle placement. When computing the wait times in line 7, a greedy assignment approach is used. The wait times are assumed to be dependent only on the structure of the network graph, and are therefore computed once for all possible arrival and assignment sets and stored offline. To handle the variation in the number of available vehicles as some vehicles are serving customers while others need to be
Algorithm 1: Predictive Positioning

1. **Input:** customer node arrival rates \( \{\lambda_1, \ldots, \lambda_n\} \)
2. **Output:** vehicle locations \( k^* \)
3. enumerate vehicle placement options \( K \)
4. enumerate possible arrival locations \( A \)
5. for \( a \in A \) do
   6. for \( k \in K \) do
      7. \( w_{k,a} \leftarrow \text{computeTotalWaitTime}(k, a) \)
      8. \( p_a \leftarrow \text{computeProbability}(a) \)
   9. \( k^* = \arg\min_{k \in K} \sum_{a \in A} w_{k,a}p_a \)
10. return \( k^* \)

assigned to predictive nodes, wait times are computed for all possible numbers of free vehicles. Arrival set probabilities in line 8 are computed any time customer arrival rate estimates are made. The predictive nodes \( k_{N_v^*}^* \) are computed for all possible number of unassigned vehicles between one and the total number of MOD vehicles. If \( N_v \) are available, they will be assigned and routed to the corresponding predictive nodes in \( k_{N_v^*}^* \) based on a greedy assignment. Once a customer arrives, one of the vehicles will be assigned, and the \( N_v - 1 \) remaining vehicles will again be assigned and routed to the corresponding predictive nodes in \( k_{N_v-1}^* \).

### 7.3 Quality of Service Ridesharing

This section introduces the ridesharing fleet management approach for maximizing customer QoS. The approach manages vehicles in the presence of ride requests using a ridesharing algorithm that performs new customer assignments to vehicles based on a customer QoS focused cost function. Two QoS focused cost functions are presented, a traditional weighted cost function, and a customer rating model that learns customer QoS preference from customer rating feedback.
7.3.1 Ridesharing Formulation

The ridesharing problem can be formulated as a Dial-A-Ride Problem (DARP), a subset of the vehicle routing problem that focuses on the pickup and dropoff of passengers. A schedule-based DARP formulation is proposed as an extension of the models presented in [21], with emphasis placed on the ordering of customers within schedules. The formulation takes as input the pickup and dropoff locations of customer requests and produces an ordered assignment of MOD vehicles. Assignments take the form of inserting the pickup $p_c$ and dropoff $e_c$ nodes of customer $c$ within the schedule of vehicle $v$, where nodes $s_i$ and $s_j$ within the schedule are assigned to be $p_c$ and $e_c$ respectively. A four-index decision variable is used, $x^v_{ci,j} \in \{0, 1\}$, that takes value 1 if vehicle $v$ is assigned customer $c$, with $p_c$ and $e_c$ respectively sequenced at $s_i$ and $s_j$; and zero otherwise. Similarly, each assignment incurs a cost $g^v_{ci,j}$. The ridesharing problem can be formulated as an integer linear program (ILP) as follows,

\[
\arg\min_{x^v_{ci,j}} \quad \sum_{v \in V} \sum_{c \in C} \sum_{i=1}^{N_o} \sum_{j=1}^{N_o} g^v_{ci,j} x^v_{ci,j} \tag{7.20}
\]

\[
s.t. \quad \sum_{v \in V} \sum_{c \in C} \sum_{i=1}^{N_o} \sum_{j=1}^{N_o} x^v_{ci,j} = N_c \tag{7.21}
\]

\[
\sum_{v \in V} \sum_{c \in C} \sum_{i=1}^{N_o} \sum_{j=1}^{N_o} x^v_{ci,j} = 1 \quad \forall c \in C \tag{7.22}
\]

\[
\sum_{i=1}^{N_o} \sum_{j=1}^{N_o} x^v_{ci,j} = 0 \quad \forall v \in V, c \in C \tag{7.23}
\]

\[
\sum_{c \in C} x^v_{ci,j} \leq 1 \quad \forall v \in V, i, j \in \{1, \ldots, N_o\} \tag{7.24}
\]

\[
\sum_{c \in C} \sum_{i=1}^{N_o} \sum_{j=k+1}^{N_o} x^v_{ci,j} \leq Q \quad \forall v \in V, k \in \{1, \ldots, N_o\} \tag{7.25}
\]

\[
x^v_{ci,j} \in \{0, 1\} \quad \forall v \in V, c \in C, i, j \in \{1, \ldots, N_o\}.
\]

The objective in Equation (7.20) is to minimize the total customer QoS cost. Equations (7.21) and (7.22) enforces that all customers are assigned. Equation (7.23) enforces that a customer is picked up before being dropped off. Equation (7.24)
enforces that at most only one customer can occupy any given schedule position. 
Equation (7.25) enforces that the capacity of the vehicle is not exceeded.

The schedule-based DARP formulation focuses on assigning a cost based on the order that customers are served within schedules. One challenge for this formulation is that of determining the cost that is incurred for a given customer assignment. As described in Section 7.1.2, a customer’s QoS is affected through many factors that are quantified through a set of ride metrics \( m_c \), where the metrics themselves are a function of the customer’s position in a vehicle’s schedule. To account for this, the assignment cost in the DARP formulation is a potentially non-linear function of a customer’s ride metrics, that is \( g_{ci} = g(m_c) \). In order to evaluate the individual costs for a particular customer assignment, the other customer assignments would also need to be considered in order to determine the customer’s experienced ride metrics. This aspect along with the four-index formulation makes solving the schedule-based DARP problem difficult and potentially computationally intractable for large domains. In order to address this, an insertion approach is introduced that sequentially inserts customers into schedules in order to minimize a cost function that depends on customer ride metrics. The following presents the choices of metric-dependent cost functions and an online insertion approach that is used to approximately solve the schedule-based DARP problem.

### 7.3.2 Ridesharing Cost Functions

Two ridesharing cost functions are presented to evaluate the customer QoS cost from a set of customer ride metrics. First, a cost function composed of a linear weighted combination of the customer metrics is proposed as

\[
g(m_c) = \sum_{i=2}^{m_c-1} w_i^{\text{rej}} 1_c^{\text{rej}} m_{c,i} + w_i^{\text{acpt}} (1 - 1_c^{\text{rej}}) m_{c,i}, \tag{7.26}
\]

where \( w_i^{\text{rej}} \) and \( w_i^{\text{acpt}} \) are weights for metric \( i \) that are used to allow for differentiating between rejected and serviced customers. For example, a service time focused cost
function would be \( g(m_c) = 1^\text{rej} c^\text{walk} + (1 - 1^\text{rej}) c^\text{service} \), where the cost is the service time if the customer receives a ride and the walk time if they are rejected. This form of cost function requires that the weights be properly chosen to reflect customer preference, and can result in poor customer QoS if the weights are wrongly chosen.

To overcome the need to choose weights, a second ratings based cost function is presented that learns and uses customer preference through feedback from 5-star ratings. The rating model utilizes a random forest of classification decision trees, based on the work of [12]. Random forest algorithms tend to prevent overfitting and have been demonstrated to perform well empirically [16]. To train the random forest model, a dataset \( D \) from \( N_D \) customers is collected in the form of 5-star ratings \( Y_{\text{train}} = \{y_1, \ldots, y_{N_D}\} \) and a ride metrics feature vector \( X_{\text{train}} = \{m_1, \ldots, m_{N_D}\} \) such that \( Y_{\text{train}} = RF(X_{\text{train}}) \) where \( RF(X) \) is the trained random forest. The trained random forest then serves as the ridesharing cost function such that \( g(m_c) = -RF(m_c) \) where the minus sign is included to maximize customer rating. It is assumed that a customer’s 5-star rating is given purely to reflect their ride metrics and not factors such as driver interactions that are not present for autonomous MOD systems.

