All-in-One Urban Mobility Mapping Application with Optional Routing Capabilities

Rebekah Thompson
College of Engineering and Computer Science
University of Tennessee at Chattanooga
Chattanooga, Tennessee
dys548@mocs.utc.edu

Jose Stovall
College of Engineering and Computer Science
University of Tennessee at Chattanooga
Chattanooga, Tennessee
ggy323@mocs.utc.edu

Daniel Velasquez
College of Engineering and Computer Science
University of Tennessee at Chattanooga
Chattanooga, Tennessee
dyf914@mocs.utc.edu

Viswa Sri Rupa Anne
Aerospace, Transportation and Advanced Systems Laboratory
Georgia Tech Research Institute
Atlanta, GA
vanne3@gatech.edu

Alex V. Samoylov
Aerospace, Transportation and Advanced Systems Laboratory
Georgia Tech Research Institute
Atlanta, GA
alex.samoylov@gtri.gatech.edu

Mina Sartipi
College of Engineering and Computer Science
University of Tennessee at Chattanooga
Chattanooga, Tennessee
mina-sartipi@utc.edu

Abstract— To create safer and less congested traffic operating environments researchers at the University of Tennessee at Chattanooga (UTC) and the Georgia Tech Research Institute (GTRI) have fostered a vision of cooperative sensing and cooperative mobility. This vision is realized in a mobile application that combines visual data extracted from cameras on roadway infrastructure with a user's coordinates via a GPS-enabled device to create a visual representation of the driving or walking environment surrounding the application user. By merging the concepts of computer vision, object detection, and mono-vision image depth calculation, this application is able to gather absolute Global Positioning System (GPS) coordinates from a user's mobile device and combine them with relative GPS coordinates determined by the infrastructure cameras and determine the position of vehicles and pedestrians without the knowledge of their absolute GPS coordinates. The joined data is then used by an iOS mobile application to display a map showing the location of other entities such as vehicles, pedestrians, and obstacles creating a real-time visual representation of the surrounding area prior to the area appearing in the user's visual perspective. Furthermore, a feature was implemented to display routing by using the results of a traffic scenario that was analyzed by rerouting algorithms in a simulated environment. By displaying where proximal entities are concentrated and showing recommended optional routes, users have the ability to be more informed and aware when making traffic decisions helping ensure a higher level of overall safety on our roadways. This vision would not be possible without high speed gigabit network infrastructure installed in Chattanooga, Tennessee and UTC's wireless testbed, which was used to test many functions of this application. This network was required to reduce the latency of the massive amount of data generated by the infrastructure and vehicles that utilize the testbed; having results from this data come back in real-time is a critical component.

Keywords—Big Data, Computer Vision, Mobile Mapping, Mobile Sensor Networks, Smart Cities, Traffic Simulation, Urban Mobility, V2X, Wireless Communication

I. INTRODUCTION

To create safer, less congested, and more efficient mobility operating environment, vehicles need to heighten the awareness of changing conditions either next to the vehicle or at some distance on the route. Typical vehicles capable of sensing surrounding environments rely on sensors that are able to sense within the line of sight of the sensor. If the line of sight is broken the vehicle might lose awareness of what happens beyond the sensing field of view. Even a bus that stopped on the side of the road can significantly impede the sensory data collection, not to mention presence of buildings or road curvature or changing road grade. To eliminate this problem, we would like to introduce the concepts of cooperative sensing and cooperative mobility, which use streams of data from multiple sensors to work together to provide cohesive information for the application. The data collected from cooperative sensing and cooperative mobility have the potential to reduce the number of accidents by collecting information about current road conditions and using Inter-Vehicle Communications (IVC), Vehicle-to-Infrastructure Communications (V2I), and Infrastructure-to-Vehicle Communications (I2V) to propagate the information throughout the fleet. In addition, image and point-cloud data are extremely processing intensive; thus, if we can augment data sensed and processed by the vehicle with data already processed elsewhere, we can improve processing efficiency and create smart reuse of the data to navigate the road network.

To achieve this objective, we employ a threefold approach: 1) gathering road conditions data from multiple sources: sensors installed on the vehicles and roadside sensor units; 2) using centralized traffic management system to simulate optimal routes, and 3) creating an application to display route navigation and route condition information to the users.

Current navigation applications primarily focus on data that suits the need of the consumer on a personal level such as traffic

conditions, quickest route, or points of interest. There are other real-time dynamic changes on the route such as accidents, road construction, or debris on the road that might affect the driver experience. Although WAZE [10] pools data from its application users and reports it to a certain extent, it only uses GPS coordinates emitted from mobile devices to plot users and locations of interest. In addition, socially pooled data may not be reported in real time and may not be accurate. The approach described hereafter will provide systematic real-time data collection from the vehicles or roadside sensor units to more fully and accurately represent the route conditions and display it to users.

This approach consists of an application which utilizes positioning from multiple sensors, including GPS-enabled mobile devices and wireless cameras, to anonymously track multiple subjects of interest on a unified mapping interface. The next two sub-sections will outline our contributions and analyze related works pertaining to the topic of this paper. Section II will explain the methods used during this project which include: 1) gaining the information to map the absolute GPS coordinate via mobile devices (latitude and longitudes reported directly from the mobile device's location feature); 2) the relative GPS coordinate (an approximate location derived from known locations of other objects or points of interest) via object detection and infrastructure communication; 3) steps taken to achieve rerouting algorithms based on obstacles hindering the driver on their typical route. Section III will discuss results related to each of the items explained in the methods section. Section IV will conclude this paper with a summary of findings from this research and future work to be explored.

A. Our Contributions

Our first contribution is providing additional information in relation to pedestrian, cyclist, and vehicle localization through both GPS-enabled devices, comparable to commercialized applications, and through the use of computer vision, which does not need to continuously gather GPS data to map individuals. The ability to be able to track objects through computer vision from static locations will also add functionality in the case of GPS being unavailable in other locations.

Our second contribution is creating routing based on the real-time data collected on the road network from other vehicles and roadside units and simulating future conditions based on the data to assign most optimal route. In addition, we implemented re-routing function that can re-route vehicles based on road blockages gathered by real-time, computer vision data.

