Augmented Vehicular Reality: Enabling Extended Vision for Future Vehicles

Hang Qiu
University of Southern California

Fawad Ahmad
University of Southern California

Ramesh Govindan
University of Southern California

Marco Gruteser
Rutgers University

Fan Bai
General Motors Research

Gorkem Kar
Rutgers University

ABSTRACT

Like today's autonomous vehicle prototypes, vehicles in the future will have rich sensors to map and identify objects in the environment. For example, many autonomous vehicle prototypes today come with line-of-sight depth perception sensors like 3D cameras. These cameras are used for improving vehicular safety in autonomous driving, but have fundamentally limited visibility due to occlusions, sensing range, and extreme weather and lighting conditions. To improve visibility and performance, not just for autonomous vehicles but for other Advanced Driving Assistance Systems (ADAS), we explore a capability called Augmented Vehicular Reality (AVR). AVR broadens the vehicle's visual horizon by enabling it to share visual information with other nearby vehicles, but requires careful techniques to align coordinate frames of reference, and to detect dynamic objects. Preliminary evaluations hint at the feasibility of AVR and also highlight research challenges in achieving AVR's potential to improve autonomous vehicles and ADAS.

CCS CONCEPTS

•Networks → Cyber-physical networks; •Computer systems organization → *Special purpose systems;*

KEYWORDS

Autonomous Cars, ADAS, Collaborative Sensing, Extended Vision

ACM Reference format:
Hang Qiu, Fawad Ahmad, Ramesh Govindan, Marco Gruteser, Fan Bai, and Gorkem Kar. 2017. Augmented Vehicular Reality: Enabling Extended Vision for Future Vehicles. In *Proceedings of Intl. Workshop on Mobile Computing Systems and Applications, Sonoma, CA, USA, February 21-22, 2017 (HotMobile '17),* 6 pages.
DOI: http://dx.doi.org/10.1145/3032970.3032976

1 INTRODUCTION

Autonomous cars are becoming a reality, but have to demonstrate reliability in the face of environmental uncertainty. A human driver can achieve, on average, ~100 million miles in between fatalities, and users will expect self-driving vehicles, and other advanced driving systems, to significantly outperform human reliability. Most of these technologies use advanced sensors for depth perception, and these sensors can be used to recognize objects and other hazards in the environment that may cause accidents.

To improve the system performance, it will be necessary to overcome the limitations of these sensors (§2). Most autonomous vehicles or ADAS technologies are equipped with depth perception sensors as well as cameras for semantic level perception. Besides limited sensing range, they all require line-of-sight visibility. However, achieving higher reliability will likely require appropriate handling of situations where (a) a vehicle's sensors cannot detect other traffic participants because line-of-sight availability does not exist, or (b) their range is impaired by extreme weather conditions, lighting conditions, sensor failures, *etc.*. In some of these cases, even human vision cannot recognize hazards in time.

In this paper, we explore the feasibility of communicating and merging *visual* information between nearby cars. This would augment vehicular visibility into hazards, and enable autonomous vehicles improve perception under challenging scenarios, or ADAS technologies to guide human users in making proper driving decisions. This capability, which we call Augmented Vehicular Reality (AVR), aims to combine emerging vision technologies that are being developed specifically for vehicles [4], together with off-the-shelf communication technologies.

Our preliminary design of AVR (§3) explores the use of stereo cameras, which can, with their depth perception, generate instantaneous 3-D views of the surroundings in a coordinate frame relative to the camera. Before it can share these instantaneous views between cars, AVR must solve three problems: how to find a common coordinate frame of reference between two cars; how to resolve perspective differences between the communicating cars; and how to minimize the communication bandwidth and latency for transferring 3-D views. To this end, we have developed novel techniques to localize a vehicle using sparse 3-D feature maps of the static environment crowd-sourced from other cars, to perform 3-D perspective transformations of the views of the cars, and to detect moving objects (as to minimize the visual information exchanged between vehicles).

In our preliminary evaluation, we demonstrate that these technologies can actually help extend the vision of neighboring vehicles (§4), and we quantify some of the bandwidth and latency costs of AVR. Much work remains in making AVR practical, including optimizing its vision processing pipeline, and aggressively reducing the bandwidth and latency by leveraging techniques such as compression and motion prediction. Our design builds on prior work in localization

This work is supported in part by the US National Science Foundation under grant number CNS-1329939 and by a grant from General Motors.