### 7.3.3 Online Approach

While the DARP formulation for the ridesharing problem can be formulated as an ILP, certain choices of cost function such as the random forest model will result in a nonlinear objective function that is difficult to solve analytically. To overcome this, a heuristic solution to the ridesharing problem is proposed that uses an insertion approach. In the insertion approach, the request, \( \{p_c, e_c\} \), of a new customer \( c \) is temporarily inserted into each of the existing vehicle schedules, \( \{s_1, \ldots, s_{N_S}\} \), to determine the additional cost that will be incurred. The schedule that incurs the least additional cost will be updated to accommodate the new request. The approach is illustrated in Figure 7-1, where a requesting customer can be inserted into an existing schedule in multiple ways and the considered cost is the rating of every customer.

Algorithm 2 presents the method for assigning a new customer request to a vehicle such that the total QoS cost to the system is minimized whenever a new customer ride
Figure 7-1: Ridesharing assignment diagram. The pickup and dropoff of a newly requesting customer (orange) is inserted into a vehicle schedule consisting of pickups and dropoffs of already assigned customers (blue, green, red). The choice of where to insert (or possibly reject) the new customer will have an impact on the rating of each customer.

request is submitted. First, line 4 computes the baseline customer QoS metrics $M_i$ for each customer already within each vehicle’s schedule. Next, line 5 enumerates all feasible ways of inserting the new request into the existing schedule, where $\hat{S}_v$ is the set of all feasible schedules $s_v$. Feasibility is met by ensuring that $p_c$ is inserted before $e_c$ and that the vehicle capacity is not exceeded. Line 6 determines the best feasible schedule $s_v^*$ based on the ride hailing cost function. Line 7 stores the new metrics $M_i^*$ for each customer (including the new customer) under the best assignment. In addition to the evaluating how metrics change by assigning the customer to each vehicle, the algorithm also computes the metrics for the case where the new customer is rejected by the MOD system in line 8. In that case, the metrics are associated with a virtual “rejection vehicle”, $v = N_v + 1$. Line 9 determines which combination of baseline and new metrics provides the lowest overall QoS cost, and then returns the vehicle, $v^*$, that is assigned the customer. If $v^*$ is the rejection vehicle, then the customer is rejected, otherwise, the schedule for $v^*$ is updated in Line 10 to accommodate the request.

While seemingly counter-intuitive, rejections are in fact important for improving customer QoS and the use of a virtual rejection vehicle overcomes the need to impose rejection constraints on customer QoS metrics. For example, if a customer’s wait time
Algorithm 2: Ridesharing

1 **Input:** request \( \{p_c, e_c\} \), schedules \( \{s_1, \ldots, s_{N_v}\} \)
2 **Output:** updated schedule for assigned vehicle \( s_v^* \)
3 for \( v = 1 : N_v \) do
4 \( M_v \leftarrow \text{computeMetrics}(s_v) \)
5 \( S_v \leftarrow \text{enumerateInsertions}(s_v, p_c, e_c) \)
6 \( s_v^* = \arg\min_{\hat{s}_v \in S_v} g(\text{computeMetrics}(\hat{s}_v)) \)
7 \( M_v^* \leftarrow \text{computeMetrics}(s_v^*) \)
8 \( M_{N_v+1}^* \leftarrow \text{computeRejectionMetrics}(p_c, e_c) \)
9 \( v^* \leftarrow \text{assignToLowestBid}(\{M, M^*\}_{1:N_v}, M_{N_v+1}^*) \)
10 \( s_{v^*} \leftarrow s_v^* \)
11 return \( s_v^* \)

is significantly longer than the time it would take for them to walk, then they may prefer to be rejected rather than to wait to use the service. Other methods [22, 26, 35] encode feasibility constraints on customer metrics and reject the customer if the wait time exceeds a threshold. In this approach, no constraint is hard-coded into the algorithm, but is rather handled by the rejection vehicle that makes a bid on how much cost would be incurred by having the customer walk. This is a more general approach where a competing bid is made to reject a customer, and the rejection is made only when the overall QoS of the system would be improved by doing so. The primary benefit of the ridesharing algorithm is the ability to evaluate customer QoS without having to encode the customer preference structure into the algorithm.

7.4 Experimental Testing

The predictive positioning and ridesharing methods are tested using simulation of the MIT MOD system. Specifically, there are two motivating test cases: 1) evaluating how wait times for customers are affected under different fleet management strategies as customer arrival rates grow; and 2) evaluating how customer QoS is affected by various ridesharing strategies operating under a range of unknown customer preference models. The MIT MOD system is used to provide simulation parameters that reflect a realistic operating environment for vehicles and customers.
7.4.1 Simulation Setup

Pedestrians and vehicles operate within the full network graph for the MIT campus. A two hour time period is simulated; during which time a subset of 10 randomly chosen nodes are assigned static customer arrival rates between 0 and 0.45 ped/min. There are 3 vehicles in the MIT MOD system each with a maximum capacity of 3 passengers. Vehicles travel between nodes according to the schedule generated by the ridesharing algorithm. A vehicle picks up or drops off its assigned customers upon reaching a scheduled node. If a vehicle’s schedule is empty, the vehicle will travel to nodes prescribed by the predictive positioning algorithm.

In addition, a simulated rating model is used as ground truth in simulation to assign 5-star ratings to MOD customers. The values and functional forms are chosen based on an assumed customer preference and are kept hidden from the 5-star rating model. If a customer is rejected, then they give either a 1 or 2 star rating based on how long they waited to be notified of their rejection. A rejected customer’s rating is

\[
\begin{align*}
    r_{\text{rejected}} &= \begin{cases} 
    2, & \text{if } \frac{t_{\text{notify}}}{t_{\text{walk}}} \leq 0.1 \\
    1, & \text{otherwise.}
    \end{cases} 
\end{align*}
\]

(7.27)

If a customer is given a ride, then the rating will be a weighted sum of 5 aggregate ratings based on wait time, ride time, service time, number of stops, and ride distance computed as

\[
r_{\text{accepted}} = w_1 r_c^{\text{wait}} + w_2 r_c^{\text{ride}} + w_3 r_c^{\text{service}} + w_4 r_c^{\text{stops}} + w_5 r_c^{\text{distance}},
\]

(7.28)

with

\[
    r_c^{\text{wait}} = \text{Range} \left( \frac{t_c^{\text{wait}}}{t_c^{\text{walk}} - t_c^{\text{direct}}}, 0, 1 \right)
\]

\[
    r_c^{\text{ride}} = \text{Range} \left( \frac{t_c^{\text{ride}} - t_c^{\text{direct}}}{t_c^{\text{walk}} - t_c^{\text{direct}}}, 0, 1 \right)
\]

\[
    r_c^{\text{service}} = \text{Range} \left( \frac{t_c^{\text{service}} - t_c^{\text{direct}}}{t_c^{\text{walk}} - t_c^{\text{direct}}}, 0, 1 \right)
\]