These two contributions work together to create an all-in-one detection and routing mobile application that displays a holistic view of the user's environment, including pedestrian locations, which gives the user a better understanding of their surrounding area beyond their physical field of view. The data used in this application can also be utilized without a visual component and can be used directly for autonomous vehicle decision and map-making and therefore expands the benefits of the data gathering and usage of this application. This would allow the knowledge of the vehicle's environment to be more robust and offers the vehicle more time for predictive analysis in relation to on-road decision making.

To maximize the process of gathering the real-time data used in the application, Chattanooga's fiber optic Internet was used. The fiber optic network is continuously used for the rapid transfer between the clients (cameras and mobile devices) and the server where the data is being analyzed. Given that the network can transfer up to 10 Gbps, this speed and bandwidth provided is a potential source for further exploration in to real-time Big Data applications for various fields. This amount of throughput allows data to be transferred efficiently. This data can be analyzed for several statistics including times and amounts of high traffic (both pedestrian and vehicular), amount of jay-walking, amount of traffic lights run, and countless others.

B. Related Works

Many previous explorations of this topic were focused on single data sources, such as GPS and Global System for Mobile Communications (GSM) coordinates [2]. In recent years, the use of many heterogeneous sources at once has been adopted in place of single data sources. One such example is Shotgun Reading via Wi-Fi, a process inspired by DNA sequencing that outputs a directed, weighted graph of the individual devices discovered after a 'burst' read of a wide area [3]. Mobile Sensor Networks have also been implemented, involving the use of a fusion algorithm to measure a scalar field and construct its map to discern occupants of the surveyed area [4]. Another viable option is multi-sensor data-fusion or fusing recorded data from multiple sensors together with pre-existing knowledge [2]. Traffic management systems typically consist of an advanced traffic information system which transmits real-time data to the user apart from routing the user based on real-time conditions. Use of simulation software to replicate the real-world scenario to manage and advise the vehicles to change speeds and warn them about collisions was studied [11].

II. METHODS

This section is divided based on the components that were used for each contribution. Sections A through C are related to the first contribution that maps local objects using GPS-enabled devices and infrastructure-based cameras using computer vision to detect and localize objects using static GPS coordinates instead of continuously changing coordinates that are normally seen through mobile devices. Sections D and E will highlight the second contribution, which relates to real-time routing and rerouting based on detected objects using traffic simulation model.

A. Tracking Objects via Computer Vision

This method used a single camera to capture and send an image to an off-site server, the server then analyzed the image for any useful information in relation to the image training set the convolutional neural network (CNN) used and would then send that information to a database to be used for adding the relative location of an object to a map. Each of these steps, including the mapped icons, were updated in real-time. This algorithm does not rely on the use of GPS services directly; instead it utilizes a trilateration algorithm which requires three known points accompanied by three known distances based on designated reference points. The following steps are taken to obtain the needed reference points:

1) Object Detection

For object detection, a CNN called You Only Look Once (YOLO) was used. YOLO version 2 can detect 80 different class types using the pre-trained model [5]. These classes include people, bicycles, cars, motorbikes, airplanes, and 75 other common objects. Using YOLO, multiple common roadway objects can be detected and tracked, further increasing driver awareness. YOLO was trained on full images rather than separate sub-sections of a single image, which increased its overall detection performance [5]. The YOLO network can analyze real-time video at approximately 45 frames per second (fps) with a latency of less than 25 milliseconds using the Titan X graphical processing unit (GPU) [5]. Another advantage to using YOLO is that the CNN looks at the training and test images globally rather than sub-sections, and it can therefore understand the features of the entire image compared to sub-sections of the image (i.e. sliding windows) [5]. This approach assists in less false positives when analyzing an image and ultimately increases its overall accuracy. This object detection process is also used in the calibration step that will be explained in the next sub-section.

2) Camera Calibration Process

The first step to obtaining the three known location points used to find the geographical coordinates of the detected object is to calibrate the camera. During the calibration procedure, absolute latitude and longitude coordinates are used together to determine the geolocation of the camera. Once the geolocation of the camera has been determined, two reference points on the left and right side of the camera's field of view are hardcoded as GPS coordinates into the calibration method to give a reference to compare with the camera and detected object to calculate and approximate location of the detected object.

During the calibration process, a person must stand a known distance away in the camera frame. At this point in the calibration process, YOLO gives an approximate pixel-height of the detected person where "x" is width and "y" is height. If the number of pixels in the detected object's bounding box increases, we can assume the object is moving closer to the camera, and if the pixel numbers decrease, we can assume the object is moving away from the camera based on the bounding box pixel area from the original calibration (see Figures 1 and 2).

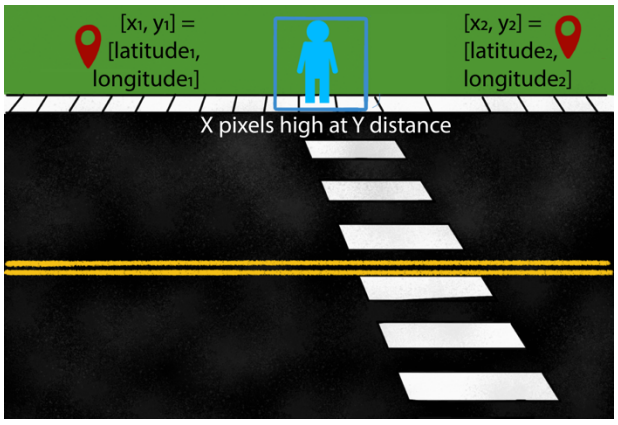

Figure 1. Detected object is farther away from camera and has smaller pixel area within bounding box.

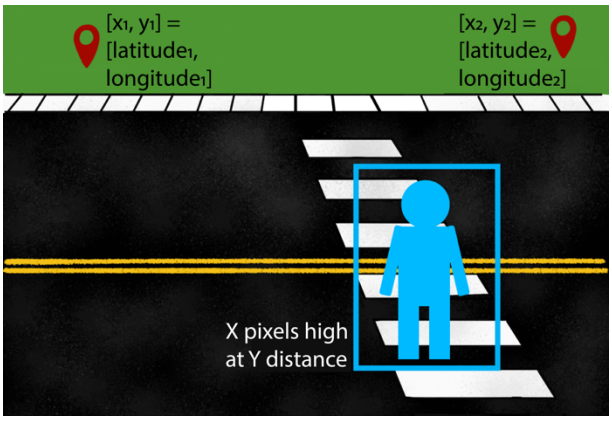

Figure 2. Detected object is closer to camera with larger pixel area within the bounding box.

