A Cloud-Assisted Design for Autonomous Driving

Swarun Kumar, Shyamnath Gollakota and Dina Katabi
Massachusetts Institute of Technology
{swarun, gshyam, dk}@mit.edu

ABSTRACT

This paper presents Carcel, a cloud-assisted system for autonomous driving. Carcel enables the cloud to have access to sensor data from autonomous vehicles as well as the roadside infrastructure. The cloud assists autonomous vehicles that use this system to avoid obstacles such as pedestrians and other vehicles that may not be directly detected by sensors on the vehicle. Further, Carcel enables vehicles to plan efficient paths that account for unexpected events such as road-work or accidents.

We evaluate a preliminary prototype of Carcel on a state-of-the-art autonomous driving system in an outdoor testbed including an autonomous golf car and six iRobot Create robots. Results show that Carcel reduces the average time vehicles need to detect obstacles such as pedestrians by 4.6× compared to today's systems that do not have access to the cloud.

CATEGORIES AND SUBJECT DESCRIPTORS

C.2.4 [**Computer Systems Organization**]: Computer-Communications Networks

GENERAL TERMS

Algorithms, Design, Performance

KEYWORDS

Autonomous Vehicles, Cloud, Wireless Networks

1. INTRODUCTION

Recently, much research has been focused on developing autonomous vehicles [9, 3, 14]. Such systems aim to navigate along an efficient and safe path from a source to destination, in the presence of changes in the environment due to pedestrians and other obstructions. Further, these systems need to deal with unexpected events such as road-blocks, traffic and accidents. Thus, autonomous vehicles need detailed and real-time information about their surroundings [11]. Typically, they use on-board sensors such as laser range-finders to build a *3D point-cloud* (see figure 2) representing the 3D (x, y, z)-coordinates of points on the surface of obstacles. However, such sensors cannot deliver vehicles information about hidden obstacles which are not directly in their line-of-sight, such as pedestrians hidden from view at an intersection. Further, these sensors are limited in range, and hence cannot report long-range data with sufficient accuracy; thereby limiting the vehicle's ability to plan ahead [14]. For these reasons, the report from the DARPA Urban Challenge identifies the need for autonomous vehicles to have access to each other's information as a key lesson learned from the contest [2].

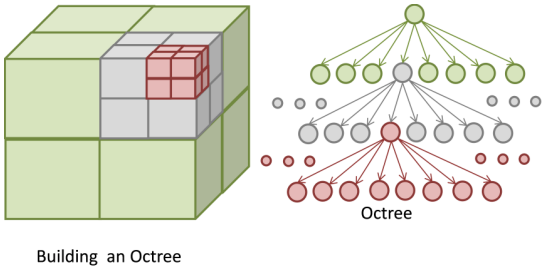

Figure 1—Representation of regions using Octree.

In this paper, we propose Carcel, a system where the cloud obtains sensor information from autonomous vehicles and static roadside infrastructure to assist autonomous cars in planning their trajectories. Carcel enables autonomous vehicles to plan safer and more efficient paths using the cloud. In particular, the cloud requests sensor information from autonomous vehicles as well as static road-side infrastructure. Further, the cloud records the current trajectory of all the vehicles. The cloud aggregates all this information and conveys information about obstacles, blind spots, and alternate paths to each vehicle.

In Carcel, the cloud has access to a larger pool of sensor data from multiple autonomous vehicles and static roadside sensors, that enables safer and more efficient path planning. Thus, the cloud has access to sensor information which may be from blind spots or outside the range of sensors on a single vehicle. Thus, the cloud can assist vehicles to make intelligent decisions that foresee obstacle not directly sensed by sensors on the vehicle itself. Further, since the cloud has longer range information about the traffic patterns and other unexpected events such as road-word or accidents, Carcel enables autonomous vehicles to plan more efficient paths.

To realize the above design, Carcel has to address two main challenges.