118
\[ r_c^{\text{stops}} = \max(1, 6 - \lambda_c^{\text{stops}}) \]
\[ r_c^{\text{distance}} = \text{Range}\left(\frac{d_c^{\text{traveled}} - d_c^{\text{direct}}}{d_c^{\text{direct}}}, 0, 0.5\right), \]

where \( \text{Range}(\alpha, \beta, \gamma) \) maps \( \alpha \) to the \( i \)-th interval of 5 exponentially spaced values between \( \beta \) and \( \gamma \) and assigns the value \( 6-i \) as the rating. Range values were chosen to reflect a set of possible expected customer satisfaction levels for each metric. For example, setting the \( \gamma \) value for \( r_c^{\text{distance}} \) to 0.5 reflects that customers would give the lowest rating once their journey distance exceeded the nominal distance by a factor of 0.5. Exponential spacing is used to cause ratings to drop off more quickly as metrics worsen for customers. The majority of customers follow the same set of weights \( w \), but some customers do not. To accommodate this, the weights are drawn from a Dirichlet distribution such that \( w \sim \text{Dir}(\hat{w}) \), where the concentration parameters \( \hat{w} \) represent the nominal weights for the population.

7.4.2 Predictive Positioning Evaluation

To evaluate the predictive positeng, simulation of the MIT MOD system is performed without the use of ridesharing (vehicle capacities are 1), with assignments evaluated using a minimum service time cost function. For each simulation, the arrival rates are assumed known and are used to generate the sets of predictive nodes using Algorithm 1. As an example, Figure 7-2 shows the predictive nodes that are determined for the possibilities of 1, 2, or 3 unallocated vehicles for a particular set of arrival rates. The predictive nodes take into account both the probability of the arrivals occurring and the vehicles' travel time to reach each node. If only a single vehicle is unallocated, it will tend to be positioned centrally within the network, but skewed towards large arrival rates. When more vehicles are unallocated, the predictive nodes are further spread out for better coverage.

The performance of the predictive positioning algorithm is evaluated over several simulations of the MOD system. For comparison, a baseline unmanaged MOD strategy is considered, where vehicles respond to pickup requests but are not repositioned.
after dropping off customers, which would represent a minimum fuel-cost approach. Figure 7-3 shows that predictive positioning is able to reduce customer service times compared to the baseline approach. When arrival rates are lower than 0.35 ped/min per vehicle, there is often time between arrivals for vehicles to reposition to the predictive nodes and service times can be reduced by up to 20%. As arrival rates increase, however, the benefits of predictive positioning are reduced as vehicles are continually allocated to requests. To reduce high normalized arrival rates, one solution could be to increase the number of vehicle. However, this would increase the capital expenses of the MOD system and may not be feasible for sudden increases in demand. Alternatively, total service times may be reduced by increasing vehicle capacities to allow for ridesharing.
Figure 7-3: Service time comparison of baseline and predictive positioning methods over a range of customer arrival rates. Lower service times are better. Without ridesharing, service times and wait times are directly correlated as ride times will be unaffected. The service times are normalized by the direct time so as not to penalize services times for longer routes. Arrival rates are normalized by the number of vehicles for generalization. Results show mean and standard deviation for 100 iterations.

7.4.3 Ridesharing Benefits

To directly evaluate the benefits of ridesharing, the same predictive positioning test was performed but with the maximum vehicle capacity increased to 3. Customers are allocated to vehicles using the ridesharing algorithm in Algorithm 2, where the service time cost function is used. Figure 7-4 shows the effect of ridesharing on customer service times in both the baseline and predictive positioning cases. When arrival rates are lower than 0.35 ped/min per vehicle, the ridesharing methods perform similarly to their single capacity counterparts because current customers are being serviced before new customers arrive, effectively resulting in only one customer in each vehicle at a time. But at higher arrival rates, the ridesharing algorithm begins to utilize the excess vehicle capacity and new customers are inserted into vehicle schedules before previous customers have finished their ride. Through the use of a combined predictive positioning and ridesharing approach, the MOD system is able achieve a
Figure 7-4: Service times for unmanaged and predictive positioning methods, with and without ridesharing, over a range of customer arrival rates. The figure shows that predictive positioning reduces service times when arrival rates are low and ridesharing reduces service times when arrival rates are high. The service times are normalized by the direct time so as not to penalize service times for customers with longer routes. Lower service times are better. The arrival rates are normalized by the number of vehicles. The mean and standard deviation from 100 iterations are shown.

better customer QoS across all arrival rates, resulting in as much as a 29% reduction in service time compared to the single capacity unmanaged MOD strategy.

7.4.4 Ridesharing QoS Evaluation

The previous analysis assigned customers based only on minimizing service time, essentially imposing an assumed customer preference. However, this assumption can be wrong and customers may give poor ratings if the true customer preference lies elsewhere in other QoS metrics. To evaluate the rating performance of an MOD system, the simulated rating model is used to assign 5-star ratings to customers based on a set of customer preference weights. Six customer preference modes are analyzed, where the weight concentration parameters are 90% skewed towards either wait time, ride time, service time, the number of stops while onboard, ride distance,
or a combined weight between service time and ride distance. Five fleet management strategies are considered. First, a minimum vehicle distance strategy is considered, where assignments are not made based on customer ratings but rather based on the traditional minimum vehicle travel distance metric that would minimize fuel consumption. Next, three focused strategies based on ride time, service time, and wait time are considered, where each strategy chooses rating weights according to its focus. Finally, the presented random forest ratings model strategy is considered where the ground truth ratings function is not available but rather customer preference is learned from 5-star customer feedback ratings. The random forest model is implemented using [34], which is trained separately under each customer preference mode for 10 runs. To further test the ratings model, all ride distance metrics are removed from the random forest feature vector in order to see if performance can be learned using only non-corresponding, but correlated metrics. The arrival rate is fixed at 0.35 ped/min per vehicle so that the MOD fleet is operating under the ridesharing regime.

Figure 7-5 shows how the performance, in terms of average received rating, for each strategy depends on the underlying customer preferences. Section 7.4.4 summarizes the average customer rating over all 20 iterations. The results show that wait time, ride time, and service time each perform best when the customer preference mode matches. Additionally, the ride time metric performs best under the excess ride distance and combined customer preference modes because ride distance and ride time are correlated. However, each of the focused fleet management strategies performs relatively poorly under at least one customer preference mode, and the minimum distance strategy always performs poorly because customer preference is not considered. In contrast, the ratings model demonstrates robust performance across all customer preference modes. The ratings model is within 0.5% of the focused strategies under their respective modes, demonstrating that it was able to learn which customer metrics were important under each mode. The excess ride distance mode illustrates how the ratings model is able to learn customer preference using elements in its feature vector that are only correlated with the customer preference metric and not included directly. Finally, when considering the average performance over all tested customer preference
Table 7.1: MOD fleet management performance as measured by average customer rating.

