Sensor-Fusion Aided Simulation of Scenarios for Autonomous Vehicles

Romil Bhardwaj
UC Berkeley
romil.bhardwaj@berkeley.edu

Sukrit Kalra
UC Berkeley
sukrit.kalra@berkeley.edu

Abstract

Autonomous vehicles (AVs) will be required to drive across a plethora of complex scenarios, many of which present unique challenges to the computational pipeline underlying the vehicle. The ability to develop and test future updates to the pipeline on such scenarios is critical in ensuring the safety of the pipeline. In order to incorporate production scenarios into the testing cycle, the vendors need to be able to reconstruct an accurate top-down representation of the environment around the vehicle using data collected from various sensors attached to the vehicle.

In this work, we present a preliminary approach that utilizes camera images along with point clouds generated by a LiDAR instance in order to construct a top-down representation of the environment around the vehicle. Furthermore, we utilize a series of such representations to estimate the movement of other agents in the environment over time and reconstruct their trajectories in a simulated environment.

1. Introduction

Innovations in hardware, algorithms and machine learning models have transformed AVs from a distant dream to a fast approaching reality. While these advancements have already facilitated the deployment of various "Level 1" and "Level 2" driver-assist features that allow modern vehicles to autonomously navigate freeways, perform lane changes and even respond to traffic lights, constant human supervision is still required to ensure safety of the vehicle. The ultimate goal of AV research, however, has been a "Level 4" or "Level 5" vehicle that is able to completely monitor the driving environment and drive itself under any circumstance, and recent regulatory approvals are poised to enable the deployment of such vehicles on public roads [6, 4, 9]. This complete removal of humans from the driving loop requires the vehicle to ensure safety across the long-tail of complex driving scenarios encountered in production [14, 11].

To achieve these levels of automation, which require the vehicle to perform all driving functions without human attention [1], state-of-the-art AVs employ a modular computational pipeline architecture. Ensuring *safety* of the AV mandates that each module of this pipeline provides the highest accuracy result possible within a given latency. In order to ensure high accuracy, vendors need to be able to capture the driving scenarios encountered in production, and utilize them to debug their components and develop and test future updates to each component [7].

Most vendors resort to logging of data generated from sensors such as cameras, LiDARs, GPS and IMU modules in order to drive their development and testing in simulation [8, 7]. While this collected data enables seamless testing of the *perception* and *localization* modules (see §2) that compute their results directly on the raw data from these sensors, downstream modules such as *prediction* and *planning* require an intermediate representation of the environment around the vehicle constructed by the upstream components. This representation (visualized in Fig. 1) is critical in ensuring the safety of the vehicle since it provides the current and past estimates of the locations of all obstacles around the AV. Moreover, any changes in the results from the *prediction* and *planning* module lead to a variation in the trajectory of the vehicle, which renders any future representation generated from the collected data incorrect.

In this work, we provide a framework that enables the reconstruction of the environment in a state-of-the-art driving simulator [12] using fusion of the sensor data collected from an AV. Specifically, our framework fuses data from multiple camera angles in order to construct a complete top-down environment representation, and uses the LiDAR data to ensure the accuracy of the scale of the obstacles. While previous work [13] has attempted to tackle this problem, it assumes a fixed size of the various types of objects in the scene, and does not reconstruct the environment in a simulator. This paper makes the following contributions:

- We enable an accurate environment representation reconstruction by fusing sensor data collected in an AV.
- We utilize this representation to reconstruct the scenario in the CARLA driving simulator [12].
- We provide a list of future works enabled by fusing real-world data with a goal to recreating scenarios.

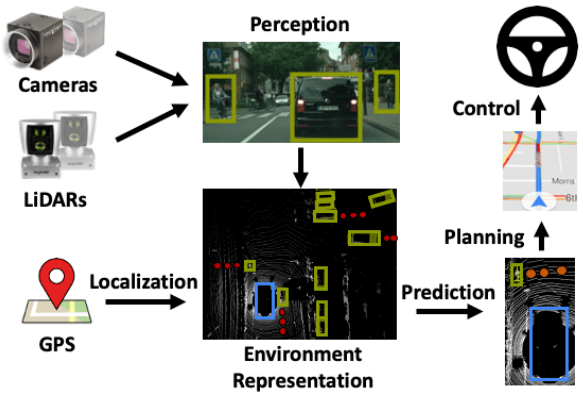

Figure 1. The computational pipeline underlying an AV. Data from sensors is used by the *perception* module to create a representation of the environment around the vehicle. This representation, along with the future trajectories predicted by the *prediction* module, is used to plan a future path for the AV by the *planning* module.