(a) How does the cloud get real-time sensor data from autonomous vehicles over a limited-bandwidth wireless link? The wireless connection between an autonomous vehicle and the cloud has a limited bandwidth. Navigation sensors, however, can generate real-time data-streams at high data rates (often at Gb/s), that makes it impractical to transmit all sensor information within the available bandwidth. Moreover, it is important to note that different pieces of information gathered by sensors are of unequal importance. A communication system that does not recognize this fact, may inundate the cloud with irrelevant or stale information. Thus, we need to efficiently utilize the wireless links between the autonomous vehicle and the cloud.

Carcel achieves this by having a request-based system which enables autonomous cars to gather information about different locations at various resolutions. In particular, in Carcel the cloud requests information at a high resolution only for important regions in the environment. The information for less important regions is

MCC'12, August 17, 2012, Helsinki, Finland.

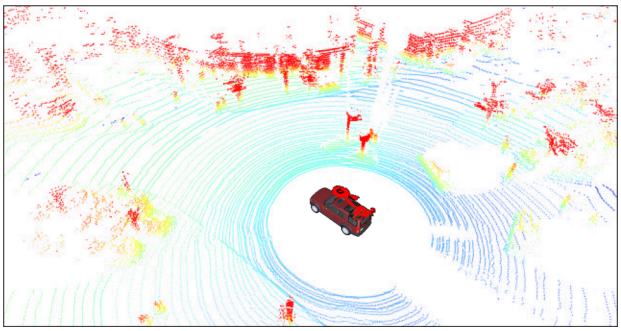

Figure 2—An example of sensory information gathered during autonomous driving. The figure shows a 3D-point cloud of a road obtained using a Velodyne laser sensor. The colors refer to elevation from the ground. A 3D-point cloud is a set of (x, y, z) coordinates of points lying on the surface of obstacles.

retrieved at a lower resolution, thereby consuming less bandwidth. The cloud infers the importance of different regions in the environment by taking into account: (1) the available low resolution information from these regions; (2) the current paths and locations of vehicles; and (3) the locations of the blind-spots of vehicles.

(b) How does the system deal with high packet losses that are inherent to wireless channels? Since the autonomous vehicles are mobile, the wireless channel experiences significant fading resulting in high packet losses. Since the communication between the autonomous vehicles and the cloud is real-time and delay-sensitive, we need to ensure that it is resilient to wireless losses.

Carcel achieves this by representing the transmitted sensor information over the wireless channel in a manner resilient to packet losses. Specifically, Carcel leverages the Octree data-structure, used typically in graphics to represent 3D objects, to provide resilience to wireless losses. At a high level, Octree is a recursive data-structure where the root represents the whole environment. As shown in Fig. 1, each node in an Octree describes a cube in the environment, and the sub-tree rooted at that node represents finer details contained in that cube. Carcel ensures that each packet transmitted to the cloud is self-contained by composing it with independent branches of the Octree derived from its root. Therefore, each packet received by the cloud can be reconstructed independent of other packets, ensuring that Carcel is resilient to packet loss.

We have built an initial prototype of Carcel using ROS, the Robot OS [12]. We integrated it with a state of the art path planner RRT*, an earlier version of which was used in the DARPA Urban Challenge [7]. We evaluated our system on an outdoor testbed composed of an autonomous Yamaha G22E golf car mounted with Hokuyo laser range sensors accessing sensory information of iRobot Create programmable robots connected to netbooks using Atheros AR9285 cards and gathering sensor information from Xbox 360 kinects. We tested the vehicle's ability to respond safely to pedestrians who suddenly exit a blind spot into a crosswalk along the car's path. Our findings are as follows:

- The cloud-based system can significantly reduce the vehicle's reaction time of today's baseline autonomous vehicles. In particular, Carcel detects the pedestrian with an average delay of 0.66 sec after the pedestrian enters the crosswalk, which is 4.6× smaller than the baseline which has a delay of 3.03 sec. If the golf car is traveling at a maximum velocity of 2 meters per second, our system is able to stop the golf car safely, even if the pedestrian appears when the golf car is 1-2 meters from the intersection. In contrast, the baseline is unable to stop the golf car safely, even if the pedestrian appears when the golf car is 6-8 meters from the intersection.
- Augmented the cloud capability to a system where the autonomous vehicles exchange information using an ad-hoc network can provide significant benefits. Specifically, Carcel detects the pedestrian with an average delay of 0.57 sec after the pedestrian enters the crosswalk, which is 3.8× smaller than that of a system where the vehicles share their sensor information only over the ad-hoc wireless network.