<table>
<thead>
<tr>
<th>Allocation Strategy</th>
<th>Wait Time</th>
<th>Ride Time</th>
<th>Service Time</th>
<th># Stops</th>
<th>Ride Distance</th>
<th>Combined Metrics</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wait Time</td>
<td>4.69</td>
<td>4.42</td>
<td>4.09</td>
<td>3.64</td>
<td>3.51</td>
<td>3.82</td>
<td>4.03</td>
</tr>
<tr>
<td>Ride Time</td>
<td>4.42</td>
<td>4.97</td>
<td>4.43</td>
<td>4.5</td>
<td>4.97</td>
<td>4.7</td>
<td>4.66</td>
</tr>
<tr>
<td>Service Time</td>
<td>4.67</td>
<td>4.88</td>
<td>4.64</td>
<td>4.3</td>
<td>4.62</td>
<td>4.63</td>
<td>4.62</td>
</tr>
<tr>
<td>Vehicle Distance</td>
<td>4.53</td>
<td>4.7</td>
<td>4.22</td>
<td>3.91</td>
<td>4.10</td>
<td>4.17</td>
<td>4.37</td>
</tr>
<tr>
<td>Ratings Model</td>
<td>4.67</td>
<td>4.96</td>
<td>4.63</td>
<td>4.95</td>
<td>4.96</td>
<td>4.69</td>
<td>4.8</td>
</tr>
</tbody>
</table>

modes, the ratings model performed best.

7.5 Summary

This chapter introduces fleet management strategies for improving customer QoS in ride request MOD environments. A predictive positioning algorithm is presented that uses customer arrival rate information to position vehicles at key nodes in the MOD network graph to minimize the expected customer wait time. A ridesharing problem formulation is developed for assigning customers to vehicles according to a QoS cost function. A customer ratings model based on random forests is developed for use as a cost function for quantifying customer QoS based on ride metrics. An insertion-based ridesharing algorithm is developed to assign customers using an arbitrary cost function without constraining customer ride metrics. In a simulated campus setting, the predictive positioning method was shown to reduce customer service times by as much as 20% when customer arrival rates are low. To improve QoS as arrival rates increase, a ridesharing approach is presented that utilizes a customer QoS based cost function. A combined predictive positioning and ridesharing approach is shown to reduce customer service times by as much as 29%. The customer ratings model is evaluated as a means for learning customer preference through feedback in the form of a 5-star rating. The ratings approach is shown to provide the best overall MOD fleet management performance over a range of customer preferences.
Figure 7-5: Average customer rating for several fleet management strategies under customer preferences that are skewed towards either (a) wait time, (b) ride time, (c) service time, (d) the number of stops while onboard, (e) the excess ride distance between the received ride and a direct ride, or (f) a combined skew between service time and excess ride distance. The figures show that the performance of each strategy is dependent on the underlying customer preference, with the exception of the ratings strategy that performs well in all cases. Each figure shows the results of 20 runs. Performance is measured by the average customer rating where higher ratings are better.
Chapter 8

Conclusion

The focus of this thesis is on addressing challenges in the field of MOD systems with respect to estimating customer demand and performing automated fleet management. The thesis explores utilizing the capabilities of autonomous vehicles to improve customer QoS through traffic estimation, customer demand estimation, and fleet management methods. The methods were evaluated using hardware and simulation experiments based on the real-world MOD system that was developed for the MIT campus and presented in Chapter 3.

Chapter 4 presented a moving observer method that allows autonomous MOD vehicles to serve as mobile sensors for estimating network-wide traffic flows. For the MIT MOD shuttles, onboard camera and Lidar sensors are used to provide pedestrian trajectory data. The moving observer method decouples vehicle motion from pedestrian trajectory data to estimate pedestrian traffic arrival rates. Experimental testing demonstrated that the moving observer method achieves comparable arrival rate accuracy to that of stationary counters, allowing for network-wide sensing with mobile sensors in place of traditional infrastructure-based sensors.

Chapter 5 presented a customer demand estimation framework that incorporates real-time traffic sensing from the moving observer method to predict future customer arrivals. Estimates of customer arrivals are made using a two-parameter approach that combines traffic arrival rates with observed customer arrival fractions. A Bayesian framework utilizes the data available from the moving observer method to make
online, recursive updates to the demand model. An active sensing planner was introduced for utilizing the customer demand model to purposefully route vehicles in an MOD network graph in order to reduce uncertainty. Experimental testing shows that the novel demand modeling and active sensing approaches successfully improve customer arrival rate accuracy across the MOD system when compared against baseline approaches that focus either on waiting for customers or continual exploration.

Chapters 6 and 7 presented the formulation of automated MOD fleet management planners that utilize estimated customer demand in order to improve customer QoS in ride hailing, ride request, and ridesharing operating frameworks.

The ride hailing planner introduced in Chapter 6 addresses the challenge of utilizing uncertain customer demand in order to maximize the number of served customers. The planner addresses uncertainty in future customer arrival locations through a chance-constrained formulation. Experimental testing of the chance-constrained ride hailing planner demonstrates a significant improvement in the number of served customers in the MOD system over a baseline exploration approach.

The ride requesting predictive positioning planner presented in Chapter 7 addresses the challenge of reducing customer wait time for future customers. The planner uses customer arrival rate information to position vehicles at key nodes in the MOD network graph that minimize the expected customer wait time. Through experimental testing, the predictive positioning approach is shown to effectively reduce customer service times when compared against a baseline planner that does not reposition vehicles after serving customers.

The ridehailing planner presented in Chapter 7 addresses the challenge of improving customer QoS in complex ridehailing scenarios. The planner was formulated specifically for assigning customers to vehicles with respect to an arbitrary cost function, enabling the usage of a novel ratings-based QoS model. The ratings-based QoS model was formulated to utilize 5-star customer rating feedback in order to quantify the mapping between customer transit metrics and customer QoS. Experimental testing of the ridesharing planner showed that the utilized customer ratings model provided the best overall MOD fleet management performance over a range of unknown customer
preferences when compared against ridesharing planners that assume a pre-defined customer preference.

8.1 Future Work

There are several potential research extensions to the work in this thesis.

Unlike predictive models that perform parameter estimation from historical data, Chapter 5 focused on a proactive customer arrival model that incorporated real-time data. However, an ideal customer demand model would utilize both historical data for establishing long-term patterns as well as real-time data for determine fluctuations and perturbations [44]. Customer arrival rates are spatio-temporally varying on both long and short-term time scales, and are expected to contain underlying trends due to periodic activities such as daily commutes and schedules. An interesting extension for future work would be to build predictive demand models that determine these spatio-temporal patterns from historical data. One potential approach would be to use a Spectral Mixture Gaussian Process (SMGP) [60] which has been demonstrated to extract patterns from temporal data by learning spectral hyperparameters. A benefit to the SMGP approach would be the ability to provide customer arrival predictions with uncertainty measures. A challenge may be that of extracting patterns from data that is non-uniformly sampled and may contain large periods of time with no observations.

Another interesting future direction is that of physically implementing a completely driverless ride hailing MOD system. Most proposed autonomous MOD systems focus only on ride requesting, where customers can communicate with an autonomous shuttle through an app. The MIT MOD system in this thesis provided ride hailing capabilities but was dependent on a driver to identify and accommodate passengers hailing a ride. Future work could focus on detecting and accommodating hailed passengers using only onboard sensing and interfaces. A potential approach for detecting hailing customers would be to train a classifier on the output of camera-based vision processing algorithms that determine full pedestrian poses such as [15]. Once the passenger has
been identified, the vehicle would need to determine how to safely pull over, determine that the passenger is onboard, retrieve destination information from the passenger, and determine when the passenger has alighted. Each of these aspects have several human-machine interaction (HMI) components to investigate.