3) Obtaining Distance from Three Known Points

There are three required distances: that from each of the two reference points and that from the camera. The distance from the reference points can be approximately calculated using a pixel scalar. This scalar is calculated using a formula which provides the distance in meters between two geolocations, which can be used to determine the physical distance between the two reference points in the camera's frame of view. Once complete, the physical distance can be used with the pixel distance between the reference points to create a ratio which can approximately convert pixels to meters within the image.

Next, the distance in pixels between the bottom-center point of a detected object and each of the known geolocations are calculated. By choosing the bottom-center point, we are allowing the point used to calculate distance to remain the same. This point only changes in regards to two dimensions instead of three. If the centermost point were used, then both the X and Y coordinates of the center would change corresponding not only to the objects X and Y movement, but also Z movement; as the detected entity moves closer or further from the camera, the bounding box size changes and with it, so does the center. If the center point of the bounding box were to be used, the point could shift with the increase or decrease of the bounding box's size and cause less accuracy when calculating the distance between each of the reference points.

The distance from the camera can be calculated by using the bounding box distance and height from the calibration process. Combining these two along with the current height of the bounding box gets us a distance:

$$\frac{DistanceAtCalibration}{(CurrentPixelHeight \div PixelHeightAtCalibration)}$$

The trilateration process must then be adapted from typical implementations. In the typical implementation all reference points are known and intersect at the same location, whereas in

1) Object Detection

For object detection, a CNN called You Only Look Once (YOLO) was used. YOLO version 2 can detect 80 different class types using the pre-trained model [5]. These classes include people, bicycles, cars, motorbikes, airplanes, and 75 other common objects. Using YOLO, multiple common roadway objects can be detected and tracked, further increasing driver awareness. YOLO was trained on full images rather than separate sub-sections of a single image, which increased its overall detection performance [5]. The YOLO network can analyze real-time video at approximately 45 frames per second (fps) with a latency of less than 25 milliseconds using the Titan X graphical processing unit (GPU) [5]. Another advantage to using YOLO is that the CNN looks at the training and test images globally rather than sub-sections, and it can therefore understand the features of the entire image compared to sub-sections of the image (i.e. sliding windows) [5]. This approach assists in less false positives when analyzing an image and ultimately increases its overall accuracy. This object detection process is also used in the calibration step that will be explained in the next sub-section.

2) Camera Calibration Process

The first step to obtaining the three known location points used to find the geographical coordinates of the detected object is to calibrate the camera. During the calibration procedure, absolute latitude and longitude coordinates are used together to determine the geolocation of the camera. Once the geolocation of the camera has been determined, two reference points on the left and right side of the camera's field of view are hardcoded as GPS coordinates into the calibration method to give a reference to compare with the camera and detected object to calculate and approximate location of the detected object.

During the calibration process, a person must stand a known distance away in the camera frame. At this point in the calibration process, YOLO gives an approximate pixel-height of the detected person where "x" is width and "y" is height. If the number of pixels in the detected object's bounding box increases, we can assume the object is moving closer to the camera, and if the pixel numbers decrease, we can assume the object is moving away from the camera based on the bounding box pixel area from the original calibration (see Figures 1 and 2).

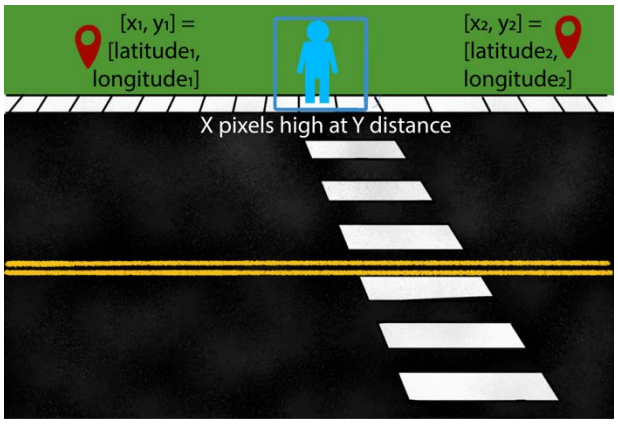

Figure 1. Detected object is farther away from camera and has smaller pixel area within bounding box.

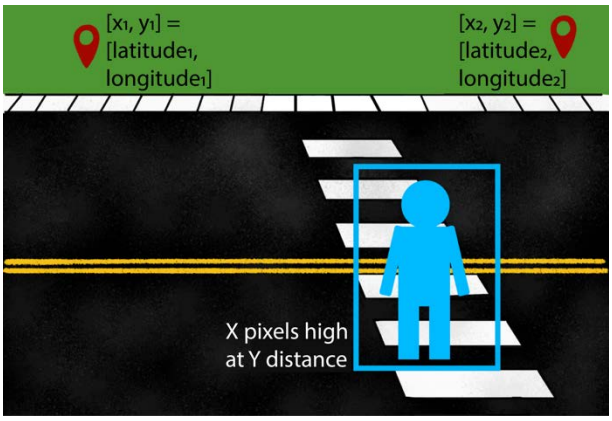

Figure 2. Detected object is closer to camera with larger pixel area within the bounding box.

3) Obtaining Distance from Three Known Points

There are three required distances: that from each of the two reference points and that from the camera. The distance from the reference points can be approximately calculated using a pixel scalar. This scalar is calculated using a formula which provides the distance in meters between two geolocations, which can be used to determine the physical distance between the two reference points in the camera's frame of view. Once complete, the physical distance can be used with the pixel distance between the reference points to create a ratio which can approximately convert pixels to meters within the image.

Next, the distance in pixels between the bottom-center point of a detected object and each of the known geolocations are calculated. By choosing the bottom-center point, we are allowing the point used to calculate distance to remain the same. This point only changes in regards to two dimensions instead of three. If the centermost point were used, then both the X and Y coordinates of the center would change corresponding not only to the objects X and Y movement, but also Z movement; as the detected entity moves closer or further from the camera, the bounding box size changes and with it, so does the center. If the center point of the bounding box were to be used, the point could shift with the increase or decrease of the bounding box's size and cause less accuracy when calculating the distance between each of the reference points.