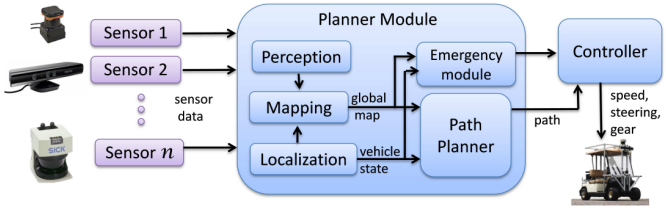

Figure 3—High-level Architecture of Autonomous Vehicles Sensor information from various sensors is provided to the path planner module which computes a path for the vehicle to navigate along.

Related Work. There has been much interest in building autonomous vehicles [9, 3, 14] spurred by public and private sector initiatives. In contrast, our paper explores the potential of integrating the cloud capability with autonomous vehicles. Our work is also related to work in networked robotics [4, 8] and vehicular ad-hoc networks (VANETs) [1, 15, 6]. While we build on this past work, Carcel focuses on communication and networking issues for autonomous vehicles. In particular, Carcel addresses the challenge of delivering high-bandwidth sensor data, inherent to autonomous vehicles, at low latencies to the cloud.

2. BACKGROUND

In this section, we provide a primer on the software architecture of autonomous vehicles. In order to navigate successfully, autonomous vehicles rely on on-board sensors to provide sensory information describing their environment. Sensors such as laser scanners and ultrasonic range finders provide accurate ranging information, describing the distance of vehicles to surrounding obstacles [14, 3, 9]. Vehicles may additionally augment their information using cameras and light sensors.

Autonomous vehicles most commonly use the Robot Operating System (ROS) framework [12]. ROS provides a publish/subscribe architecture, where modules (e.g. laser range-finders) publish information through topics (e.g. /laser_data) which can then be subscribed to by one or more modules. We describe the commonly described high-level modules of autonomous vehicles below: (Figure 3)

- **Sensor modules:** Each sensor attached to an autonomous vehicle has an associated module which converts raw sensor information into a generic format. The most commonly used format is the 3D-point cloud which provides a set of 3-Dimensional (x, y, z) coordinates of points that are located on the surface of obstacles. The sensor information is published, along with a time-stamp, denoting the time at which the sensor data was obtained.
- **Planner module:** The planner subscribes to information from the sensor modules and builds a detailed global map of the environment. It uses this map to compute an obstacle-free path for the vehicle towards the destination. The planner is sub-divided into five sub-modules:

  - *Perception*: The perception module subscribes to 3D-point cloud information from the sensor modules and uses obstacle de-

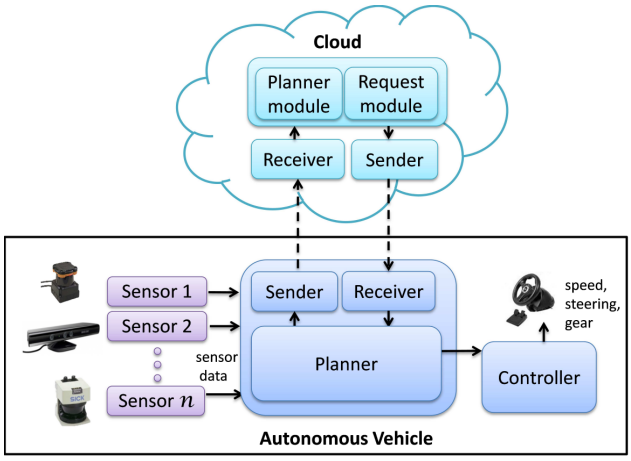

Figure 4—System Design. The figure illustrates the different modules in the autonomous vehicles and the cloud, and how they interact.

tection algorithms to identify obstacles in the frame of reference centered around the vehicle. The module then publishes a map of these obstacles.