Finally, campus-based MOD systems such as the one developed in this thesis operate in pedestrian rich environments that are potentially quite different from the road environments for which autonomous vehicles are currently being developed. Figure 8-1 shows an example of an MIT MOD shuttle driving in the presence of pedestrian traffic. In the MIT MOD system, a driver performs actuation of the vehicle, allowing for safe navigation as the vehicle moves within pedestrian traffic. Future work could focus on safe autonomous driving for MOD shuttles embedded in pedestrian traffic. The challenge is that of avoiding the “freezing robot problem”, where naive algorithms cause vehicles to act overly conservative (to the point of freezing) in order to enforce safety constraints. Instead, algorithms will need to maintain a sufficient level of transportation quality (speed) for onboard customers while ensuring safety and acceptability among the pedestrians in the environment. The problem is addressed in [20], where the focus is on indoor ground robots which could be extended to accommodate the dynamics and intricacies of MOD shuttles.

Figure 8-1: MIT MOD vehicle in pedestrian traffic. The figure illustrates the level of pedestrian interaction required for campus-based MOD systems.
Bibliography


Appendix A

MOD Sensing Hardware and Software Design

This appendix provides a detailed description of the sensing suite for the MIT MOD vehicles that were designed for the thesis. The sensing hardware modifications made to the stock golf carts are discussed and the software that was developed and implemented to enable the advanced sensing capabilities is presented.

A.1 Hardware

Each of the three GEM vehicles is equipped with an identical sensing suite consisting of sensors commonly found in robotics and autonomous vehicle applications. The primary goal of the hardware is to provide a perception system capable of localizing the vehicle, detecting obstacles, and identifying the presence of passengers. This is accomplished through the use of camera, Lidar, and ultrasonic sensors, as well as the computing and power systems able to run those.

A.1.1 Cameras

The vehicles are equipped Logitech C920 cameras as pictured in Figure A-1a. The C920 has a 70.42° horizontal field of view (FOV) and 43.3° vertical FOV. It is capable
Figure A-1: Logitech C920 camera. The figures show the camera, mount, and mounting locations on the vehicle.

of outputting video at a 1080p resolution at 30 fps. A USB 2.0 interface provides power and communication and video can be compressed using mjpeg, which is important for streaming three cameras on a single USB bus.

Three C920 cameras are mounted on each vehicle, one in the middle and one on each corner. Figures A-1c and A-1d show the mounted position of the cameras. The cameras are mounted using custom 3D printed mounts as shown in Figure A-1b.

A.1.2 2D Lidar

The vehicles are equipped Sick LMS 151 2D Lidars as shown in Figure A-2a. The 151 provides a 270° horizontal FOV with a 0.5° resolution and 50 Hz update rate. The sensing range is between 0.5 m and 50 m. Power is provided through a 10.8-30 V interface, and communication is provided over Ethernet.

There are two 151 Lidars on each vehicle, one on the hood to detect pedestrians and one on the roof to sense buildings. Figures A-2c and A-2d show the mounted position of the 2D Lidars. The Lidars are mounted using manufacturer supplied
mounts and weather hoods as shown in Figure A-2b.

### A.1.3 3D Lidar

The vehicles are equipped with a Velodyne VLP-16 3D Lidar as shown in Figure A-3a. The VLP-16 provides a 360° horizontal FOV with a 0.2° resolution and 10 Hz update rate. The vertical FOV is split evenly with 15° FOV above center and 15° below, with a 2° resolution resulting in 16 beams. The sensing range is between 1 m and 100 m. Power is provided through a 9-18 V interface, and communication is provided over Ethernet.

There is one VLP-16 on each vehicle, mounted on the roof. Figures A-3c and A-3d show the mounted position of the Velodyne. The Velodyne is mounted using a tripod mount, as shown in Figure A-3b, allowing for the device to be easily removed when the vehicle is stored.
A.1.4 Seat Sensors

The vehicles are equipped with HRLV-MaxSonar-EZ4 ultrasonic sonar sensors above each passenger seat to detect the presence of passengers. The EZ4 is a narrow beam sonar with range between 1 mm and 5 m. Power and communication are provided through a TTL serial to USB adapter. The sensors are mounted on the interior ceiling of the vehicle using adhesive, as shown in Figure A-4.

A.1.5 Computing

Each vehicle is equipped with two computers, a Gigabyte Brix PC for processing data and a Dell Laptop PC for visualization.

Most processing is performed using the Gigabyte Brix GB-BXi7G3-760, which is a small form factor desktop PC. The computer features an i7-4710HQ processor,
Figure A-4: Seat sensor. An HRLV-MaxSonar-EZ4 sonar serves as the passenger occupancy sensor and is mounted above every passenger seat.

16GB DDR3 RAM, a 256GB SSD, and an nVidia GeForce GTX 760 GPU. The main advantage to this computer is the discrete GPU that allows for processing tasks that require nVidia specific hardware such as CUDA. The Brix resides in the communications box mounted on the rear of the vehicle shown in Figure A-5. The Brix itself is shown in Figure A-8.

A Dell Inspiron I5547 laptop is used to interface with the Brix and visualize data. The I5547 features an i7-4510U processor with 8GB RAM. The laptop is placed on the dash of the vehicle to allow for easy access from either the driver or a researcher, as shown in Figure A-6. The laptop is powered from an accessory power port on the vehicle and communicates with the Brix over a Gigabit Ethernet connection.
Figure A-5: Computing and battery. The computing box and battery box are mounted on a bed in the rear of the vehicle.

Figure A-6: Laptop location. The laptop is placed on the dash of the vehicle to provide a display to the driver. The laptop can also be removed and used by a researcher in the front passenger seat.
A.1.6 Power and Communication

Each component communicates with the Gigabyte Brix through either USB or Ethernet. The components and computer are all powered from a single high-capacity 12V 125Ah lead acid battery, mounted on the rear of vehicle as shown in Figure A-5. To meet the voltage requirement of each component, various power converters are used. Figure A-7 shows the wiring diagram for both the power and data of each component. Figure A-8 shows the hubs and power converters mounted in the computing box at the rear of the vehicle.
Figure A-8: Hardware wiring components. The figures show the layout and labels for the various wiring components in the box at the rear of the vehicle.
Figure A-9: Software overview diagram. The diagram shows an overview of the software used on the vehicles, where boxes indicate processing nodes and arrows indicate passed data types. The software is split into four main components: localization (blue), clustering (orange), pedestrian detection (green), and pointcloud logging (purple). The pointclouds from the velodyne are logged to establish ground truth for pedestrian tracking algorithm development.

A.2 Software

The vehicles run a software suite capable of interfacing with the hardware and implementing perception based systems. The ultimate goal for the software suite is to provide real-time localization and pedestrian tracking on the vehicles. Figure A-9 shows a diagram overview of the software stack.

A.2.1 Robot Operating System (ROS)

The Robot Operating System (ROS) [53] provides a software base that allows components to interface with one another. ROS is a middleware service that provides message-passing through independent processes. Each process is run as a node that publishes and subscribes to messages to and from other nodes. These processes include lower-level hardware drivers as well as higher-level perception algorithms. ROS
software can be provided either through an open-source package or through custom code. ROS also includes the RVIZ package [33] that provides visualization of the outputs from many of the other packages.

ROS is run on each of the computers in the vehicles on top of the Ubuntu 14.04 operating system. Each ROS package is configured through an XML-formatted launch file with specific parameters. Details for the ROS packages and launch files used in the MIT MOD system are provided below.

### A.2.2 Hardware Drivers

ROS provides open source packages capable of interfacing with many robotic sensors. The following provides an overview of the packages used for interfacing with vehicle’s camera and Lidar sensors.