HotMobile '17, Sonoma, CA, USA
© 2017 ACM. 978-1-4503-4907-9/17/02…$15.00
DOI: http://dx.doi.org/10.1145/3032970.3032976

and scene matching (§6), but, to our knowledge, no one has explored this novel, and important, capability.

2 BACKGROUND, MOTIVATION AND CHALLENGES

Sensing Capabilities in Future Cars. A crucial component of an autonomous car is a 3D sensing capability that provides depth perception of a car's surroundings. Modern autonomous vehicle prototype mainly rely on a rich variety of accurate perception sensors, including advanced multi-beam LiDar, radar, long-range ultrasonic and forward-facing or surrounding-view camera sensors, to detect and track moving objects while producing a high-definition (HD) map for localization [28][29]. This HD map makes the car aware of its surroundings: *i.e.,* where is the curb, what is the height of the traffic light, *etc.*, and is able to provide sub-meter-level mapping and localization accuracy. Recent research in autonomous driving [1] [2] [3] has leveraged some of these advanced sensors to improve perception.

Abstractly, regardless of the details, this collection of sensors, generates successive *point clouds*, each of which represents the instantaneous 3D view of the environment. A point in the point cloud represents a voxel in the 3D space, and is associated with a position in 3-D and possibly a color attribute depending on the sensor used. For example, the 64-beam Velodyne LiDAR can collect point clouds at 10 Hz containing a total of 2.2 million points each second encompassing a 360°view. In this paper, we focus on stereo cameras, which can collect point clouds at faster rates (60 Hz) with over 55 million points per second, but with a limited field of view (110°).

The Problem. Most of these sensors only provide line-of-sight perception (*e.g.,* LiDar, Radar, (stereo) camera and Infrared sensors) and obstacles can often block a vehicle's sensing range. Moreover, the effective sensing range of these sensors is often limited by different weather conditions (*e.g.,* fog, rain, snow, *etc.*) or lighting conditions [26][14]. In this paper, we explore the feasibility of a simple idea: *extending the visual range of vehicles through communication*. We use the term *Augmented Vehicular Reality* (AVR) to denote the capability that embodies this idea.

AVR can be helpful in many settings. For example, consider a very simple platoon of two cars, a leader, and a follower. The leader can communicate, to the follower, those objects (e.g. stop signs, crosswalks, potholes, pedestrians) in its visual range that the follower is unlikely to be able to see, either because the leader obstructs the follower's view, or because the objects are beyond the follower's range. The human or autonomous driving system in the follower's car can use this information to make safe driving decisions. Specific examples include buses that occlude children crossing at a crosswalk, or trucks that occlude a left-turning vehicle's view of approaching cars.

AVR can also extend a vehicular vision over larger spatial regions, or over time. Today, for example, navigation apps like Waze warn users of hazards like parked cars on the shoulder, police vehicles, or objects on the road. An AVR capability across vehicles can potentially provide more accurate information (today's navigation apps rely on crowdsourcing, and their accuracy is spotty), together with precise positions of these hazards. Finally, AVR can also be used to augment HD maps to include temporary features in the environment like lane closures or other construction activity.

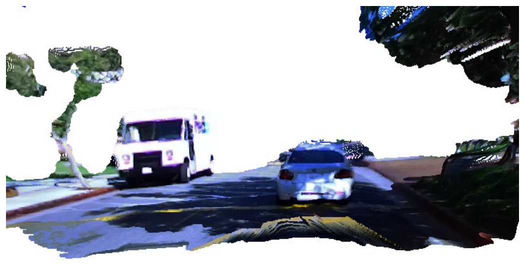
Figure 1—Single Vehicle View of Point Cloud

Challenges. An AVR capability poses several fundamental challenges, some of which we discuss in this paper, and others that we defer to future work §5.

First, for AVR, each vehicle needs to transform the view received from the other vehicles into its own view. To be able to do this very accurately, this *perspective transformation* needs both the exact position of the sensor (LiDar, camera, *etc.*) and the orientation at a high resolution, which can pose new challenges to current localization systems.