The distance from the camera can be calculated by using the bounding box distance and height from the calibration process. Combining these two along with the current height of the bounding box gets us a distance:

$$\frac{DistanceAtCalibration}{(CurrentPixelHeight \div PixelHeightAtCalibration)}$$

The trilateration process must then be adapted from typical implementations. In the typical implementation all reference points are known and intersect at the same location, whereas in

this implementation measurement data taken via computer vision are approximate. Therefore, minor adjustments must be made. The algorithm begins by creating a ratio from the actual distance, in meters, and distance in the image, in pixels, between the reference points. This ratio is then applied to the pixel distance between the subject and both reference points. This gives two approximate radii for the circles needed to be formed. The third circle is formed from the point where the camera is mounted. An approximate distance from the camera can be calculated using data from the calibration process. During this process, the pixel-height of a person standing at a known distance is recorded, as is the known distance. With this, the new pixel-height can be used to approximate a new distance. The intersections of the three circles can be used to determine approximately where the subject is located in real, three-dimensional space.

Using the distance and geolocation two circles are formed: one at the camera and one at either reference point. The radii of the circles are their respective distances. Next, the two intersections between these circles are found and the intersection closest to the unused reference point is chosen, unless it is smaller; in this case, the further point is chosen (see Figures 3 and 4). Through this method, the ability to take the pixel coordinates of any object in an image and convert them to a geolocation that is not relative to the scene is possible.

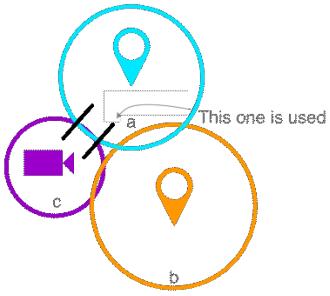

Figure 3. Intersection point used in the case only two areas intersect.

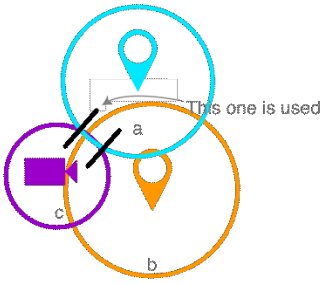

Figure 4. Intersection point used in the case all three areas intersect.

B. *Tracking Objects via GPS-Enabled Devices*

1) Cellular GPS

The second form of object localization was done by utilizing the GPS functionality of a user's mobile device if the All-in-One (AIO) mobile mapping application is downloaded onto a user's iOS device. Given the application has access to the user's location services, the application will send the user's coordinates to the database to be mapped. To obtain the user's GPS coordinates, the application utilized a Cordova plugin to provide current GPS information collected from a combination of network signals such as the IP address, Wi-Fi, Bluetooth MAC addresses, RFID, and GSM/CDMA cell IDS [6]. The user's device geo-coordinates were then updated every 250 milliseconds while the user had the application open and active.

2) Mobile Application Anonymity and Mobile Type Classification

To ensure the security of the user's location once the application had been downloaded, the AIO application requires iOS privacy control permissions to be granted by the user before it could access its location. The application did not have the ability to monitor a user's location while inactive on the mobile device. Privacy and anonymity concerns were prioritized, and no self-identifying ID, service provider, or name information was associated to any of the participating device's geo-coordinates stored in the database. Random IDs were generated by the application and used to identify entries on the Firebase server. A user could choose to identify their type of transportation as pedestrian, vehicle, or cyclist. A user who chose not to identify their transportation type was treated as a pedestrian. The description of the user was associated with the device's geo-coordinates and sent back to the cloud database. The transportation's type was then used to determine the type of icon that would be used to visualize the coordinates on the mobile map.

Apart from the initial image from the camera or GPS coordinates from the mobile device, all processing and data extraction explained during the previous three sections is performed on the off-site server. Given the amount of data this project was able to transfer in real-time with minimal delay, this demonstrates the possibility of the creation or utilization this high-speed network for other Big Data applications in the future.

C. *Mapping Objects*

Once the objects have been identified and the location has been determined, the mobile application or computer vision algorithm will automatically upload the relative information, such as the latitude and longitude of the object, to Google's Firebase database. Using the relative information, the database places a custom icon corresponding to the identification of the object onto the Google Maps API used for this project. Each of the icons shown on the screen of the application, shown in Figure 5, represent a different object based on the user's preference on the mobile device and the identity the computer vision algorithm gave the object upon detection. These details were successfully implemented into a mobile-based map as well as a web-based map to show each of the icons moving in real-time along with the user. The process of obtaining the data

represented on the application from both the infrastructure camera and mobile application is shown in Figure 6.

Figure 5. Custom icons created to represent different objects related to items being used by the application and placed on the map.

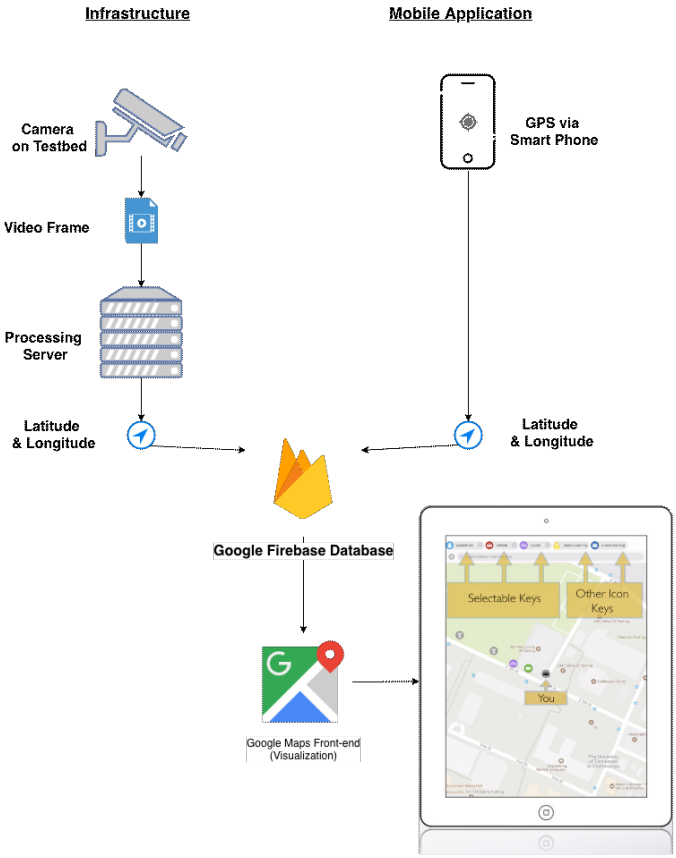

Figure 6. Architecture for object data mapped to application.