#### Camera Drivers

The data from the cameras is retrieved through the `usb_cam` ROS package [52]. The package reads the raw data from the camera device and publishes an Image topic, as shown in Figure A-10a. Example code that is used to launch the left camera is provided:

```
<launch>
  <node name="usb_cam_left" pkg="usb_cam" type="usb_cam_node"
    output="screen">
    <param name="video_device" value="/dev/cameraLeft"/>
    <param name="image_width" value="640"/>
    <param name="image_height" value="360"/>
    <param name="pixel_format" value="mjpeg"/>
    <param name="camera_frame_id" value="usb_cam_left"/>
    <param name="framerate" value="30"/>
    <param name="io_method" value="mmap"/>
    <param name="camera_info_url" value="file://$(find
```
2D Lidar Driver

The data from the 2D Lidars is retrieved through the LMS1xx ROS package [10]. The package reads the raw data from the Lidar device and publishes a Laserscan topic, as shown in Figure A-10b. Example code that is used to launch the upper Lidar is provided:

```xml
<launch>
  <arg name="upper_ip" default="192.168.0.26" />
  <node pkg="lms1xx" name="lms151_upper" type="LMS1xx_node" output="screen">
    <param name="host" value="$(arg-upper_ip)" />
    <param name="frame_id" value="laser_upper" />
    <param name="topic_name" value="scan_upper" />
  </node>
</launch>
```

3D Lidar Driver

The data from the 3D Lidar is retrieved through the Velodyne ROS package [46]. The package reads the raw data from the Velodyne device and publishes a PointCloud2 topic, as shown in Figure A-10c. Example code that is used to launch the VLP-16 is provided:

```xml
<launch>
  <arg name="pcap" default="" />
</launch>
```
A.2.3 Localization

Localization is the process of determining the precise position and orientation of a vehicle within a global frame. In dense urban environments such as MIT campus, GPS sensors provide poor estimates of position. One option is to track the position of the vehicle over time using information about relative motion of the vehicle, known as odometry. Traditionally, this is performed using encoders that measure the rotation of the wheels of the vehicle in order to estimate relative motion. However, odometry alone is typically not able to provide a robust localization solution because any errors between measurements will accumulate to large errors in position estimates, known as drift. To correct the drift, MOD vehicles are localized to a known map using the upper Lidar. To accomplish this, a simultaneous localization and mapping (SLAM) package is used to first obtain a map from recorded laserscans. Then, a localization
package is used to match laserscans to the obtained map in order to precisely localize the vehicle.

**Odometry**

Rather than use wheel encoders, odometry for the vehicles is provided through more precise laserscan matching. The scan matching is performed through the Hector SLAM ROS Package [39] that is based on the work in [38]. The package subscribes to the Laserscan topic published by the LMSxx package for the upper Lidar and publishes a transformation between the vehicle and an odometry frame. Example code that is used to launch the Hector SLAM node is provided:

```xml
<launch>
  <node pkg="hector_mapping" type="hector_mapping" name="hector_mapping" >
  
  </node>

<!— Laser Parameters —>
<param name="laser_max_dist" value="50"/>

<!— Frame names —>
<param name="map_frame" value="odom" />
<param name="base_frame" value="base_link" />
<param name="odom_frame" value="base_link" />

<!— Tf use —>
<param name="use_tf_scan_transformation" value="true"/>
<param name="use_tf_pose_start_estimate" value="false"/>
<param name="pub_map_odom_transform" value="false"/>

<!— Map size / start point —>
<param name="map_resolution" value="0.1"/>
<param name="map_size" value="4000"/>
```
SLAM

A map is generated using the SLAM Karto package [30]. The package subscribes to the odometry transformation from the Hector SLAM package and the Laserscan
from the upper Lidar and produces an occupancy grid map. SLAM Karto uses a graph SLAM approach capable of making loop closures based on laserscan matching. To generate a map, a vehicle is manually driven while recording a bag file of all of the observed laserscans. The file is then processed using SLAM Karto to generate an occupancy grid map. The occupancy grid map generated for the the MIT MOD operating region is shown in Figure A-11. Example code that is used to launch the SLAM Karto node is provided:

```
<launch>
  <node pkg="slam_karto" type="slam_karto" name="slam_karto"
        output="screen">
    <remap from="scan" to="scan_upper"/>
    <param name="map_update_interval" value="190" />
    <rosparam command="load" file="$(find_slam_karto)/config/
               mapper_params.yaml" />
  </node>
</launch>
```

where the contents of mapper_params.yaml are:

```
resolution: 0.1
odom_frame: "odom"

# General Parameters
use_scan_matching: true
use_scan_barycenter: true
minimum_travel_distance: 0.5
minimum_travel_heading: 0.5
minimum_solver_distance: 20
scan_buffer_size: 1000000
scan_buffer_maximum_scan_distance: 10000.0
link_match_minimum_response_fine: 0.8
```
link_scan_maximum_distance: 100.0

do_loop_closing: true

loop_match_minimum_chain_size: 1

loop_match_maximum_variance_coarse: 0.8

loop_match_minimum_response_coarse: 0.001

loop_match_minimum_response_fine: 0.1

# Correlation Parameters – Correlation Parameters

correlation_search_space_dimension: 1

correlation_search_space_resolution: 0.05

correlation_search_space_smear_deviation: 0.1

# Correlation Parameters – Loop Closure Parameters

loop_search_space_dimension: 75.0

loop_search_space_resolution: 0.75

loop_search_space_smear_deviation: 0.375

loop_search_maximum_distance: 100

# Scan Matcher Parameters

distance_variance_penalty: 0.9

angle_variance_penalty: 0.9

fine_search_angle_offset: 0.00349

coarse_search_angle_offset: 0.349

coarse_angle_resolution: 0.0349

minimum_angle_penalty: 0.9

minimum_distance_penalty: 0.5

use_response_expansion: false
Localization

Localization is provided by the Adaptive Monte Carlo Localization (AMCL) package [29] that is based on the work in [57]. The package reads in odometry information provided by Hector SLAM, laserscans from the upper Lidar, and the occupancy grid map and produces a pose message with the position and orientation relative to the origin of the map. AMCL uses a particle filter that maintains pose estimate particles that are probabilistically weighted based on the alignment of the laserscan measurement projected onto the map. An example of visualization of a laserscan, map, and pose estimate is shown in Figure A-12. Example code that is used to launch the AMCL node is provided:

```xml
<launch>
  <node pkg="amcl" type="amcl" name="amcl">
    <param name="update_min_d" value="0.1"/>
    <param name="update_min_a" value="0.1"/>
    <param name="recovery_alpha_fast" value="0"/>
    <param name="recovery_alpha_slow" value="0"/>
    <param name="resample_interval" value="5"/>
    <param name="min_particles" value="200"/>
    <param name="max_particles" value="500"/>
    <param name="laser_max_range" value="50"/>
    <param name="laser_min_range" value="2"/>
    <param name="laser_max_beams" value="541"/>
    <param name="laser_max_beams" value="54"/>
    <param name="odom_frame_id" value="odom"/>
    <param name="base_frame_id" value="base_link"/>
    <param name="initial_pose_x" value="-1000.0"/>
    <param name="initial_pose_y" value="-1000.0"/>
    <param name="initial_pose_a" value="0.0"/>
    <param name="initial_cov_xx" value="1"/>
  </node>
</launch>
```
A.2.4 Pedestrian Tracking

Pedestrian tracking extracts useful pedestrian trajectory data from the camera and Lidar sensors. First, laserscans are clustered together to provide obstacle trajectories. Simultaneously, pedestrians in the camera images are identified with bounding boxes. Then, obstacle clusters with corresponding bounding boxes are classified as pedestrians, resulting in pedestrian trajectories.