2. Background

A modern AV is equipped with multiple instances of various sensors such as cameras, LiDARs, radars, GPS, etc. which generate data at different frequencies [2, 5, 3]. This data is fused by a computational pipeline (Fig. 1) to perceive the environment around the vehicle, and generate control commands for the underlying machinery. The computation that allows the pipeline to achieve its goal is split across the following five modules that collectively ensure that an AV reaches its destination while following traffic laws and preventing collisions with other agents in the environment:

Localization. Similar to any robot, the first task of an AV is to compute its location in the world with respect to certain pre-defined fixed points. These static points of reference (such as buildings, traffic lights etc.) are represented in a high-definition map (HDMap), which provides the vehicle with a decimeter-level accurate mapping of the environment around it. In order to create the *environment representation*, the localization module fuses data from multiple sensors such as GNSS, IMU and LiDAR and uses it to triangulate a highly-accurate location of the vehicle in this map.

Perception. In order to ensure *safety*, this *representation* needs to be augmented with an accurate and real-time location of the obstacles around the AV. Any instance of missed or misclassified obstacles along with obstacles that are detected late either lead to a collision or require emergency maneuvers, which affect the comfort of the passengers. Moreover, for obstacles that move over time such as vehicles and pedestrians, called *agents*, the representation must also correlate each instance across sensor readings. This allows the representation to provide information about the speed of each agent along with the direction of its movement, which aids the prediction of its future trajectory.

Thus, the goal of the *perception* module is to synchronize data from sensors such as cameras, LiDARs and radars in order to accurately detect obstacles and track their movement through time. To achieve this, a perception module utilizes two components: (*i*) *Object detection*, which applies machine learning models on data retrieved from sensors in order to detect obstacles in the environment, and (*ii*) *Object tracking*, which utilizes algorithms and machine learning models in order to assign a unique identifier to each detected obstacle. This identifier can be used by downstream operators to query the *environment representation* in order to retrieve each obstacle's past locations and calculate their speed and trajectory.

Prediction. Once the perception module has finished building the *critical* environment representation, the pipeline needs to predict the future trajectories of each agent in order to ensure that the future motion of the AV does not collide with any obstacle. To achieve this task, the *prediction* module utilizes machine learning models that take as input the environment representation, and provide a set of predicted trajectories for each obstacle in the representation along with the probability of the obstacle taking that trajectory. Moreover, it is imperative to note that while the short term trajectory predictions might be accurate, the accuracy of the trajectory decreases with an increase in the prediction horizon. As a result, while a highly accurate prediction model can marginally offset the effects of an increased latency in building the representation, it is still desirable to ensure that the accuracy of the representation itself is maximized in the context of the driving environment. This allows the prediction module to produce accurate future trajectories instead of compensating for the latency of its upstream components, hence increasing *safety*.

Planning. While the HDMap provides high-level route instructions, it is the task of the *planning* module to compute low-level trajectory "waypoints" that the *control* module can follow. To achieve this, the planning module relies on algorithms that require as input a representation of the environment containing the locations of obstacles, traffic lights, signs etc., along with the future trajectories of all agents in the environment. The algorithm then computes a *fine-grained* motion plan (with decimeter-level waypoints) that seeks to ensure the safety of all agents in the environment, and the comfort of the passengers inside the AV.

Control. The *control* module computes the steering, acceleration and braking commands for the underlying machinery from the waypoints generated by the planning module.

Thus, a critical component that enables accurate testing of the *prediction* and *planning* components is the *environment representation*, which is a top-down representation of the environment around the AV and depicts the other agents present around the AV. An accurate representation represented in a dynamic world simulator allows vendors to explore the effects of changes to algorithms and models.

3. Design

In this section, we discuss the design of our framework that enables the construction of an *environment representation*. We start by discussing an initial straw man approach to solving the problem (§3.1). §3.2 then discusses the limitations of this approach, and §3.3 provides further refinements that constitute the design of our framework.