A key feature that needed to be confronted was the ability to delete items from the Google Firebase database. This was necessary to avoid having objects displayed on the map that have already left the camera's field of view or if a user has closed the application. The solution was simple for the mobile-based communication; if the device stopped sending information for a certain period of time, the database will delete their entry and delete their icon from the map. Regarding deleting the entry from the database in relation to the camera, if the object moves out of the camera's field of view, the database entry related to that object is deleted. In doing this, the stability of tracking of the object and retaining the unique identification information of the object had to be continuously considered.

For the mobile front-end map application, choosing Google Maps had several benefits in addition to its multifaceted API. Aside from its availability on several platforms, this choice was advantageous for its ability to utilize custom icons for its markers. The latitude and longitude stored in the database were used to determine the location of these markers, and their location was updated in a regular interval to correspond to the user's movement. This polling rate from the server on the front-end corresponds with the update rate of the locations from the computer vision algorithm.

In addition to the mobile front-end, a web interface can be made for testing and performance benchmarking. The Google Maps and Google Firebase APIs both offer JavaScript implementations, which aid in the creation of a rudimentary map which can track the same subjects the mobile front-end can see. This JavaScript-based front-end uses the same polling rate from the database as the mobile front-end and the computer vision algorithm.

D. Centralized Traffic Management System Design Utilizing SUMO

To facilitate sharing of collected data throughout the vehicular and pedestrian environments, a system is proposed where the Centralized Traffic Management System (CTMS) along with computational power inside the vehicle will work in concert and will be able to access road network traffic conditions and optimize road edge vehicle loading. The CTMS will accumulate the information from different road segments and constantly re-evaluate the most optimal routes for vehicles. See-through [7, 8] with AIO technology developed for this project by The University of Tennessee at Chattanooga (UTC) can be employed and communicate via V2V and V2I channels. Through V2V communication, vehicles can share the information about pedestrians, bicycles, or obstacles on the road and at the same time push this information to the CTMS to be used by route planning algorithms and other vehicles. The CTMS is intended to provide information, a monitoring function, navigation, etc. The distributed control uses the computing resources on the vehicles to capture data and send information. The CTMS is designed to listen to the vehicles via the AIO application and other stationary cameras for information and calculate routes for the vehicle. The system contains two major components, a server to communicate with the car (V2X) and a virtual traffic simulation to track vehicles on the road and calculate optimal routes using the A* search algorithm. To create the described system, a virtual traffic simulation was created using the Simulation of Urban Mobility (SUMO) [9] as a simulation environment, a server which communicates with the car to receive and send information and route and reroute vehicles, and an in-car AIO mobility application that can send and receive route information and can graphically represent the route to the driver. SUMO simulation software was chosen for this study as it includes required modules such as altering traffic lights, simulating variable message signs, and simulating Bluetooth devices. In addition to having many developed modules built in, the open source

nature of the software allows for easy development of required modules using Python. The Traffic Control Interface (TraCI) feature of the software gives access to the running traffic simulation allowing the user to alter the simulation in real-time. TraCI has a Python interface which facilitates the interaction between the simulation and the server.

E. Routing Capabilities

The routing functionality was also integrated with a CTMS in order to test routes and analyze traffic situations. Typically routing and re-routing is done based on current road conditions such as speed and road blockages at the time when route is calculated. For this project we are using traffic modeling and simulation to assign routes based on simulated conditions for the duration of the trip and not just current conditions. Routing that is done based on current speeds for instance does not consider that speeds may change in the future and original route may not be optimal at that time. In the simulation we are looking ahead into the future and assigning routes based on condition that will exist throughout the trip. Example of routing and re-routing is shown in Figure 7 and Figure 8. Figure 7 displays route that was originally identified as the most optimal route by the simulation. Once the obstacle on the road is detected (as shown in Figure 8) then the simulation created alternative route shown in Figure 8. For this test, a simple traffic scenario was chosen in order to explain how the vehicle was routed using a two-step process. First, a CTMS was used to simulate the traffic environment and decide the most efficient path possible using rerouting algorithms. Second, data displayed in AIO application will be provided by the CTMS.

Figure 7. Original route taken by vehicle given by a navigation application.

Figure 8. Alternative route given by application once object has been detected and reported to the server.

III. Results and Discussion

A. Object Tracking

Combining the components discussed in the methods section provides an integrated solution for congregating data gathered. Figure 9 and Figure 10 show the driver's actual field of view (left) relative to the AIO application communication range (right). The application also gives the driver a wider view of objects to come ahead while driving and can help them prepare decision-making tasks earlier compared to what they are able to see only from their field of view. For example, in Figure 10, the driver is able to see a stalled vehicle on the side of the road, but the hill in front of them prevents the ability to see the vehicle approaching them from the other lane. If the driver were to pass the stalled vehicle, an accident could occur, but using the AIO mapping application, the driver is able to see a vehicle approaching on the map and avoid an accident by waiting for the driver to pass and rechecking the map to make sure the lane is clear before advancing around the stalled vehicle.

Figure 9. An infrastructure camera captures video from its field of view (left) and the information is displayed in real-time on the mapping application (right).

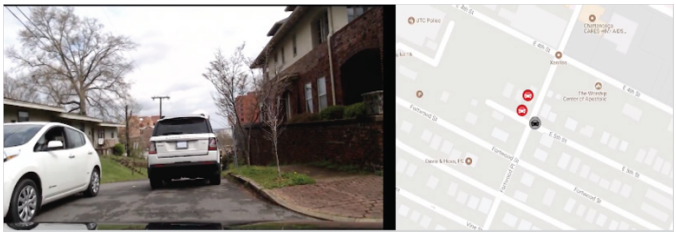

Figure 10. A vehicle waits as another vehicle passes to allow the driver to drive around the stalled vehicle on the side of the road (left) and maps the three vehicles and their positions on the real-time map (right).