Second, the high volume of data generated by the sensors can easily overwhelm the capabilities of most existing or future wireless communication systems. Fortunately, successive point clouds contain significant redundancies. For example, static objects in the environment may, in most cases, not need to be communicated between vehicles, because they may already be available in precomputed HD maps stored onboard. Realizing AVR in the short term hinges on our ability to be able to isolate dynamic objects of interest for sharing between vehicles.

Finally, AVR needs to have an extremely low end to end latency in order to achieve real-time extended vision. Thus, it requires fast object extraction, low communication latency, and fast perspective transformation and merging processing. With the advent of specialized vision systems for vehicles [4], the latency of some of these steps is approaching the realm of feasibility, but low latency communication will remain a challenge. Finally, AVR requires tight time synchronization between sender and receiver in order to correctly position views received from other vehicles over time.

3 AVR DESIGN

AVR creates an extended 3D map, updated in real-time, of a vehicle's surroundings regardless of any line-of-sight occlusions or sensing range limitations. With the help of several state-of-the-art enabling technologies, AVR leverages a crowd-sourced *sparse* HD map to enable vehicles to position themselves relative to each other, by positioning themselves relative to the same static objects in the environment. It utilizes wireless communication to share vehicle views among other vehicles so that each of them is aware of not only the exact position but also the surroundings of its neighbor. To minimize the communication overhead, AVR shares only moving objects by carefully analyzing the motion of the objects in the environment.

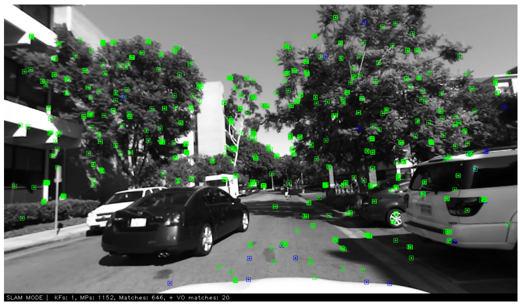

Figure 2—Feature Detection and 3D Matching

Specifically, AVR works as follows. Each vehicle continuously captures point clouds and, using a pre-computed sparse HD map containing *features* of static objects in the environment, localizes itself in a coordinate frame relative to the camera that captured the sparse HD map. Meanwhile, each vehicle also isolates point cloud of dynamic objects in the environment, and transmits these to vehicles nearby, either using V2V technologies, or using road-side or cloud infrastructure as an intermediary. Other vehicles can also position themselves in the same relative coordinate frame of reference, so, when they receive point clouds from other vehicles, they can merge these into their own camera's reference frame. Before doing so, they must do a perspective transformation that accounts for perspective differences between the two vehicles.

Since we are exploring the feasibility, our initial exploration of AVR uses an inexpensive (2 orders of magnitude cheaper than high-end LiDAR devices) off-the-shelf stereo camera, together with processing software that analyzes concurrent frames from the two cameras to determine the point cloud. Specifically, by analyzing the disparity between the left and right camera, the software can determine the 3D coordinate of each voxel, relative to the camera's coordinate frame of reference. Figure 1 shows the point cloud generated by the stereo camera while cruising on our campus. In addition to resolving the depth of the surroundings, AVR also incorporates a state-of-the-art object recognition framework [25]. Trained on a vehicular scenario, the framework detects interesting objects, *i.e.,* cars, pedestrians, *etc.*, and localizes and draws a 2D bounding box on the frame. In summary, each vehicle can continuously generate the location, and the type of the surrounding objects.

In theory, AVR can share information at several levels of granularity between vehicles: the entire point cloud at each instant, the point cloud representing some subset of objects in the environment, the object detected in a two-dimensional view, or the label (type) of object. These are in decreasing order of communication complexity, and we evaluate these later in §4.

3.1 Localization using Sparse 3-D Feature Maps

To solve the problem of localizing one vehicle with respect to another, AVR leverages prior work in stereo-vision based simultaneous localization and mapping (SLAM, [23]). This work generates sparse 3-D *features* of the environment, where each feature is associated with a precise position. AVR uses this capability in the following way. Suppose car A drives through a street, and computes the 3-D features of the environment using its stereo vision camera. These 3-D features contribute to a static map of the environment. Another car, B, if it has this static map, can use this idea in reverse: it can determine 3-D features using its stereo camera, then position those features in car A's coordinate frame of reference by matching the features, thereby enabling it to track its own camera's position. A third car, C, which also shares A's map, can position itself also in A's coordinate frame of reference. Thus, B and C can each position themselves in a consistent frame of reference, so that B can correctly overlay C's shared view over its own.