- *Localization*: The localization module publishes the current location of the vehicle in a global map. The module often relies on a combination of GPS, odometry and other more advanced sensory infrastructure, often leading to accuracies as low as a few centimeters [9].
- *Mapper*: The mapper subscribes to the map of obstacles and the current position of the vehicle from the perception and localization modules respectively. It publishes a global map which contains the locations of all obstacles.
- *Path planner*: The path planner subscribes to the global map of obstacles from the mapper as well as the vehicle's position and publishes an obstacle-free path for the vehicle to navigate along.
- *Emergency module*: The emergency module subscribes to the map of obstacles and detects unexpected changes to the map resulting in new obstacles obstructing the vehicle's path (e.g. a pedestrian crossing the road unexpectedly). Depending on the location of the obstacle, it can either update the vehicle's trajectory path to maneuver around the obstacle or stop the vehicle altogether.

- **Controller**: The controller subscribes to the vehicle's planned path and issues steering, acceleration and velocity control commands to the vehicle so that it follows the planned path.

3. System Description

A high-level design of our cloud based system is shown in Figure 4. In Carcel, the cloud obtains sensor information from autonomous vehicles and static infrastructure sensors. However, navigation sensors can generate real-time data-streams at high data rates (often at Gb/s), that makes it impractical for vehicles to transmit all their sensor information within available bandwidth. A key observation is that different pieces of information gathered by sensors are of unequal importance. Carcel leverages this fact by making the cloud request information at a higher resolution, only from important regions in the environment.

The *request module* of the cloud issues these requests to the vehicle. It analyzes aggregate sensor information represented using the Octree data structure and generates requests for different regions at specific resolutions based on lacunae in the available sensor data. Requests are sent preferentially to regions which are closer to the current locations of vehicles, are close to the current trajectory of the vehicles or are in the vehicles' blind spots.

The *planner module* aggregates sensor information obtained from various autonomous vehicle and static road-side infrastructure sensors via the receiver module. It also records the current trajectory of all the vehicles. The planner analyzes the aggregate sensor information it has obtained to detect obstacles that may obstruct the current path of any vehicle. Information about obstacles and available alternate paths are conveyed to each vehicle via the sender module.

On each autonomous vehicle, the *receiver module* records requests and instructs the sender to transmit a greater proportion of sensor data from regions which have been requested, at the appropriate resolution. The receiver also alerts the planner of obstacles or alternate trajectories reported by the cloud. The *sender module* transmits sensor information from various requested regions, proportional to the number of requests received from the cloud for each of these regions. The sender also transmits the current location and trajectory of the vehicle.

To achieve the above design, we need to address the following key challenges:

(a) How do we represent different regions in the environment? Carcel uses the Octree representation to identify and name regions in the environment. In particular, Carcel partitions the world recursively into cubes. It begins with a known bounding cube that encompasses the entire environment of the vehicles. Each cube is then recursively divides into eight smaller cubes as depicted in figure 1. Carcel maintains an Octree, a recursive structure, where each node in the Octree represents a cube and a sub-tree rooted at that node represents the recursive divisions of the cube. Such a representation enables Carcel to name regions at different locations in the environment. Each node in the Octree (i.e. each region) is given a global unique identifier *id* to distinguish between different regions.

The Octree data structure can also be used to represent sensor information (i.e. a 3D-point cloud). Each node in the Octree (i.e. each region) is tagged with one of the following attributes: (1) *occupied*, if the point cloud representation has any points in the corresponding cube (i.e., the cube has some object and the car should not drive through it); (2) *unoccupied*, if there are no points in this region (i.e., the cube is vacant and the car may drive through it); and (3) *unknown*, if there is insufficient sensor information about this region (i.e., the cube may have some object, but these have not been picked up by the sensors). We note that a parent is occupied if any of its descendants are occupied. A parent is unoccupied only if all of its descendants are known and unoccupied. A parent is unknown, if none of its descendants are occupied and at least one of them is unknown.