As the web-based front-end is written in JavaScript, it is trivial to keep its features in line with the mobile application. As such, a web front-end has been launched for testing purposes with no issue. This interface does not offer tracking of the current user (mostly due the lower accuracy of IP-based geolocation), but it does allow the user to pan through the Google Maps window and view all tracked subjects. During this phase the accuracy of our system has a margin of error of +/- 3 meters. Reducing this error is one of the future goals of the project.

B. Rerouting Due to Obstacle

The current implementation for routing and rerouting drivers requires the two-step process described in the methods section. The simulation produces an original route which the vehicle uses until the obstacle or a road closure is reported on the route. If this happens the simulation provides an alternative route based on what the CTMS deems most efficient in the simulated traffic environment. The AIO rerouting feature can help drivers avoid potentially hazardous obstacles detected by the infrastructure cameras by giving the driver an alternative route to consider. Furthermore, this feature adds to the AIO application's ability to assist the driver by visualizing useful roadway information that can aid the driver's decision-making ability.

IV. Conclusion and Future Research

Our vision is to continue to develop the CTMS, infrastructure object detection, and the AIO application. The way it will work is as the vehicle moves on a road towards its destination, the stationary cameras or other vehicles gather information about obstacles on the road and sends it to the CTMS. The CTMS runs a simulation including data that was just received in real time and calculates the alternative routes for the vehicles on the road and sends the new route information to the vehicles. Using this approach will help us to have system level control over the road network and optimize the performance. Our next steps for this research include the investigation of the efficiency of various routing algorithms like A* search, Dijkstra, etc. We also plan to explore the efficient use of IVC to increase mobility, improve safety, and investigate benefits of connected vehicles and cooperative mobility. The other possible application of this technology can be platooning among connected fleet to increase the road throughput capacity. The CTMS used in this application could be tested on Chattanooga's network over time to observe how large these applications could theoretically be scaled before implementing the in a real-world environment. If successful, a similar model could be tested elsewhere with another city's data to enhance the development of their roadways.

Additional future research includes the possibility of implementing UTC's See-Through technology into the AIO application. This integration could expand the view range of a driver and increase their awareness of the driving environment ahead. Further work can be accomplished through the addition of cameras along UTC's testbed in order to add more locations where computer vision detection could take place. Gathering obstacle and pedestrian activity from supplementary locations would allow the AIO application to provide a more accurate map of the current environment to the AIO user. This would in turn help the user to make better informed decisions, resulting in a campus safer for drivers and pedestrians. Next steps will also be taken towards automating the obstacle detection rerouting process. The computer vision implementation of tracking objects offers the ability to detect and track up to 80 different objects. Should one of the objects not be categorized as a pedestrian, vehicle, or bicycle, then it is broadly categorized as an object warning. In the event that an object warning is detected, future research will be focused on implementing the optional rerouting process between the AIO application and the CTMS in an automated way that replaces the current manual two-step rerouting process.

Following the advancements on the application described above, we want to make scalability a primary focus. In doing so, we will be able to use large quantities of continuously flowing data that we anticipate from the upcoming testbed expansion in Chattanooga, Tennessee. This testbed will be connected to gigabit fiber internet with up to 10Gbps bandwidth, allowing our application to use the multiple cameras, radars, LiDARs, and other hardware located on the testbed. With access to this equipment and location, we can test the limits of our application to provide users, and eventually autonomous vehicles, supplemental roadway data to make better decisions. This will also provide enough data throughput for other Big Data applications to be developed and tested at varying scales.

Acknowledgments

The project team would like to thank EPB Fiber Optics, the City of Chattanooga, and the Enterprise Center of Chattanooga for their help in constructing the testbed used in this project. Also, we extend our gratitude to the NSF US Ignite for allowing us the opportunity to create this project by funding it under award number (1647161).

References

1. National Highway Traffic Safety Administration. « Critical Reasons for Crashes Investigated in the National Motor Vehicle Crash Causation Survey ». U.S. Department of Transportation. 2015.
2. Veloso, Marco & Bento, Carlos & Pereira, Francisco. (2009). Transportation Systems Working Paper Series Multi-Sensor Data Fusion on Intelligent Transport Systems.
3. Zhou, Mu, Albert Kai-Sun Wong, Zengshan Tian, Victoria Ying Zhang, Xiang Yu, and Xin Luo. "Adaptive Mobility Mapping for People Tracking Using Unlabelled Wi-Fi Shotgun Reads." *IEEE Communications Letters* 17, no. 1 (January 2013): 87-90. doi:10.1109/lcomm.2012.112812.121999.
4. La, Hung Manh, and Weihua Sheng. "Distributed Sensor Fusion for Scalar Field Mapping Using Mobile Sensor Networks." *IEEE Transactions on Cybernetics* 43, no. 2 (April 2013): 766-78. doi:10.1109/tsmcb.2012.2215919.
5. Redmon, J., Divvala, S., Girshick, R., and Farhadi, A., "You Only Look Once: Unified Real-Time Object Detection," https://pjreddie.com/media/files/papers/yolo.pdf , accessed Feb. 2017.
6. "Mirror of Apache Cordova Plugin Geolocation." GitHub. https://github.com/apache/cordova-plugin-geolocation.
7. Thompson, Rebekah L., Jose Stovall, Zhen Hu, Austin Harris, Jin Cho, and Mina Sartipi. "See-Through Technology Using V2X Communication." Proceedings of ACM Mid-Southeast, Gatlinburg, Tennessee.
8. Thompson, R., Hu, Z., Cho, J., Stovall, J. et al., "Enhancing Driver Awareness Using See-Through Technology," SAE Technical Paper 2018-01-0611, 2018, https://doi.org/10.4271/2018-01-0611.
9. Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. "Recent Development and Applications of SUMO - Simulation of Urban Mobility"; International Journal on Advances in Systems and Measurements, 5 (3&4):128-138, December 2012.
10. "Waze." Free Community-based GPS, Maps & Traffic Navigation App. 2018. Accessed July 23, 2018. https://www.waze.com/
11. Vincente Milané, Jorge Villagra, Jorge Godoy, Javier Simó, Joshué Pérez Rastelli, et al.. An Intelligent V2I-Based Traffic Management System. IEEE Transactions on Intelligent Transportation Systems, IEEE,2012,13(1),pp49-58. http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=armnumber=61210906. <hal-00732884>

this implementation measurement data taken via computer vision are approximate. Therefore, minor adjustments must be made. The algorithm begins by creating a ratio from the actual distance, in meters, and distance in the image, in pixels, between the reference points. This ratio is then applied to the pixel distance between the subject and both reference points. This gives two approximate radii for the circles needed to be formed. The third circle is formed from the point where the camera is mounted. An approximate distance from the camera can be calculated using data from the calibration process. During this process, the pixel-height of a person standing at a known distance is recorded, as is the known distance. With this, the new pixel-height can be used to approximate a new distance. The intersections of the three circles can be used to determine approximately where the subject is located in real, three-dimensional space.

