Augmented Vehicular Reality: Enabling Extended Vision for Future Vehicles

Hang Qiu, Fawad Ahmad, Ramesh Govindan, Marco Gruteser, Fan Bai, Gorkem Kar

Abstract
As autonomous cars do today, future vehicles will have rich sensors to map and identify objects in the environment. For example, many autonomous cars 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 occlusion, sensing range, extreme weather, and lighting conditions. To improve visibility, and therefore safety, 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 highlight research challenges in achieving AVR’s potential to improve autonomous vehicles and ADAS.

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 minimize accidents, 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 safe 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 merging visual information between nearby cars. This would augment vehicular visibility into hazards, and enable autonomous vehicles to avoid accidents, or ADAS technologies to guide human users in making safe driving decisions. This capability, that we call Augmented Vehicular Reality (AVR), aims to combine emerging vision technologies that are being developed specifically for vehicles [21], 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, aggressively reducing the bandwidth and latency by leveraging techniques such as compression and motion prediction. Our design builds on prior work in localization 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.

Google self-driving car [7] mainly relies on an expensive LiDar system including a 64-beam Velodyne LiDar, in addition to radar and camera sensors, which produce a high-definition (HD) map accurate to within 1 inch. 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.. Tesla has developed a semi-autonomous system, AutoPilot [10], whose sensors include a forward radar, a forward-looking camera, and 12 long-range ultrasonic sensors around the car. Uber self-driving taxis [12] have a similar LiDar system as Google’s, and is also equipped with 20 cameras.

Abstractly, regardless of the details, this collection of sensors, generates successive point clouds, each which represents
the instantaneous 3D view of the environment. A point in the point cloud represents a voxel in the 3D space, which is associated with a position in 3-D and may or may not have a color attribute depending on the sensor used. For example, the 64-beam Velodyne LiDAR can collect point clouds at 10 Hz with in total 2.2 million points per second in 360 degrees. 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 degrees).

**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\(^1\). 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 provide, to the follower, those objects 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 decisions and safe driving. Other 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.

**Challenges.** To provide vehicles with AVR, there are several fundamental challenges to address, some of which we discuss in this paper, and others which we defer to future work \(\S 5\).

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 \[21\], 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.

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 3-D 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.
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 a 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.

**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, [19]). 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 C can correctly overlay C’s shared view in its own.

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 is doing a 3D scan of the street. A feature is considered stable only when its absolute position remains 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, the 3-D map 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 of our campus and the localized camera positions as it travels.

### 3.1 Extending the Vision

With the help of a common world static HD map developed above, 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) with car B, AVR needs to transform the objects or point cloud in one vehicle’s local view to another vehicle’s. Vehicles have very different perspectives of the world depending on different location and orientation of their sensors. Note that unlike the features, the point cloud is generated in the local camera coordinate frame without any knowledge of the world at all. In this section, we describe how AVR achieves this.

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

$$T_{cw} = \begin{bmatrix}
\text{Rot}_X.x & \text{Rot}_Y.x & \text{Rot}_Z.x & \text{Translation}.x \\
\text{Rot}_X.y & \text{Rot}_Y.y & \text{Rot}_Z.y & \text{Translation}.y \\
\text{Rot}_X.z & \text{Rot}_Y.z & \text{Rot}_Z.z & \text{Translation}.z \\
0 & 0 & 0 & 1
\end{bmatrix} \tag{1}
$$

The translation is equivalent to the camera coordinate in the world domain, and the rotation matrix indicates the rotation of the camera coordinate frame against the world 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).
we use this information).

whether it’s possible to isolate only dynamic objects in the

3.2 Dynamic Objects Detection and Isolation

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’s 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 [13, 18] 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 neighbor-
ing 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 (§3.1), the homography matrix $H$, except 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.
Such matching among pixels indicates the matching
among the corresponding voxels in the point cloud. Therefore,
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). Since the two frames are immediate
neighbors to each other, AVR can approximate and acceler-
ate the computation by adding the translation vector directly
while omitting the rotation matrix.

One optimization we have experimented with is to exploit
object detectors [22] to narrow the space for moving objects
(cars, pedestrians). Once we have identified the pixels asso-
ciated with these moving objects, we can identify if the
 correponding 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 [24] (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 [22], a
state-of-the-art object detector. For the SLAM algorithm,
AVR uses ORB-SLAM2 [19], currently the highest ranked
open source visual odometry algorithm on KITTI benchmark,
which is 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 establish the
map as well as realizing the extended vision.

Results: Extended Vision. Figure 7 shows the point cloud
view 3 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 exten-
sion 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 representa-
tions 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

\[
\begin{bmatrix}
  x' \\
  y' \\
  z' \\
  1
\end{bmatrix} = T_{cw} * \begin{bmatrix}
  x \\
  y \\
  z \\
  1
\end{bmatrix}
\]

Assuming camera $A$ has a pose of $T_{aw}$, and camera $B$
has a pose of $T_{bw}$, similarly, transforming a voxel $V_a$
from camera $A$ to $X_b$ in camera $B$ follows Equation (3).