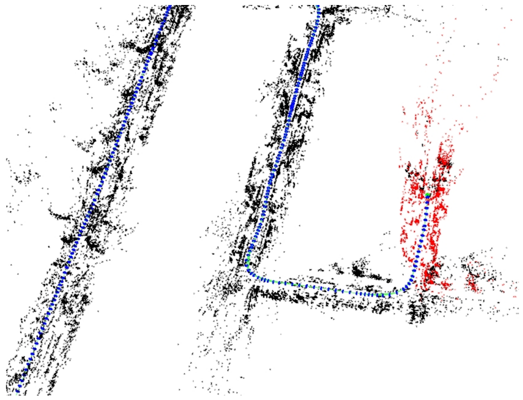

Figure 3—Crowdsourcing static HD map

In AVR, this idea works as follows. As a car traverses a street, all stable features on the street, from the buildings, the traffic signs, the sidewalks, *etc.*, are recorded, together with their coordinates, as if the camera were doing a 3D scan of the street. A feature is considered stable only when its absolute world 3D coordinates remain at the same position (within a noise threshold) across a series of consecutive frames. Figure 2 shows the features detected in an example frame. Each green dot represents a stable feature. Therefore, features from moving objects, such as the passing car on the left in (Figure 2), would not be matched or recorded.

Each car can then crowd-source its collected map. We have left to future work the mechanisms for this crowd-sourcing; even though it is relatively sparse, this map is still voluminous and is not amenable to real-time crowd-sourcing. However, it is relatively straightforward to stitch together crowd-sourced segments from two different cars, so that a consistent 3-D map can be built. Suppose car A traverses one segment of street X; car B can traverse that same segment of X, then upload a 3-D map of a traversal of a perpendicular street Y by placing Y's 3-D map in the same coordinate frame as X's. Figure 3 shows the static HD map created and the localized camera positions as it travels. Each black dot is a stored feature, whereas the red dots are those that are currently active for feature matching.

AVR needs only one traversal to collect features sufficient for a map since these features represent static objects in the environment. The amount of data needed for each road segment depends on the complexity of the environment. As an example, AVR creates 97MB of features for a 0.1 mile stretch of a road on our campus.

3.2 Extending Vehicular Vision

With the help of a common coordinate frame, vehicles are able to precisely localize themselves (more precisely, their cameras), both in 3D position and orientation, with respect to other vehicles easily. However, if car A wants to share its point cloud (or objects in it)

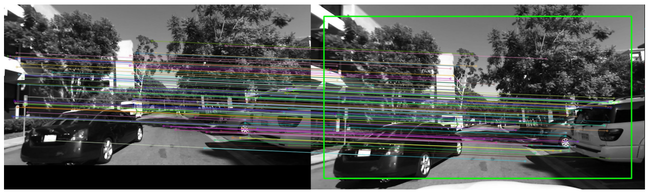

Figure 4—Finding the Homography of the Current Frame in the Previous Frame using SURF Features

Figure 5—ZED Stereo Camera

Figure 6—ZED Mounted on top of the Windshield with Smartphone Attached

with car *B*, AVR needs to transform the objects or point cloud in *A*'s local view to *B*'s. Vehicles can have very different perspectives of the world depending on the location and orientation of their sensors and point clouds are generated in the local coordinate frame. In this section, we describe how AVR performs *perspective transformations* between the local and global coordinate frames.

Consider two views from two different vehicles. AVR introduces the common coordinate frame to bridge the gap between the two perspectives. Specifically, the camera pose in the common frame is represented by a transformation matrix, Tcw, as shown in Equation (1), which includes a 3 x 3 rotation matrix and a 3 element translation vector.

$$Tcw = \begin{bmatrix} RotX.x & RotY.x & RotZ.x & Translation.x \\ RotX.y & RotY.y & RotZ.y & Translation.y \\ RotX.z & RotY.z & RotZ.z & Translation.z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

The translation is equivalent to the camera coordinate in the common frame, and the rotation matrix indicates the rotation of the camera coordinate frame against the common coordinate frame. Therefore, transforming a voxel in the camera (c) domain point cloud ($V = [x, y, z, 1]^T$) to a voxel in the world (w) domain ($V' = [x', y', z', 1]^T$) follows Equation (2):

