Sensor Fusion for Improved Object Detection Performance of Toyota's Autonomous Vehicles

The Toyota Research Institute (TRI) was established to advance research on artificial intelligence, robotics, and autonomous vehicle technology. TRI plays a crucial role in developing Toyota's autonomous vehicle technologies. My project is directed at aiding TRI's goal of moving towards fully autonomous driving, by implementing a data-level approach to camera-LiDAR fusion, improving object detection accuracy. I will motivate and explain my approach below:

The goal of fully autonomous driving can only be realized by fulfilling these two criteria: cars should have a reliable perception of their surroundings, and they should have a reliable way of making decisions based on their perceptions. I will focus on the first criterion. A car's perception of its surroundings is often measured by the accuracy of its object detection capabilities. Object detection accuracy has improved substantially over the past few years, however, there is still significant room for improvement, given that state-of-the-art models achieve an Average Precision (measured by bounding boxes overlapping by ≥ 50%) of about 90% across all classes of objects.

With the increased availability of LiDAR (Light Detection and Ranging) sensors, a type of sensor that accurately determines distances to objects across its surroundings, there arose a need to identify which sensor should be used for object detection, with most concluding that a fusion of LiDAR and camera input data would be ideal for maximizing object detection accuracy.

The question on the minds of many researchers, though, is *how* to best fuse the LiDAR and camera input data. As Xiang et al.[1] outlined, there are three levels at which different input modalities can be fused with each other: data, feature, and result. The data level refers to utilizing the raw input data, the feature level refers to utilizing extracted features from the raw data, and the result level refers to utilizing the outputs of the object detection of each separate input.

Given the above terminology as a foundation, many different approaches can, and have, been posed. Fusion can be performed across different levels of representation, or the same level. An example of fusion across differing levels of representation is fusing feature-level LiDAR input with data-level camera input. A popular example of fusion across the same level of representation is fusing feature-level LiDAR input with feature-level camera input. Some advantages of this approach are that you can benefit from a fusion of only the important features of both input modalities, and that you are dealing with only a subset of the total input size, which is useful when dealing with large quantities of data.

My approach for this project (my feature selection), though, is data-level fusion: combining data-level LiDAR input with data-level camera input. Despite the above outlined advantages of

feature-level fusion, data-level fusion has the quality of retaining the entirety of the both inputs' data, rather than reducing each input to features that may result in lost information.

The dataset I am using is the KITTI object detection dataset, due to the fact that it includes high quality and diverse driving conditions, with a car that has both camera and LiDAR sensors. It's worth noting that raw LiDAR input is called a point cloud, and looks like an assortment of points in 3D space that each represent the laser reaching that object.

The way I chose to implement my feature selection approach is by aligning the LiDAR point cloud with the camera image (using KITTI's provided calibration data), overlaying the LiDAR distance information over the raw camera frame (this is visualized in the ipynb file). This allows the CNN model to benefit from the color, shape, and texture information of the camera input, along with having available to it the distance information from the LIiDAR input.

I chose a multimodal version of the PointNet[2] model, since this model is effective at making good use of LiDAR data, and with some added layers, it can handle camera images as well. Due to the scope of this project, I only implemented multilabel binary classification as the object detection result, not 3D bounding boxes. This means that for each of the 9 classes of objects: 'Car', 'Cyclist', 'Person_sitting', 'Van', 'Pedestrian', 'Misc', 'DontCare', 'Truck', 'Tram', the model outputs a length-9 array of zeros and ones to indicate whether or not it detects each of these classes in the provided input.

I trained the model on 1500 LiDAR+camera inputs, and used an on-the-fly data generator to process batches of LiDAR and image data so as to not overwhelm the GPU. I used batch sizes of 4, 8, and 16, and found that 8 produced the most accurate and stable results. I used a differing number of epochs and found that 10 epochs was effecting at training the model, but not to the point of overfitting. I subsampled the LiDAR points to reduce the load on the GPU and found that 10,000 points (out of an average of 120,000 total points) effectively saved on memory while not compromising on information too much.

The results of the multimodal object detection accuracy was 86%, though it seemed like my model was underfitting, as evidenced by the training loss and volatile validation loss. In the future, I would utilize a more robust model, as it seemed that the model I used — which I simplified in order to not overwhelm my GPU memory — was not robust enough to capture enough of the patterns present in the data. Additionally, with more resources at my disposal, I would train the model on a larger portion of the total dataset, as it seemed like 1500 instances may not have been enough.

I implemented the PointNet model on LiDAR data alone, in order to see how my fusion technique compared to simply using LiDAR data on a model optimized for LiDAR data. A key aspect of this setup was that the model voxelized the LiDAR inputs — converted each point cloud from a set of continuous points into discrete pixels, joined into larger masses. This is a more meaningful representation that was not implemented in my fusion approach since I was performing fusion at the data-level, not feature-level.

The result of the LiDAR only model was a similar accuracy of ~86%, but with more consistent improvements in training, and inference results that suggested the model was learning more nuanced patterns present in the dataset.

I see this result as a proof against my initial assertion. Seemingly, a less all-inclusive, but more focused representation of the sensor inputs can allow CNN models to learn more effectively, rather than be distracted by inconsequential elements of the data. However, this might be the case here due to my environment not having a high enough computational power. In the future, I would utilize a more powerful GPU on a more robust version of my model (perhaps even utilize PointNet++[3]), and see if the model starts to pick up on more patterns. If that fails, I would be very willing to admit that feature-level fusion *is* the right approach to camera-LiDAR fusion in most object detection contexts. This would influence whether or not I'd recommend TRI to use my model. Either way, though, I think this process is valuable in revealing what types of sensor fusion are superior, and what it is that data-level fusion has/lacks in comparison to feature-level fusion. These insights could potentially influence future decisions at TRI.

If successful, this model would be deployed to detect objects in semi-autonomous/fully-autonomous vehicles, which would serve as the backbone for a model that makes decisions based on its perception of its surroundings.

References:

[1] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

[2]C. Qi, Hao Su, Kaichun Mo, and L. Guibas. PointNet: Deep learning on point sets for 3d classification and segmentation. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[3]Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Neural Information Processing Systems (NeurIPS), 2017.