3.1. Straw Man Approach

The first step in computing a top-down representation of the environment from a camera image involves detecting the objects in the scene. Our straw man approach utilizes off-the-shelf object detection models such as Faster-RCNN [16], SSD [15] and the EfficientDet family of models [17]. to infer a bounding box in the camera frame for the object along with a label such as *person* or *vehicle*.

In order to map these boxes to their appropriate locations in the top-down representation, we require both the width and the height of each object. Similar to previous work [13], our straw man approach utilizes an average height and width for both *persons* and *vehicles* gathered from the National Center for Health Statistics and Consumer Reports respectively. This provides us with the dimensions of the top-down bounding box for each detected object in the scene.

Finally, our straw man approach assumes a straight linear surface and computes the depth of each obstacle from the AV, along with the distance between each obstacle using the current location of the vehicle and the camera calibration matrix computed from the location of the camera with respect to the vehicle. These distances, coupled with the dimensions of each obstacle retrieved earlier, allows us to compute a homography for each frame in order to compute the top-down representation of that frame.

3.2. Limitations of the straw man approach

While this straw man approach enables the construction of a top-down representation of the environment around the AV, it suffers from the following drawbacks owing to its strict assumptions about the environment and the obstacles:

- The heterogeneity of the obstacles is not represented by the average dimensions chosen by the straw man approach. As a result, the scale of the objects is not maintained.
- The assumption of a straight road to find the distance of the obstacles is limiting and does not work across the different road surfaces experienced during regular driving.
- The per-frame homography computation produces jittery top-down representations across time.

While the jittery representation complicates the generation of the trajectory of the obstacles, the assumptions about the dimensions and distance of the obstacles leads to an incorrect simulation, since any change in the scenario prevents the vendors from ensuring that their updates perform well.

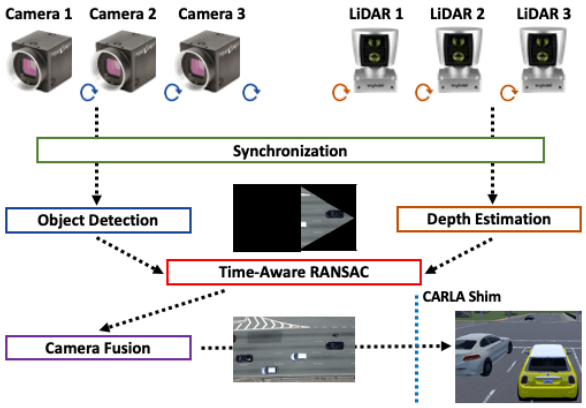

Figure 2. The architecture of our solution that fuses data from multiple sensors (including cameras, LiDARs and GPS modules) in order to create a top-down representation of the environment. This representation is used to determine trajectories for all the agents in the scene, which is then recreated in the CARLA simulator.

3.3. Our Approach

We now discuss our design that addresses the limitations of the straw man approach discussed in §3.1 (depicted in Fig. 2). The key observation in our approach is the utilization of the multiple sensors present in an AV. Specifically, we utilize data from the multiple cameras, LiDARs and GPS modules in order to accurately detect the dimensions and distance of the obstacles in the scene. We introduce *time-aware RANSAC*, which enables the construction of a stable top-down representation of the environment over time. This aids in the reconstruction of the trajectories of the obstacles over time, which is then replayed in the CARLA simulator [12]. The remainder of the section discusses the different parts of the design in detail.

Synchronization. As discussed in §2, an AV is equipped with multiple instances of various sensors such as cameras, LiDARs and GPS modules. While this allows the AV to capture the entire environment around it, these sensors run at different frequencies which complicates the process of fusing data from these sensors in order to build the *environment representation*. Taking the most recent data from all the sensors at the frequency of the slowest sensor hinders both the accuracy and the latency of the fusion module. For example, running at the frequency of the LiDAR sensor, the latest camera image available might not contain the objects in the same location as they are in the LiDAR reading.

In our design, we run at the frequency of the camera, and utilize the periodicity of the sensors to wait if the incoming LiDAR reading is closer in time to the camera image than the last available one. In order to ensure the correct fusion of this data, we transform the LiDAR data into the camera frame. To achieve this, we first transform the LiDAR data to the frame of the vehicle based on the location of the LiDAR

with respect to the center of the vehicle. This data is further transformed to the camera frame by applying the transform matrix with respect to the front-facing camera.

Object Detection. Similar to the straw man approach, we utilize off-the-shelf object detection models such as Faster-RCNN [16], SSD [15] and the EfficientDet family [17].