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = Tcw * \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (2)$$

Assuming camera *A* has a pose of Taw, and camera *B* has a pose of Tbw, similarly, transforming a voxel V_a from camera *A* to X_b in camera *B* follows Equation (3).

$$V_b = Tbw^{-1} * Taw * V_a \quad (3)$$

3.3 Detecting and Isolating Dynamic Objects

Transferring and transforming the full point cloud of a vehicle's surroundings is infeasible given the constraints of today's wireless technologies. In this section, we explore whether it is possible to isolate only dynamic objects in the environment. A simple way to do this is to analyze the motion of each voxel in successive frames. Unfortunately, it is a non-trivial task to match two voxels among consecutive frames. Prior point cloud matching techniques [18, 22] involve heavy computation unsuitable for real-time applications.

AVR exploits the fact that its cameras capture video, and uses 2D feature matching to match 3-D voxels: in earlier steps of our computation, we compute the correspondence between pixels and voxels, details omitted, and in this step, we use this information.

AVR extracts SURF features for matching and finds the homography transformation matrix between two neighboring frames. Specifically, it tries to find the position of the current frame in the last frame. The intuition behind this is that vehicles usually move forward and the last frame often captures more of the scene than the current frame (Figure 4). Similar to the perspective transformation matrix discussed above (§3.2), the homography matrix H, which is in 2D, can transform one pixel ($P = [x, y]^T$) to the same pixel ($P' = [x'y']^T$) in the last frame with a different location. This pixel matching can be used to match corresponding voxels in the point cloud. Then, thresholding the Euclidean distance between matching voxels from consecutive frames can isolate the dynamic points from the static ones. Note that before calculating the displacement, the matching voxels should be transformed into the same coordinate frame, either the last frame or the current frame, following Equation (2). In our initial experiments, AVR can successfully filter out all stationary objects, and extract only the moving objects in the scene. With the vehicle cruising at 20mph, the stereo camera recording at 30fps, the average displacement of the stationary voxels is $< 5cm$ per frame.

One optimization we have experimented with is to exploit object detectors [25] to narrow the space for moving objects (cars, pedestrians). Once we have identified the pixels associated with these moving objects, we can determine if the corresponding voxels are moving, thereby reducing the search space for moving objects. We evaluate how much bandwidth this optimization can save in the following section §4.

4 PRELIMINARY EVALUATION

Experimental Setup. We experiment and evaluate AVR on two cars, each equipped with one ZED [5] (Figure 5) stereo camera mounted on the top of the front windshield and a smartphone attached to it (Figure 6). The stereo camera records the video stream and computes the 3D point cloud as well as depth information, while the mobile phone records GPS and all motion sensors, *i.e.,* gyroscope, accelerometer, magnetometer, *etc.*. ZED can create the real-time point cloud at a framerate of 30fps on a Titan X GPU with a resolution of 720P (1280 x 720), and up to 100 fps with VGA (640 x 480). For object recognition, AVR uses YOLO [25], a state-of-the-art object detector. For the SLAM algorithm, AVR uses ORB-SLAM2 [23], currently the highest ranked open source visual odometry algorithm on KITTI benchmark, a widely accepted vehicle vision benchmark. Using ORB-SLAM2, we have collected several traces on and around campus with one car following the other to both collect the sparse maps and to achieve extended vehicular vision.

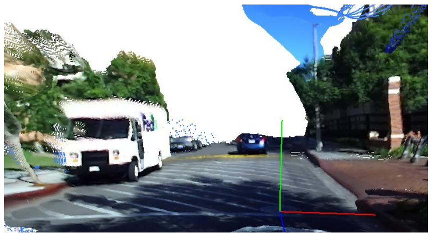
Figure 7—Point Cloud of Leader

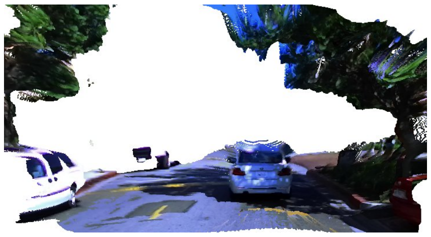
Figure 8—Point Cloud of Follower

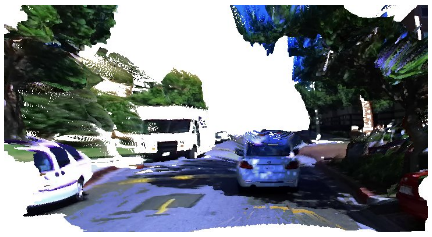
Figure 9—Extended Point Cloud

	VGA (640 x 480)	720P (1280 x 720)	1080P (1920 x 1080)
Full	4.91 MB	14.75 MB	33.18 MB
Dynamic	0.79 MB	2.36 MB	5.30 MB
Object	0.33 MB	0.98 MB	2.21 MB
Labels	0.05 MB	0.05 MB	0.05 MB

Table 1—Point Cloud Data Size Per Frame.