Using the distance and geolocation two circles are formed: one at the camera and one at either reference point. The radii of the circles are their respective distances. Next, the two intersections between these circles are found and the intersection closest to the unused reference point is chosen, unless it is smaller; in this case, the further point is chosen (see Figures 3 and 4). Through this method, the ability to take the pixel coordinates of any object in an image and convert them to a geolocation that is not relative to the scene is possible.

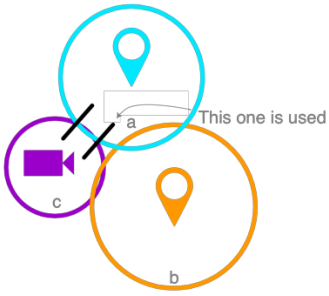

Figure 3. Intersection point used in the case only two areas intersect.

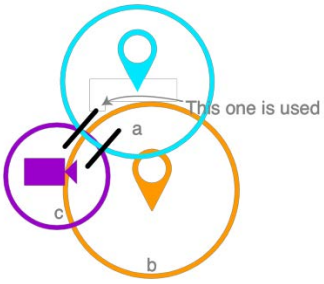

Figure 4. Intersection point used in the case all three areas intersect.

B. *Tracking Objects via GPS-Enabled Devices*

1) Cellular GPS

The second form of object localization was done by utilizing the GPS functionality of a user's mobile device if the All-in-One (AIO) mobile mapping application is downloaded onto a user's iOS device. Given the application has access to the user's location services, the application will send the user's coordinates to the database to be mapped. To obtain the user's GPS coordinates, the application utilized a Cordova plugin to provide current GPS information collected from a combination of network signals such as the IP address, Wi-Fi, Bluetooth MAC addresses, RFID, and GSM/CDMA cell IDS [6]. The user's device geo-coordinates were then updated every 250 milliseconds while the user had the application open and active.

2) Mobile Application Anonymity and Mobile Type Classification

To ensure the security of the user's location once the application had been downloaded, the AIO application requires iOS privacy control permissions to be granted by the user before it could access its location. The application did not have the ability to monitor a user's location while inactive on the mobile device. Privacy and anonymity concerns were prioritized, and no self-identifying ID, service provider, or name information was associated to any of the participating device's geo-coordinates stored in the database. Random IDs were generated by the application and used to identify entries on the Firebase server. A user could choose to identify their type of transportation as pedestrian, vehicle, or cyclist. A user who chose not to identify their transportation type was treated as a pedestrian. The description of the user was associated with the device's geo-coordinates and sent back to the cloud database. The transportation's type was then used to determine the type of icon that would be used to visualize the coordinates on the mobile map.

Apart from the initial image from the camera or GPS coordinates from the mobile device, all processing and data extraction explained during the previous three sections is performed on the off-site server. Given the amount of data this project was able to transfer in real-time with minimal delay, this demonstrates the possibility of the creation or utilization this high-speed network for other Big Data applications in the future.

C. *Mapping Objects*

Once the objects have been identified and the location has been determined, the mobile application or computer vision algorithm will automatically upload the relative information, such as the latitude and longitude of the object, to Google's Firebase database. Using the relative information, the database places a custom icon corresponding to the identification of the object onto the Google Maps API used for this project. Each of the icons shown on the screen of the application, shown in Figure 5, represent a different object based on the user's preference on the mobile device and the identity the computer vision algorithm gave the object upon detection. These details were successfully implemented into a mobile-based map as well as a web-based map to show each of the icons moving in real-time along with the user. The process of obtaining the data

represented on the application from both the infrastructure camera and mobile application is shown in Figure 6.

Figure 5. Custom icons created to represent different objects related to items being used by the application and placed on the map.

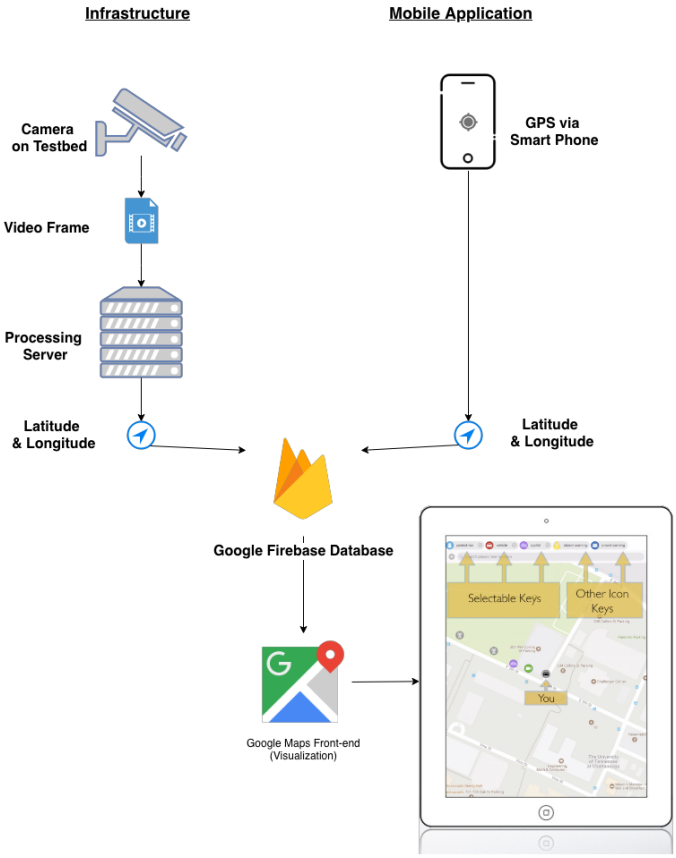

Figure 6. Architecture for object data mapped to application.