(b) How does the cloud obtain information about regions at different resolutions? The Octree data structure provides us a mechanism to query information about various regions at different resolutions. In particular, information from deeper levels of the Octree provide higher resolution information about a region. Similarly, information at higher levels provide a lower resolution view of the same region. Thus, in Carcel, a request for a region is represented as a 2-tuple (id, res), where *id* represents the location of the region and *res* represents the resolution (depth) at which the region is requested. The autonomous vehicles can use their sensor information represented in the form of the Octree data structure to respond to such a request. In particular, the response contains the encoding of the sub-tree in the Octree data-structure rooted at node *id* of the tree, and truncated to depth *res*.

(c) How do autonomous vehicles transmit sensor information from these regions in a loss-resilient manner? Sensor informa-

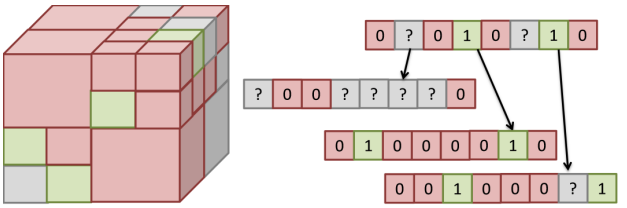

Figure 5—Octree Representation Representation of 3D-point cloud using a 2-level Octree. Vertices that are unoccupied are not expanded; vertices that are completely occupied or unknown are terminated by special leaf vertices; these have been omitted for clarity.

tion transmitted to the cloud can be significantly affected by packet loss that often occurs in a wireless medium. This is because, loss of a single packet in the Octree encoding of sensor data can often corrupt all the information stored in the Octree. To see why this is the case, let us first understand how the Octree data-structure is encoded [13, 5]. The standard Octree encoding technique proceeds as follows: Each node in the Octree be represented by a tuple of length 8 representing the occupancy of each of its children. Encoding is performed by traversing the tree in a top-down and breadth-first fashion and reading off the corresponding tuples. The root node is always assumed to be occupied. Now, for each node, its occupied and unknown children are recursively encoded. No information is encoded for unoccupied nodes since their corresponding regions are assumed to be entirely unoccupied. Additionally, nodes whose descendants are entirely occupied or unknown are terminated by a special 8-tuple (with all entries marked occupied or unknown respectively) and are not further encoded. At the receiver, the decoder can faithfully recover the Octree by following the same traversal rules as the encoder.

The problem with this encoding, however, is that loss of even a single byte in the Octree representation corrupts all data following that byte. This is because a byte corresponding to a node in the Octree describes the number and location of its children. Loss of this byte corrupts the entire sub-tree rooted at that node. For example, loss of the first byte in the Octree shown in Fig. 5 corrupts the entire Octree, since information detailing the locations of the root's child nodes is lost. To address this problem, Carcel ensures that each packet transmitted by the sender is self-contained and independent of other packets. In particular, Carcel randomly chooses sub-trees starting from the root node of the Octree, whose encoding fits within a single packet. To choose the sub-tree, Carcel picks a random leaf node l in the Octree. We construct the packet by including information about the path from this leaf node to the root. We also include information about all the descendants of the parent node $P(l)$ of leaf node l in the packet. We proceed recursively, including all descendants of the parent node $P(P(l))$ of node $P(l)$, as long as these are within the size of the packet. The sub-tree obtained at the end of this process has the property that the paths from all nodes in the sub-tree to the root of the Octree is known. This ensures that each packet can be reconstructed independent of other packets. Additionally, the bottom-up encoding process ensures minimal overlap between packets, thereby requiring fewer packets to encode the entire Octree.

Besides loss-resilience, Carcel's approach for encoding has several other desirable properties: (1) It is computationally efficient since it is linear in the size of the sub-tree corresponding to a region; (2) Packets received from different vehicles sensing the same region have minimal overlap, since the randomization process ensures that at any point in time, they transmit different parts of the region's Octree; and (3) It supports a form of unequal error protection because the transmitted sub-trees contain paths of all nodes to the root of the region. Therefore, nodes at higher levels in the tree are less likely to be lost than nodes at lower levels. Hence, the loss of a packet typically results in the loss of resolution as opposed to complete loss of information.