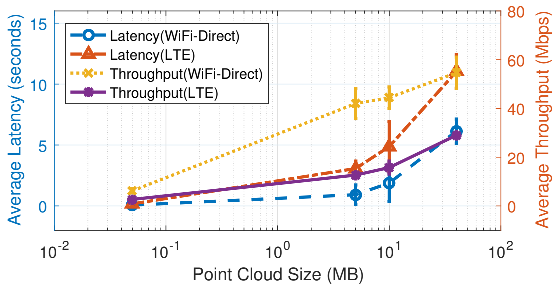

Figure 10—Throughput and Latency of Point Cloud Transmission

Results: Extended Vision. Figure 7 shows the point cloud view[1] of the leader, where there is FedEx truck parked on the left and a segment of sidewalk on the right. Figure 8 shows the perspective of the follower where a white sedan is parked on the left. Note that neither the FedEx truck or the extension of the sidewalk can be fully observed due to the limited sensing range of the ZED stereo camera. Figure 9 shows the extended view of the follower car, where the point cloud of both the FedEx and the sidewalk, obtained from the leader, can be perfectly merged into the follower's perspective.

Results: Bandwidth and Latency. Table 1 summarizes the bandwidth requirement of transferring different representations of vehicle surroundings with various granularities. Specifically, we consider the following four representations. The most fine-grained form is the full point cloud where the car can share everything it sees with other cars (Full). A more lightweight representation is the voxels belonging to dynamic objects (Dynamic). A comparably lightweight includes point clouds belonging to the objects detected by YOLO (Objects). The most coarse-grained form is the 3D bounding box and the label of the object (Label). The numbers in the table are the average point cloud data sizes evaluated over campus cruising traces with multiple moving vehicles in the scene.

AVR can potentially share these representations either via direct V2V communication, or indirectly through the road side units or the cloud. However, existing state-of-the-art wireless communication technologies cannot support any representation except coarse labels. Theoretically, DSRC / 802.11p [6] can achieve 3-27 Mbps, depending on modulation and error correction coding (ECC) rate. Both WiFi-direct (802.11n/ac) and LTE/LTE-direct [7] can achieve up to 300Mbps. To explore the transmission overhead in practice, we establish both a peer-to-peer (P2P) WiFi direct link and a server for LTE transmission over cloud, to measure the performance of point cloud transfer. Figure 10 shows the mean and variance of the throughput and latency for different point cloud sizes. As expected, the throughput generally increases as the file size increases because it allows the TCP congestion window to open up fully and utilize more bandwidth. The peak throughput reaches 60 Mbps after transmitting over 50 MB. What's interesting is that as we decrease the point cloud size in log scale, the latency drops only linearly. Clearly, for anything but labels, the latency of V2V communication is still inadequate, and we plan to explore this dimension in the future.

In terms of processing overhead, AVR executes two major components at runtime: localization and point cloud manipulation. ORB-SLAM requires 0.06sec per frame. The total point cloud manipulation time, including perspective transformation, merging, and 3D rendering, is on average 1.337sec per view. Our current implementation uses the CPU for this task. We plan to further optimize the computation using GPU for both localization and point cloud manipulation to accelerate the processing speed.

5 REMAINING CHALLENGES AND FUTURE WORK

We see several directions of future work. The first is to try to leverage other techniques adopted in latency sensitive applications [9, 11] like gaming and interactive VR, such as dead reckoning or motion prediction. Leveraging the trajectory of the vehicle, information from maps, and the constraints on pedestrians, we might be able to correctly predict motion over short time scales. Receivers can use these trajectories to update their local views, re-synchronizing occasionally. While compensating for latency, these kinds of techniques can also reduce bandwidth significantly. In addition, we also plan to explore advanced voxel compression techniques to address the bandwidth bottleneck. Further, we have developed our AVR prototype in a modular fashion, and plan to explore using other sensors such as LiDAR that can generate 3-D point clouds (in theory, the AVR framework should translate directly to these sensors), and more advanced object detection frameworks [10]. Finally, future work will need to address security and privacy concerns that may arise from communicating visual information between vehicles.