Depth Estimation. A major limitation of the straw man approach that affects the accuracy of the *environment representation* is the assumption of the height and dimensions of the obstacle according to the label assigned by the object detector. In our design, we utilize the synchronized and transformed point cloud generated from the multiple instances of the LiDAR sensor. Specifically, we find all the points of the point cloud that lie inside the bounding box generated by the object detector. We then calculate the height and width of the obstacle by subtracting the distance between the lowest and the highest point that lies in the bounding box.

Our approach also utilizes the synchronized and transformed LiDAR data to calculate the distance of each obstacle from the AV. Specifically, we assign to each obstacle in the scene a depth equivalent to the distance of the point closest to the middle point of the bounding box from the pointcloud. This allows our design to forego the assumption of a straight road, and work across different road surfaces.

Time-Aware RANSAC. In order to calculate the homography that converts the images from the camera frame to the top-down environment representation frame, the straw man approach ran RANSAC across the bounding boxes detected for each frame. This approach minimizes the error across a frame, but leads to a jittery representation over time since each frame is transformed with a different homography. A jittery top-down representation complicates an accurate calculation of the trajectory of each obstacle in the scene.

Our design introduces *time-aware RANSAC* that seeks to minimize the jitter caused due to the homography transformation across time. Time-aware RANSAC calculates the homography by executing RANSAC across both the obstacle and the time dimension. Specifically, we calculate the homography by choosing a fixed number of points from a single frame. The homography with the minimum error is chosen by executing RANSAC with the homography across all the frames in the dataset. While this approach minimizes the error, it is computationally expensive. As an optimization, our implementation chooses the homography with the minimum error by executing RANSAC across all the obstacles in the scene. However, to prevent jitter, the homography with the minimum error across time is chosen as the single homography for the video stream. In order to calculate this homography, we execute RANSAC across the minimum computed homography of each frame.

Camera Fusion. In order to build a complete representation of the environment around the AV, we utilize images captured from cameras located at different parts of the vehicle. To fuse the images together, we utilize the static locations of the camera with respect to the vehicle to calculate the extrinsic and intrinsic matrices of all the cameras. Further, every image is projected on the frame of the center camera in order to retrieve the complete top-down representation of the scene. This provides us with a panoramic view of the entire area captured by the cameras.

CARLA Shim. Once the representation of the environment is complete, we use the current location of the vehicle (retrieved from fusing the GPS and IMU sensors) to calculate the location of each obstacle in every frame. These points are then transformed to corresponding waypoints on a CARLA map with a predefined starting point for the AV. We developed a simple waypoint follower control algorithm that generates steering and acceleration commands for the vehicles by applying PID control to the given trajectory. We also match the size and label of the obstacle to the corresponding vehicle in CARLA. However, we are limited by the number of vehicles in CARLA, and cannot ensure that the dimensions of the real vehicles remain unaltered.

4. Results

To evaluate our methodology, we chose the Waymo Open Dataset [10], which provides images from multiple camera angles along with the LiDAR and GPS information required for the precise simulation of the scenario in the CARLA simulator. Fig. 3 shows the results of each step in our pipeline on a test scenario from the dataset. Our project website contains the videos of the time-aware RANSAC on multiple scenarios, along with a simulation of this particular scenario in CARLA.

5. Limitations & Future Work

The current implementation of our work suffers from the following limitations:

- Lane detection is aided by a top-down representation and it would be useful to evaluate the efficacy of lane detection on the representation generated by our system.
- Our approach does not yet do detection of colors, and we randomize the color of the vehicles in the simulator. It would be useful to maintain the color in the simulation.
- It would be useful to fit a curve to the waypoints generated from the top-down representation in order to smoothen the trajectory of the vehicle in simulation. The low sampling frequency of the cameras hinders faithful reproduction of the scenario.
- Our current approach executes time-aware RANSAC across all the camera angles, which is computationally expensive, and might lead to blurring. It would be useful to fuse the images together and then run RANSAC.