Figure 6—Carcel's Outdoor Setup. A picture of the actual golf car route showing the pedestrian crosswalk that poses a hazardous blind-spot for the golf car. This setup makes visual confirmation of a pedestrian difficult before he is actually on the road.

4. IMPLEMENTATION

We implement Carcel in the cloud as well as autonomous vehicles using the Robot Operating System (ROS) [12]. Our ROS implementation is operated on the Ubuntu 11.04 distribution (with linux kernel version 2.6.38-8-generic), that runs on the ASUS netbooks attached to the iRobot Create robots. We implement Carcel in an environment with a global map containing 80 regions. To have an initial evaluation of such a cloud based system, we use WiFi wireless links to connect to the cloud.

Our implementation of Carcel on autonomous vehicles subscribes to multiple topics containing sensor information in ROS's PointCloud format. The system tracks request messages from the cloud, and accordingly transmits the sensor information in the form of UDP packets to the cloud. The path planners in the vehicles and the cloud use the RRT* algorithm [7], an updated version of the autonomous path planning algorithm used by MIT's vehicle in the DARPA urban challenge.

We evaluate Carcel in an outdoor testbed in a campus-like setting that contains an autonomous Yamaha G22E golf car equipped with multiple sensors, including SICK range finders, cameras and Hokuyo laser sensors. The autonomous car navigates in a campus-like environment in the presence of other vehicles as well as pedestrians. We implement Carcel on the golf car as well as six iRobot Create robots mounted with Kinect sensors placed in multiple locations. The Kinects are connected to Asus EEPC 1015PX netbooks equipped with Atheros AR9285 wireless network adapters. A centralized server collects sensor information from the golf car and robots over a WiFi network. Figure 6 illustrates a pedestrian crosswalk where the experiments were conducted. Some of the robots were placed in the lobby adjacent to the crosswalk which was a blind-spot for the vehicle.

4.1 Metric

We use two metrics to evaluate Carcel:

- *Response time.* This measures the time it takes for the car to respond (send a STOP command) to a pedestrian, from the time the pedestrian starts entering the crosswalk.
- *Distance to pedestrian.* This measures the distance from the pedestrian crosswalk at which the car stops once it detects a pedestrian.

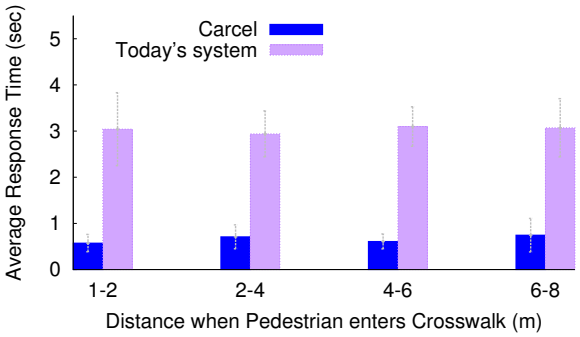

Figure 7—Time Delay Averages for Carcel and baseline. Distances on the x-axis are grouped into bins of two meters representing distances of the golf car from the crosswalk at the time the pedestrian enters the crosswalk.

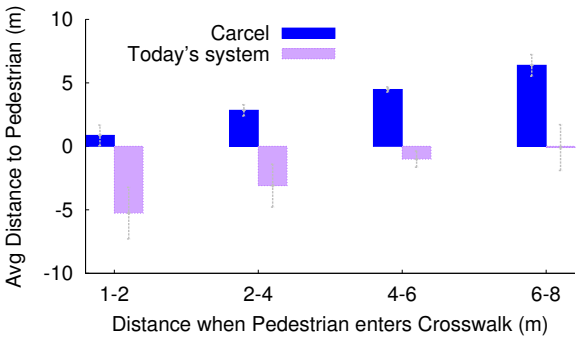

Figure 8—Comparison of golf car distance from the crosswalk when the pedestrian enters the crosswalk to the distance from crosswalk at which the receiver issues a stop command to the vehicle. Distances on the x-axis are grouped into bins of 2 m.