[1] In a standard openGL 3D viewer, users can rotate and zoom the point cloud to view different perspective. The paper only shows one vehicle perspective of the point cloud.

6 RELATED WORK

Prior research has explored position enhancement using differential GPS [15], inertial sensors on a smartphone [8, 16], onboard vehicle sensors [19], and WiFi and cellular signals [27]. Visual SLAM techniques have used monocular cameras [12], stereo camera [13], and lidar [17]. Kinect [24] can also produce high-quality 3D scan of an indoor environment using infrared. By contrast, AVR positions of vehicles in a common coordinate frame of reference using features computed by a visual SLAM technique. While autonomous driving is becoming a reality [1, 2], and ADAS systems such as lane maintenance and adaptive cruise control are already available in many cars, we are unaware of efforts to collaboratively share visual objects across vehicles in an effort to enhance these capabilities. Existing point cloud matching approaches [18, 22] involve heavy computation of three-dimensional similarity, while AVR adopts a lightweight approach exploiting pixel-voxel correspondence. Finally, AVR can leverage complementary approaches such as [20, 21] for content-centric methods to retrieve sensor information from nearby vehicles, or over cloud infrastructure.

7 CONCLUSION

In this paper, we have discussed a new capability, augmented vehicular reality, and have sketched the design and implementation of an AVR system. AVR is not only beneficial to human drivers as a driving assistant system, but also, more importantly, enables extended vision for autonomous driving cars to make better and safer decisions. With extended vision, we have demonstrated that AVR can effectively remove sensing range limitations and line-of-sight occlusions. Future work on AVR will focus on improving the throughput of processing objects, addressing the bandwidth constraints, and reducing latency.

REFERENCES

[1] *Google Self-Driving Car Project Monthly Report September 2016*. https://static.googleusercontent.com/media/www.google.com/en//selfdrivingcar/files/reports/report-0916.pdf.

[2] *Here's How Tesla's Autopilot Works*. http://www.businessinsider.com/how-teslas-autopilot-works-2016-7.

[3] *How Uber's First Self-Driving Car Works*. http://www.businessinsider.com/how-ubers-driverless-cars-work-2016-9.

[4] *NVidia Drive PX 2*. http://www.nvidia.com/object/drive-px.html.

[5] *ZED Stereo Camera*. https://www.stereolabs.com/.

[6] 2010. IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments. *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)* (July 2010), 1–51.

[7] David Astély, Erik Dahlman, Anders Furuskär, Ylva Jading, Magnus Lindström, and Stefan Parkvall. 2009. LTE: The Evolution of Mobile Broadband. *Comm. Mag.* 47, 4 (April 2009).

[8] Cheng Bo, Xiang-Yang Li, Taeho Jung, Xufei Mao, Yue Tao, and Lan Yao. 2013. SmartLoc: Push the Limit of the Inertial Sensor Based Metropolitan Localization Using Smartphone. In *Proceedings of the 19th Annual International Conference on Mobile Computing and Networking (MobiCom '13)*.

[9] Kevin Boos, David Chu, and Eduardo Cuervo. FlashBack: Immersive Virtual Reality on Mobile Devices via Rendering Memoization. In *Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '16)*.

[10] Zhaowei Cai, Quanfu Fan, Rogério Schmidt Feris, and Nuno Vasconcelos. 2016. A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection. In *Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part IV*. 354–370.

[11] Tiffany Yu-Han Chen, Lenin S. Ravindranath, Shuo Deng, Paramvir Victor Bahl, and Hari Balakrishnan. 2015. Glimpse: Continuous, Real-Time Object Recognition on Mobile Devices. In *13th ACM Conference on Embedded Networked Sensor Systems (SenSys)*. Seoul, South Korea.

[12] Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. 2007. MonoSLAM: Real-Time Single Camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.* 29, 6 (June 2007).

[13] J. Engel, J. Stueckler, and D. Cremers. 2015. Large-Scale Direct SLAM with Stereo Cameras.