Obstacle Clustering

Obstacle clustering is provided from a custom package created using the dynamic means algorithm [14]. The approach takes in laserscans from the lower Lidar and outputs clusters as 2D shapes with a defined centroid position. The dynamic means algorithm is computationally expensive so pre-processing of the laserscans is performed. First, the Laserscan is converted into a PointCloud2 datatype using the scan_to_cloud_converter package [24]. Example code for this conversion node is provided:

```xml
<launch>
  <node pkg="scan_to_cloud_converter" type="scan_to_cloud_converter_node" name="lidar_to_cloud" output="screen"/>
</launch>
```
Next, any portions of the scan that align with the static data in the occupancy grid map are removed, as those are not expected to correspond to pedestrians. This is accomplished through a custom map_filter_node that uses the pose of the vehicle provided by AMCL to align the scan’s pointcloud with the occupancy grid map, and the points that correspond with the map are removed. Note that the map_filter_node is part of a custom pcl_clustering package that is not openly available; to request access to this code, please contact the thesis author. Example code for launching the map filter is provided:

```xml
<launch>
  <node pkg="pcl_clustering" type="map_filter_node" name="map_filter" output="screen">
    <remap from="~input" to="scan_lower_cloud"/>
    <remap from="~output" to="~filtered_cloud"/>
    <remap from="~map" to="map_lower"/>
    <param name="~map_frame" value="map"/>
    <param name="~publish_map_cloud" value="false"/>
    <param name="~passthough/flag" value="false"/>
    <param name="~diff_filter/flag" value="true"/>
    <param name="~diff_filter/octree_res" value="0.5"/>
    <param name="~downsample/flag" value="false"/>
    <param name="~downsample/leaf_size" value="0.15"/>
    <param name="~noise_remove/flag" value="true"/>
    <param name="~noise_remove/radius" value="0.33"/>
    <param name="~noise_remove/min_neighbor" value="3"/>
  </node>
</launch>
```
After filtering, the remaining pointcloud is then clustered using the dynamic means cluster_node. The output is a custom cluster message containing an ID and centroid pose vector for each obstacle observed by the lower Lidar. Example code for launching the cluster_node is provided:

```xml
<launch>
  <node pkg="pcl_clustering" type="cluster_node" name="cluster"
>
    <remap from="~input" to="map_filter/filtered_cloud"/>
    <param name="euclidean/flag" value="false"/>
    <param name="region_growing/flag" value="false"/>
    <param name="dynmeans/flag" value="true"/>
    <param name="dynmeans/lambda" value="1.0"/>
    <param name="dynmeans/t_q" value="25.0"/>
    <param name="dynmeans/k_tau" value="1.01"/>
    <param name="dynmeans/n_restart" value="3"/>
    <param name="dynmeans/min_size" value="3"/>
    <param name="dynmeans/max_size" value="500"/>
    <param name="dynmeans/use_alpha_beta" value="true"/>
    <param name="dynmeans/alpha" value="1.0"/>
    <param name="dynmeans/beta" value="0.025"/>
  </node>
</launch>
```

**Pedestrian Detection**

The pedestrian detection algorithm detects the existence and location of pedestrians in the camera image and provides a representation of their location in a global frame. First, the pedestrians are classified in the image frame using the multiple_cameras_very_fast_detector node that subscribes to the Image topics published
by the left, front, and right camera nodes, identifies bounding boxes for each pedestrian, and publishes the left, right, and center positions of the bounding boxes in each image. This node provides a ROS wrapper for the open source VeryFast detector [11] that performs the detections. Note that the node is part of a custom human_detector package that is not openly available; to request access to this code, please contact the thesis author. Example code for launching the detector is provided:

```xml
<launch>
  <node name="very_fast" pkg="human_detector" type="multiple_cameras_very_fast_detector">
    <param name="score_thresh" value="0.05"/>
  </node>
</launch>
```

The bounding box information from the detector is specific to the frame of each image. In order to pair this information with clusters, this information needs to be projected into a common global frame. Because the clusters exist in the 3D map frame, projecting data from the 2D image frame will result in an under-defined transformation. Essentially, the bounding box can place an object with respect to the horizontal and vertical FOV of the camera, but not depth. To account for this depth uncertainty, the bounding boxes are converted into vectors that project outward in the direction of the camera. A custom world_vector_converter node is used to transform the bounding boxes in each image into vectors in the map frame. Example code for launching the converter is provided:

```xml
<launch>
  <node name="vector_converter_middle" pkg="human_detector" type="world_vector_converter.py" output="screen">
    <param name="score_thresh" value="$(arg_ped_detection_thresh)"/>
    <remap from="camera/rgb/camera_info" to="usb_cam_middle/camera_info"/>
  </node>
</launch>
```
Cluster Classification

To obtain the trajectories of only pedestrians, the obstacle trajectories that correspond to pedestrians are determined. A custom cluster_classifier node subscribes to the clusters provided by the cluster node and the vectors provided by the converter node, performs a probabilistic classification, and outputs the likelihood of the cluster being a pedestrian. The cluster classification approach uses angular distance between clusters and vectors to assign a pedestrian likelihood to each cluster; more details for the algorithm are presented [32]. If the likelihood of a cluster exceeds a given threshold, then the cluster can be classified as a pedestrian. Example code for launching the cluster classifier is provided:

```
<launch>
  <node name="cluster_classifier" pkg="human_detector" type="cluster_classifier.py" output="screen">
    <param name="timer_period" value="0.03" />
    <param name="cam_radius" value="20" />
  </node>
</launch>
```

A.2.5 Data collection procedure

The above presented a generalized overview for performing vehicle localization and pedestrian tracking using the sensors on the vehicle. Here the specifics for running the code on the MIT MOD vehicles are presented. These instructions assume access to the specific computing hardware on the MIT MOD vehicles and do not generalize
to other systems.

**Recording Pedestrian Trajectories**

To record pedestrian trajectories, all of the localization and pedestrian tracking processes are run on the Brix computer, which is interfaced through the laptop. First, from a terminal on the laptop, connect to the Brix computer via ssh:

```
ssh swarm@$VEHICLE_ID
```

Then, in the ssh terminal, launch ped_saver:

```
roslaunch ford_ros ped_saver.launch
```

This will begin the localization and pedestrian tracking process.

**Visualization**

After the processes are running on the Brix, the data can be visualized on the laptop using RVIZ. To start the visualization, in a local terminal window, launch rviz:

```
roslaunch ford_ros gem_rviz.launch
```
Figure A-10: Sensor data comparison. The images show visualizations of the same scene with either Image, Laserscan, or Pointcloud data. An image is an array of RGB pixel values, a Laserscan is a vector of angular range and intensity measurements, and a Pointcloud is an array of x, y, z points with intensity values.
Figure A-11: Occupancy grid map for MIT campus. The map was generated from laserscan data using the SLAM Karto ROS package. The grid map is an image file with black pixels indicating occupied space, white pixels indicating free space, and gray pixels indicating unclassified space.