To measure the response time, we log the time at which the pedestrian was detected at the netbook mounted on the robot which is placed to view pedestrians entering the crosswalk. We compare this time against the time when the golf car's emergency module issues the stop command to the vehicle. All vehicles use NTP to synchronize their time within tens of milliseconds in the implementation. To compute the distance to pedestrian metric, we use the localization module on the vehicles that measure the distance of the vehicle from the crosswalk when the pedestrian entered the crosswalk, and when the golf car issued a stop command.

5. RESULTS

We compare the benefits of using a cloud-based system in two scenarios:

- The vehicles do not directly communicate with each other.
- The vehicles are additionally connected by an inter-vehicular ad-hoc wireless network over which they broadcast and share their sensor information.

Experiment 1: We first consider today's systems where the autonomous vehicles are neither connected to the cloud nor interconnected using an inter-vehicular network. We run an experiment where the pedestrian enters the blind spot in the crosswalk, when the vehicle is at distances of around eight meters, six meters, four meters and two meters from the crosswalk. The golf car travels at a speed of 2 meters/sec during the experiments. The results of the experiments are averaged over five runs at each of the distances.

Results: Fig. 7 plots the average reaction time in seconds. The x-axis plots the distance between the car and the crosswalk when the pedestrian starts crossing. The results show that Carcel's reaction times are significantly smaller than that of today's systems. The average time for pedestrian detection with Carcel is 0.66 sec, which is 4.6× smaller than the response time of 3.03 sec in today's systems.

Fig 8 plots the average distance to pedestrian in meters. The results show that today's systems that do not use the cloud, fail to stop the golf car even when it is around 6-8 meters away from the crosswalk as the pedestrian enters it. In contrast, with Carcel the golf car safely stops before the crosswalk, even when the pedestrian enters the crosswalk as the golf car is just 1-2 meters away from the crosswalk and the golf car is traveling at two meters per second. This is because, using the cloud, Carcel can receive information about the critical region in the blind spots of the vehicle. This is achieved by issuing requests for regions in the environment, including the blind spot, through which pedestrians may enter to obstruct the current planned path of the vehicle. This enables the cloud to accurately confirm pedestrian detection well before a similar conclusion is reached by the golf car's processor in today's systems.

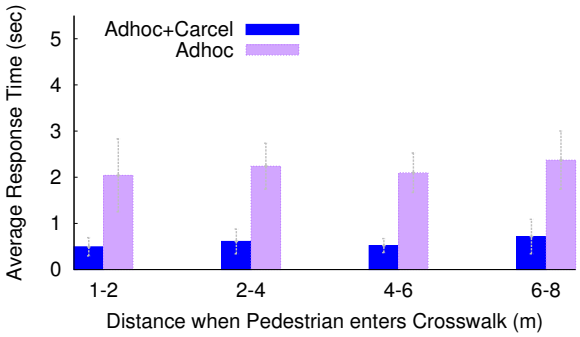

Figure 9—Time Delay Averages for Carcel and baseline when vehicles are additionally connected by ad-hoc wireless network. Distances on the x-axis are grouped into bins of 2 meters.

Experiment 2: Next, we consider a scenario where the vehicles are connected by an inter-vehicular ad-hoc wireless network over which they broadcast and share their sensor information. We evaluate the benefits of augmenting access to the cloud using Carcel, in such a scenario.

Results 2: Fig. 7 plots the average reaction time for both ad-hoc networked vehicles and a cloud augmented ad-hoc networked vehicles. The graph shows that Carcel's reaction times are still significantly smaller than the ad-hoc networked system. The maximum reaction time of pedestrian detection with Carcel is 0.57 sec, which is 3.8× smaller than the delay of 2.18 sec in the ad-hoc networked systems. This is because, Carcel can leverage the cloud to detect obstacles and pedestrians faster, even when the vehicles already use an ad-hoc wireless network to share information.