[14] A. Gern, R. Moebus, and U. Franke. 2002. Vision-based lane recognition under adverse weather conditions using optical flow. In *Intelligent Vehicle Symposium, 2002. IEEE*, Vol. 2. 652–657 vol.2.

[15] Mahanth Gowda, Justin Manweiler, Ashutosh Dhekne, Romit Roy Choudhury, and Justin D. Weisz. 2016. Tracking Drone Orientation with Multiple GPS Receivers. In *Proceedings of the 22nd Annual International Conference on Mobile Computing and Networking (MobiCom '16)*.

[16] Santanu Guha, Kurt Plarre, Daniel Lissner, Somnath Mitra, Bhagavathy Krishna, Prabal Dutta, and Santosh Kumar. 2010. AutoWitness: Locating and Tracking Stolen Property While Tolerating GPS and Radio Outages. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems (SenSys '10)*.

[17] Wolfgang Hess, Damon Kohler, Holger Rapp, and Daniel Andor. 2016. Real-Time Loop Closure in 2D LIDAR SLAM. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*. 1271–1278.

[18] Jing Huang and Suya You. 2012. Point cloud matching based on 3D self-similarity. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2012 IEEE Computer Society Conference on*. IEEE, 41–48.

[19] Yurong Jiang, Hang Qiu, Matthew McCartney, Gaurav Sukhatme, Marco Gruteser, Fan Bai, Donald Grimm, and Ramesh Govindan. 2015. CARLOC: Precise Positioning of Automobiles. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15)*.

[20] Swarun Kumar, Shyamnath Gollakota, and Dina Katabi. 2012. A Cloud-assisted Design for Autonomous Driving. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing (MCC '12)*.

[21] Swarun Kumar, Lixin Shi, Nabeel Ahmed, Stephanie Gil, Dina Katabi, and Daniela Rus. 2012. CarSpeak: A Content-centric Network for Autonomous Driving. *SIGCOMM Comput. Commun. Rev.* 42, 4 (Aug. 2012).

[22] Mathieu Labbé and François Michaud. 2014. Online global loop closure detection for large-scale multi-session graph-based slam. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2661–2666.

[23] Raul MurArtal, J. M. M. Montiel, and Juan D. Tardos. 2015. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics* (2015).

[24] Richard A. Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew J. Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. 2011. KinectFusion: Real-time Dense Surface Mapping and Tracking. In *Proceedings of the 2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR '11)*.

[25] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. 2015. You Only Look Once: Unified, Real-Time Object Detection. *CoRR* abs/1506.02640 (2015). http://arxiv.org/abs/1506.02640

[26] Miguel Angel Sotelo, Francisco Javier Rodriguez, Luis Magdalena, Luis Miguel Bergasa, and Luciano Boquete. 2004. A Color Vision-Based Lane Tracking System for Autonomous Driving on Unmarked Roads. *Auton. Robots* 16, 1 (Jan. 2004), 95–116.

[27] Arvind Thiagarajan, Lenin Ravindranath, Katrina LaCurts, Samuel Madden, Hari Balakrishnan, Sivan Toledo, and Jakob Eriksson. 2009. VTrack: Accurate, Energy-aware Road Traffic Delay Estimation Using Mobile Phones. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*.

[28] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. 2006. Stanley: The Robot That Won the DARPA Grand Challenge: Research Articles. *J. Robot. Syst.* 23, 9 (Sept. 2006), 661–692.

[29] Chris Urmson, Joshua Anhalt, Drew Bagnell, Christopher Baker, Robert Bittner, M. N. Clark, John Dolan, Dave Duggins, Tugrul Galatali, Chris Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas M. Howard, Sascha Kolski, Alonzo Kelly, Maxim Likhachev, Matt McNaughton, Nick Miller, Kevin Peterson, Brian Pilnick, Raj Rajkumar, Paul Rybski, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod Snider, Anthony Stentz, William "Red" Whittaker, Ziv Wolkowicki, Jason Ziglar, Hong Bae, Thomas Brown, Daniel Demitrish, Bakhtiar Litkouhi, Jim Nickolaou, Varsha Sadekar, Wende Zhang, Joshua Struble, Michael Taylor, Michael Darms, and Dave Ferguson. 2008. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *J. Field Robot.* 25, 8 (Aug. 2008), 425–466.