Figure A-12: Localization with AMCL. The background shows a zoomed in portion of the known occupancy grid map. The red arrows indicate the poses of the particles that represent the belief in the vehicle's position. The reported pose is the weighted average of the particles. The yellow and green line shows the laserscan projected onto the map based on the reported pose. The alignment between the laserscan and map indicates that the vehicle is well localized.
Appendix B

Marginalization of Arrival Probability

Section 5.2 presents a two-parameter model for estimating customer arrivals based on pedestrian arrival rate parameters, $\lambda_n$, that are modeled using Gamma distributions with hyperparameters, $\alpha_n$ and $\beta_n$, and customer fractions, $p_n$, that are modeled using Beta distributions with hyperparameters $a_n$ and $b_n$. The probability of the predicted number of customer arrivals, $\hat{c}_n$, over a time period of $t_{pred}$ is determined through marginalization of the pedestrian arrival rate and customer fraction parameters. The derivation of the analytical expression for the probability of $\hat{c}_n$ is provided as,

$$P(\hat{c}_n; t_{pred}, \alpha_n, \beta_n, a_n, b_n)$$

$$= \int_0^1 \int_0^\infty P(\hat{c}_n; \lambda_n, p_n, t_{pred}) \cdot P(\lambda_n; \alpha_n, \beta_n) \cdot P(p_n; a_n, b_n) \, d\lambda_n dp_n.$$  

$$= \int_0^1 \int_0^\infty \text{Pois}(\hat{c}_n; p_n \lambda_n t_{pred}) \cdot \text{Gamma}(\lambda_n; \alpha_n, \beta_n) \cdot \text{Beta}(p_n; a_n, b_n) \, d\lambda_n dp_n$$

$$= \int_0^1 \text{Beta}(p_n; a_n, b_n) \left[ \int_0^\infty \text{Pois}(\hat{c}_n; p_n \lambda_n t_{pred}) \cdot \text{Gamma}(\lambda_n; \alpha_n, \beta_n) \right] \, d\lambda_n dp_n$$

$$= \int_0^1 \text{Beta}(p_n; a_n, b_n) \int_0^\infty \frac{e^{-p_n \lambda_n t_{pred}} (p_n \lambda_n t_{pred})^{\hat{c}_n}}{\hat{c}_n!} \frac{\beta_n^{\alpha_n} \lambda_n^{\alpha_n-1} e^{-\lambda_n \beta_n}}{\Gamma(\alpha_n)} \, d\lambda_n dp_n$$

$$= \int_0^1 \text{Beta}(p_n; a_n, b_n) \left[ \int_0^\infty e^{-p_n \lambda_n t_{pred}} \lambda_n^{\alpha_n + \hat{c}_n} \, d\lambda_n \right] \frac{\beta_n^{\alpha_n} (p_n t_{pred})^{\hat{c}_n}}{\hat{c}_n! \Gamma(\alpha_n)}$$

$$= \int_0^1 \text{Beta}(p_n; a_n, b_n) \frac{\beta_n^{\alpha_n} (p_n t_{pred})^{\hat{c}_n}}{\hat{c}_n! \Gamma(\alpha_n)} \left[ \int_0^\infty e^{-p_n (\beta_n + p_n t_{pred}) \lambda_n} (\alpha_n + \hat{c}_n)^{-1} \, d\lambda_n \right] dp_n$$

$$= \int_0^1 \text{Beta}(p_n; a_n, b_n) \frac{\beta_n^{\alpha_n} (p_n t_{pred})^{\hat{c}_n}}{\hat{c}_n! \Gamma(\alpha_n)} \frac{\Gamma(\alpha_n + \hat{c}_n)}{(\beta_n + p_n t_{pred})^{\alpha_n + \hat{c}_n}} \, dp_n$$

$$= \int_0^1 \frac{p_n^{\alpha_n-1} (1 - p_n)^{b_n-1}}{B(a_n, b_n)} \frac{\beta_n^{\alpha_n} (p_n t_{pred})^{\hat{c}_n}}{\hat{c}_n! \Gamma(\alpha_n)} \left[ \int_0^\infty e^{-p_n (\beta_n + p_n t_{pred}) \lambda_n} (\alpha_n + \hat{c}_n)^{-1} \, d\lambda_n \right] \, dp_n$$
\[
\begin{align*}
\beta_n^\alpha t_{\text{pred}}^\beta \Gamma(\alpha_n + \hat{\alpha}_n) & \cdot \frac{\int_0^1 p_n^{\alpha_n + \hat{\alpha}_n - 1}(1 - p_n)^{b_n - 1}}{B(a_n, b_n) \hat{\alpha}_n ! \Gamma(\alpha_n)} \cdot \int_0^1 p_n^{\alpha_n + \hat{\alpha}_n - 1}(1 - p_n)^{b_n - 1} d\rho_n \\
= \beta_n^\alpha t_{\text{pred}}^\beta \Gamma(\alpha_n + \hat{\alpha}_n) & \cdot \frac{\beta_n^{-\alpha_n - \hat{\alpha}_n} \Gamma(b_n) \Gamma(\alpha_n + \hat{\alpha}_n)}{B(a_n, b_n) \hat{\alpha}_n ! \Gamma(\alpha_n)} \\
\cdot 2\tilde{F}_1 \left( a_n + \hat{\alpha}_n, \alpha_n + \hat{\alpha}_n, a_n + b_n + \hat{\alpha}_n, -\frac{t_{\text{pred}}}{\beta_n} \right) \\
= \frac{\Gamma(a_n + b_n)}{\Gamma(a_n) \Gamma(b_n)} \cdot \frac{t_{\text{pred}}^{\hat{\alpha}_n} \Gamma(\alpha_n + \hat{\alpha}_n) \Gamma(b_n) \Gamma(\alpha_n + \hat{\alpha}_n)}{\beta_n^{\hat{\alpha}_n} \hat{\alpha}_n ! \Gamma(\alpha_n)} \\
\cdot 2\tilde{F}_1 \left( a_n + \hat{\alpha}_n, \alpha_n + \hat{\alpha}_n, a_n + b_n + \hat{\alpha}_n, -\frac{t_{\text{pred}}}{\beta_n} \right) \\
= \frac{\Gamma(\alpha_n + \hat{\alpha}_n) \Gamma(a_n + \hat{\alpha}_n) \Gamma(a_n + b_n)}{\hat{\alpha}_n ! \Gamma(\alpha_n) \Gamma(a_n + b_n + \hat{\alpha}_n)} \cdot \left( \frac{t_{\text{pred}}}{\beta_n} \right)^{\hat{\alpha}_n} \\
\cdot 2\tilde{F}_1 \left( a_n + \hat{\alpha}_n, \alpha_n + \hat{\alpha}_n, a_n + b_n + \hat{\alpha}_n, -\frac{t_{\text{pred}}}{\beta_n} \right).
\end{align*}
\]

\(\Gamma(\cdot)\) represents the Gamma function, \(B(\cdot, \cdot)\) represents the Beta function, \(2F_1(\cdot, \cdot, \cdot, \cdot)\) represents the hypergeometric function, and \(2\tilde{F}_1(\cdot, \cdot, \cdot, \cdot)\) represents the regularized hypergeometric function. The Gamma and Beta functions are related by

\[B(x, y) = \frac{\Gamma(x) \Gamma(y)}{\Gamma(x + y)}.\]

The hypergeometric and regularized hypergeometric functions are related by

\[2\tilde{F}_1(w, x, y, z) = \frac{2F_1(w, x, y, z)}{\Gamma(y)}.\]