6. DISCUSSION

This paper provides a proof of concept of Carcel, where the cloud assists autonomous vehicles with path planning in order to avoid pedestrians and other obstacles in the environment. However, more research is needed to realize the full potential of the cloud. For instance, autonomous vehicles often require accurate localization, beyond what GPS and odometric sensors can provide [10]. By leveraging positions of known landmarks detected in the environments, vehicles can potentially obtain this localization information from the cloud. Similarly the cloud can provide a host of other services for autonomous vehicles, such as real-time information about traffic, congestion, detours, accidents, etc. The key challenge in design-

ing such a system is ensuring low latency and high availability over an inherently unreliable wireless medium. We believe that a comprehensive cloud-assisted design of autonomous vehicular systems can significantly improve the safety and reliability of autonomous vehicles and can bring commercial autonomous cars closer to reality.

Acknowledgments: We thank Lixin Shi, Stephanie Gil, Nabeel Ahmed and Prof. Daniela Rus for their support. This work is funded by NSF and SMART-FM. We thank the members of the MIT Center for Wireless Networks and Mobile Computing, including Amazon.com, Cisco, Intel, Mediatek, Microsoft, and ST Microelectronics, for their interest and support.

7. CONCLUSION

In this paper, we present Carcel, a cloud-assisted system for autonomous driving. Carcel enables autonomous cars to plan safer and more efficient paths by sharing their sensor information with the cloud. We evaluate an initial prototype of Carcel using field experiments with iRobot Create robots and a Yamaha golf car. Our results demonstrate that Carcel can significantly improve the safety of autonomous driving, by providing vehicles greater access to critical information in their blind spots.

8. REFERENCES

[1] L. Bononi and M. Di Felice. A cross layered mac and clustering scheme for efficient broadcast in vanets. In *MASS*, pages 1–8, 2007.

[2] M. Campbell, M. Egerstedt, J. P. How, and R. M. Murray. Autonomous driving in urban environments: approaches, lessons and challenges. *Philosophical Transactions of the Royal Society Series A*, 368:4649–4672, 2010.

[3] Z. Chong, B. Qin, T. Bandyopadhyay, T. Wongpiromsarn, E. Rankin, M. Ang, E. Frazzoli, D. Rus, D. Hsu, and K. Low. Autonomous personal vehicle for the first- and last-mile transportation services. In *CIS 2011*, pages 253 –260, 2011.

[4] S. Gil, M. Schwager, B. Julian, and D. Rus. Optimizing communication in air-ground robot networks using decentralized control. In *ICRA 2010*, pages 1964–71, 2010.

[5] Y. Huang, J. Peng, C.-C. Kuo, and M. Gopi. A generic scheme for progressive point cloud coding. *Visualization and Computer Graphics, IEEE Transactions on*, 2008.

[6] A. Iwai and M. Aoyama. Automotive cloud service systems based on service-oriented architecture and its evaluation. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 638 –645, july 2011.

[7] S. Karaman and E. Frazzoli. Sampling-based algorithms for optimal motion planning. *CoRR*, 2011.

[8] A. Kolling and S. Carpin. Multi-robot pursuit-evasion without maps. In *ICRA 2010*, pages 3045 –3051, 2010.

[9] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Kolter, D. Langer, O. Pink, V. Pratt, M. Sokolsky, G. Stanek, D. Stavens, A. Teichman, M. Werling, and S. Thrun. Towards fully autonomous driving: Systems and algorithms. In *IEEE IV 2011*, pages 163 –168, 2011.

[10] J. Levinson, M. Montemerlo, and S. Thrun. Map-based precision vehicle localization in urban environments. In *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.

[11] F. Maurelli, D. Droeschel, T. Wisspeintner, S. May, and H. Surmann. A 3d laser scanner system for autonomous vehicle navigation. In *ICAR 2009.*, pages 1 –6, june 2009.

[12] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[13] R. Schnabel and R. Klein. Octree-based point-cloud compression. In *Symposium on Point-Based Graphics*, 2006.

[14] C. Urmson, C. R. Baker , J. M. Dolan, P. Rybski, B. Salesky, W. R. L. Whittaker, D. Ferguson, and M. Darms. Autonomous driving in traffic: Boss and the urban challenge. *AI Magazine*, pages 17–29.

[15] J. Wang, J. Cho, S. Lee, and T. Ma. Real time services for future cloud computing enabled vehicle networks. In *Wireless Communications and Signal Processing (WCSP), 2011 International Conference on*, pages 1 –5, November 2011.