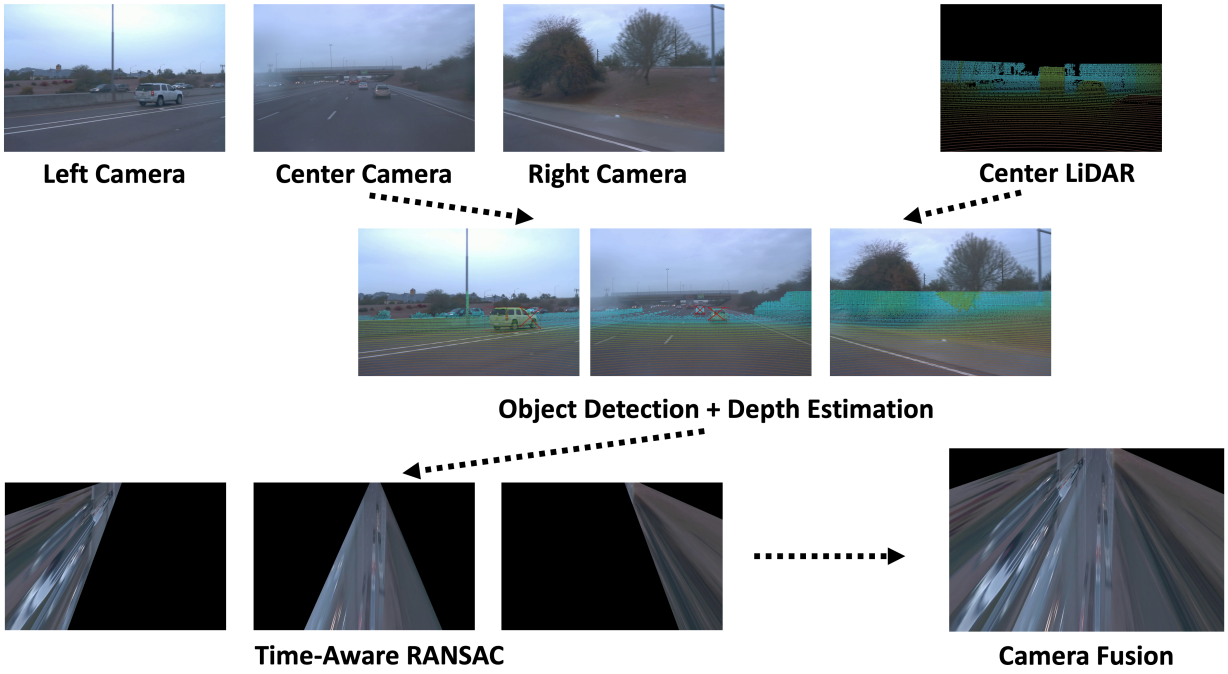

Figure 3. The results of each step in our pipeline on a set of three images captured from the Waymo Dataset [10].

References

[1] Automated vehicles for safety. https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

[2] How does a self-driving car see? https://blogs.nvidia.com/blog/2019/04/15/how-does-a-self-driving-car-see/.

[3] How uber self-driving cars see the world. https://www.therobotreport.com/how-uber-self-driving-cars-see-world/.

[4] Hyundai-backed motional to launch fully driverless cars in las vegas. https://tinyurl.com/yytftr2h.

[5] Introducing the 5th generation waymo driver. https://blog.waymo.com/2020/03/introducing-5th-generation-waymo-driver.html.

[6] Robotaxi companies can now win approval to operate in california. https://tinyurl.com/yavxgxjc.

[7] Tesla is collecting insane amount of data from its full self-driving test fleet. https://tinyurl.com/y89gtu9j.

[8] Waymo offers a peek into the huge trove of data collected by its self-driving cars. https://tinyurl.com/y59oneag.

[9] Waymo restarts robotaxi service without human safety drivers. https://tinyurl.com/ycrcsqhu.

[10] Waymo open dataset: An autonomous driving dataset, 2019.

[11] D. Anguelov. Taming the long tail of autonomous driving challenges. https://www.youtube.com/watch?v=Q0nGo2-y0xY, 2019.

[12] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Conference on Robot Learning (CoRL)*, pages 1–16, 2017.

[13] D. Hoiem, A. A. Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15, 2008.

[14] A. Karpathy. Cvpr '20 - workshop on scalability in autonomous driving. https://sites.google.com/view/cvpr20-scalability/archived-talks/keynotes, 2020.

[15] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In *Proceedings of the 14th European Conference on Computer Vision (ECCV)*, pages 21–37. Springer, 2016.

[16] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[17] M. Tan, R. Pang, and Q. V. Le. Efficientdet: Scalable and efficient object detection (v4). In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.