A key feature that needed to be confronted was the ability to delete items from the Google Firebase database. This was necessary to avoid having objects displayed on the map that have already left the camera's field of view or if a user has closed the application. The solution was simple for the mobile-based communication; if the device stopped sending information for a certain period of time, the database will delete their entry and delete their icon from the map. Regarding deleting the entry from the database in relation to the camera, if the object moves out of the camera's field of view, the database entry related to that object is deleted. In doing this, the stability of tracking of the object and retaining the unique identification information of the object had to be continuously considered.

For the mobile front-end map application, choosing Google Maps had several benefits in addition to its multifaceted API. Aside from its availability on several platforms, this choice was advantageous for its ability to utilize custom icons for its markers. The latitude and longitude stored in the database were used to determine the location of these markers, and their location was updated in a regular interval to correspond to the user's movement. This polling rate from the server on the front-end corresponds with the update rate of the locations from the computer vision algorithm.

In addition to the mobile front-end, a web interface can be made for testing and performance benchmarking. The Google Maps and Google Firebase APIs both offer JavaScript implementations, which aid in the creation of a rudimentary map which can track the same subjects the mobile front-end can see. This JavaScript-based front-end uses the same polling rate from the database as the mobile front-end and the computer vision algorithm.

D. Centralized Traffic Management System Design Utilizing SUMO

To facilitate sharing of collected data throughout the vehicular and pedestrian environments, a system is proposed where the Centralized Traffic Management System (CTMS) along with computational power inside the vehicle will work in concert and will be able to access road network traffic conditions and optimize road edge vehicle loading. The CTMS will accumulate the information from different road segments and constantly re-evaluate the most optimal routes for vehicles. See-through [7, 8] with AIO technology developed for this project by The University of Tennessee at Chattanooga (UTC) can be employed and communicate via V2V and V2I channels. Through V2V communication, vehicles can share the information about pedestrians, bicycles, or obstacles on the road and at the same time push this information to the CTMS to be used by route planning algorithms and other vehicles. The CTMS is intended to provide information, a monitoring function, navigation, etc. The distributed control uses the computing resources on the vehicles to capture data and send information. The CTMS is designed to listen to the vehicles via the AIO application and other stationary cameras for information and calculate routes for the vehicle. The system contains two major components, a server to communicate with the car (V2X) and a virtual traffic simulation to track vehicles on the road and calculate optimal routes using the A* search algorithm. To create the described system, a virtual traffic simulation was created using the Simulation of Urban Mobility (SUMO) [9] as a simulation environment, a server which communicates with the car to receive and send information and route and reroute vehicles, and an in-car AIO mobility application that can send and receive route information and can graphically represent the route to the driver. SUMO simulation software was chosen for this study as it includes required modules such as altering traffic lights, simulating variable message signs, and simulating Bluetooth devices. In addition to having many developed modules built in, the open source

nature of the software allows for easy development of required modules using Python. The Traffic Control Interface (TraCI) feature of the software gives access to the running traffic simulation allowing the user to alter the simulation in real-time. TraCI has a Python interface which facilitates the interaction between the simulation and the server.

E. Routing Capabilities

The routing functionality was also integrated with a CTMS in order to test routes and analyze traffic situations. Typically routing and re-routing is done based on current road conditions such as speed and road blockages at the time when route is calculated. For this project we are using traffic modeling and simulation to assign routes based on simulated conditions for the duration of the trip and not just current conditions. Routing that is done based on current speeds for instance does not consider that speeds may change in the future and original route may not be optimal at that time. In the simulation we are looking ahead into the future and assigning routes based on condition that will exist throughout the trip. Example of routing and re-routing is shown in Figure 7 and Figure 8. Figure 7 displays route that was originally identified as the most optimal route by the simulation. Once the obstacle on the road is detected (as shown in Figure 8) then the simulation created alternative route shown in Figure 8. For this test, a simple traffic scenario was chosen in order to explain how the vehicle was routed using a two-step process. First, a CTMS was used to simulate the traffic environment and decide the most efficient path possible using rerouting algorithms. Second, data displayed in AIO application will be provided by the CTMS.

Figure 7. Original route taken by vehicle given by a navigation application.

Figure 8. Alternative route given by application once object has been detected and reported to the server.

III. Results and Discussion

A. Object Tracking

Combining the components discussed in the methods section provides an integrated solution for congregating data gathered. Figure 9 and Figure 10 show the driver's actual field of view (left) relative to the AIO application communication range (right). The application also gives the driver a wider view of objects to come ahead while driving and can help them prepare decision-making tasks earlier compared to what they are able to see only from their field of view. For example, in Figure 10, the driver is able to see a stalled vehicle on the side of the road, but the hill in front of them prevents the ability to see the vehicle approaching them from the other lane. If the driver were to pass the stalled vehicle, an accident could occur, but using the AIO mapping application, the driver is able to see a vehicle approaching on the map and avoid an accident by waiting for the driver to pass and rechecking the map to make sure the lane is clear before advancing around the stalled vehicle.

Figure 9. An infrastructure camera captures video from its field of view (left) and the information is displayed in real-time on the mapping application (right).

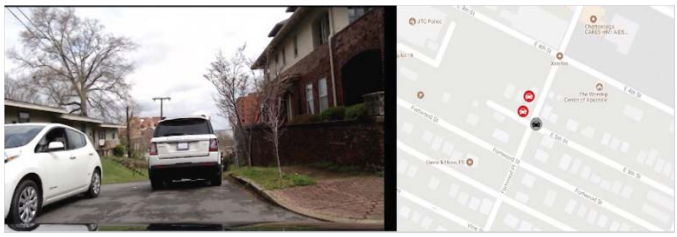

Figure 10. A vehicle waits as another vehicle passes to allow the driver to drive around the stalled vehicle on the side of the road (left) and maps the three vehicles and their positions on the real-time map (right).

As the web-based front-end is written in JavaScript, it is trivial to keep its features in line with the mobile application. As such, a web front-end has been launched for testing purposes with no issue. This interface does not offer tracking of the current user (mostly due the lower accuracy of IP-based geolocation), but it does allow the user to pan through the Google Maps window and view all tracked subjects. During this phase the accuracy of our system has a margin of error of +/- 3 meters. Reducing this error is one of the future goals of the project.