\[
V_b = T_{bw}^{-1} * T_{aw} * V_a
\]

\[\text{Figure 4—Finding the Homography of the Current Frame in the Previous Frame using SURF Features}\]

\[\text{Figure 5—ZED Stereo Camera}\]

\[\text{Figure 6—ZED Mounted on top of the Windshield with Smartphone Attached}\]

\[\text{Table 1 summarizes the bandwidth and latency}\]

\[\text{while omitting the rotation matrix.}\]

\[\text{One optimization we have experimented with is to exploit}\]

\[\text{object detectors [22] to narrow the space for moving objects}\]

\[\text{(cars, pedestrians). Once we have identified the pixels asso-}\]

\[\text{ciated with these moving objects, we can identify if the}\]

\[\text{corresponding voxels are moving, thereby reducing the search}\]

\[\text{space for moving objects. We evaluate how much bandwidth}\]

\[\text{this optimization can save in the following section §4.}\]

\[\text{4. PRELIMINARY EVALUATION}\]

\[\text{Experimental Setup. We experiment and evaluate AVR on}\]

\[\text{two cars, each equipped with one ZED [24] (Figure 5) stereo}\]

\[\text{camera mounted on the top of the front windshield and a}\]

\[\text{smartphone attached to it (Figure 6). The stereo camera}\]

\[\text{records the video stream and computes the 3D point cloud as}\]

\[\text{well as depth information, while the mobile phone records}\]

\[\text{GPS and all motion sensors, i.e., gyroscope, accelerometer,}\]

\[\text{magnetometer, etc.}. ZED can create the real-time point cloud}\]

\[\text{at a framerate of 30fps on a Titan X GPU with a resolution}\]

\[\text{of 720P (1280 x 720), and up to 100 fps with VGA (640 x 480)}.\]

\[\text{For object recognition, AVR uses YOLO [22], a}\]

\[\text{state-of-the-art object detector. For the SLAM algorithm,}\]

\[\text{AVR uses ORB-SLAM2 [19], currently the highest ranked}\]

\[\text{open source visual odometry algorithm on KITTI benchmark,}\]

\[\text{which is a widely accepted vehicle vision benchmark. Using}\]

\[\text{ORB-SLAM2, we have collected several traces on and around}\]

\[\text{campus with one car following the other to both establish the}\]

\[\text{map as well as realizing the extended vision.}\]

\[\text{Results: Extended Vision. Figure 7 shows the point cloud}\]

\[\text{view 3 of the leader, where there is FedEx truck parked on}\]

\[\text{the left and a segment of sidewalk on the right. Figure 8 shows}\]

\[\text{the perspective of the follower where a white sedan is parked}\]

\[\text{on the left. Note that neither the FedEx truck or the exten-}\]

\[\text{sion of the sidewalk can be fully observed due to the limited}\]

\[\text{sensing range of the ZED stereo camera. Figure 9 shows the}\]

\[\text{extended view of the follower car, where the point cloud of}\]

\[\text{both the FedEx and the sidewalk, obtained from the leader,}\]

\[\text{can be perfectly merged into the follower’s perspective.}\]

\[\text{Results: Bandwidth and Latency. Table 1 summarizes the}\]

\[\text{bandwidth requirement of transferring different representa-}\]

\[\text{tions of vehicle surroundings with various granularities.}\]

\[\text{Specifically, we consider the following four representations.}\]

\[\text{The most fine-grained form is the full point cloud where the}\]

\[\text{In a standard openGL 3D viewer, users can rotate and zoom the}\]

\[\text{point cloud to view different perspective. The paper only shows one}\]

\[\text{vehicle perspective of the point cloud.}\]
Figure 7—Point Cloud of Leader

Figure 8—Point Cloud of Follower

Figure 9—Extended Point Cloud

Table 1—Point Cloud Data Size Per Frame.

<table>
<thead>
<tr>
<th></th>
<th>VGA (640 x 480)</th>
<th>720P (1280 x 720)</th>
<th>1080P (1920 x 1080)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Full</td>
<td>4.91 MB</td>
<td>14.75 MB</td>
<td>33.18 MB</td>
</tr>
<tr>
<td>Dynamic</td>
<td>0.79 MB</td>
<td>2.36 MB</td>
<td>5.30 MB</td>
</tr>
<tr>
<td>Object</td>
<td>0.33 MB</td>
<td>0.98 MB</td>
<td>2.21 MB</td>
</tr>
<tr>
<td>Labels</td>
<td>0.05 MB</td>
<td>0.05 MB</td>
<td>0.05 MB</td>
</tr>
</tbody>
</table>

To explore the transmission overhead in reality, we establish a peer-to-peer (P2P) WiFi direct link 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 60Mbps after transmitting over 50MB. 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.

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 [2, 4] 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. Finally, 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 [3].

6. RELATED WORK

Several prior work has explored position enhancement using differential GPS [8], inertial sensors on a smartphone [1, 9], onboard vehicle sensors [15], and WiFi and cellular signals [23]. Visual SLAM techniques have used monocular cameras [5], stereo camera [6], and lidar [11]. Kinect [20] can also produce high-quality 3D scan of an indoor environment using infrared. By contrast, AVR does relative positioning of two vehicles in a common coordinate frame of reference using features computed by a visual SLAM technique. While autonomous driving is becoming a reality [7, 10], 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 [13, 18] 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 [17, 16] for content-centric methods to retrieve sensor information from nearby vehicles, or over cloud infrastructure.

7. CONCLUSION

In this paper, we present the concept of augmented vehicular reality and the design and implementation of 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


