Camera-Radar Sensor Fusion for Depth Estimation

Lonnie D. Chien, '23

Follow this and additional works at: https://works.swarthmore.edu/theses

Part of the Engineering Commons

Recommended Citation
https://works.swarthmore.edu/theses/281

This work is licensed under a Creative Commons Attribution 4.0 International License.
This work is brought to you for free by Swarthmore College Libraries' Works. It has been accepted for inclusion in Senior Theses, Projects, and Awards by an authorized administrator of Works. For more information, please contact myworks@swarthmore.edu.
ENGR 090: Senior Design Project

Camera-Radar Sensor Fusion for Depth Estimation

Lonnie Chien

Faculty Advisor: Stephen Phillips

May 5, 2023

Swarthmore College

Department of Engineering
Abstract

Self-driving cars rely on sensors for their perception of the surrounding environment. Depth estimation provides crucial information for the control systems of autonomous vehicles, as avoiding collisions and accidents is impossible without knowledge of the 3-dimensional locations of other objects on the road. Monocular depth estimation, or depth estimation from a single camera, shows promise as a relatively inexpensive yet effective solution to the depth estimation task. Cameras, however, are limited by the image formation process in their ability to calculate 3-dimensional depth from 2-dimensional images. Traditional machine-learning models use images paired with ground-truth depth labels for training. The image-label pairs can be difficult to obtain, which calls for the development of unsupervised machine-learning models. Sensor fusion, such as supplementing images with radar, also has the potential to improve monocular depth models. For the E90 project, we investigate the effectiveness of unsupervised machine learning models and sensor fusion in improving monocular depth estimation for self-driving cars.
5.1. Radar Background Information

5.2. Improvements to Model with Radar

6. Satisfaction of ABET Criteria for Design

6.1. What constraints govern your design problem?

6.2. What requirements will you (did you) develop?

6.3. How will you (or show that you did) evaluate your solution against the requirements?

6.4. What professional standards and codes, if any, govern your design (and if none, state why or that you were unable to find any).

8. Acknowledgements

9. References…
1. Background and Motivations

Autonomous vehicles, or self-driving cars, are able to navigate roads without requiring any input from a human driver. Several self-driving vehicles are already available on the market as of 2023, ranging from small delivery vehicles such as those developed by Nuro to sedans and personal, family, or luxury vehicles such as those developed by universally recognized car manufacturers including Honda, BMW, and Tesla among others. The autonomous vehicle market, which includes both semi-autonomous and fully-autonomous vehicles, was valued at 87.5 billion USD in 2021. This valuation is expected to reach 614.9 billion USD by the year 2030 [3].

1.1. Demand for Autonomous Vehicles

There are two primary motivations for developing these vehicles, that is, self-driving cars have the potential for dramatic improvements to the safety and convenience of our transportation systems. With the potential to save lives, autonomous vehicles are already revolutionizing transportation systems. 42,915 people were killed in motor vehicle crashes last year according to the National Highway Traffic Safety Administration (NHTSA) [13]. 94% of all automobile accidents are caused by human error [5]. Examples of human error include distracted driving, reckless driving, and drunk driving. These accidents are frequent because driving is a strenuous and difficult task that requires constant undivided attention from the driver for an extended period of time. Human drivers are also unpredictable. “Road rage” incidents have become commonplace as drivers’ ability to operate an automobile can be influenced by emotion.

Automated driving can make the movement of vehicles considerably more predictable. In addition to roadside emergencies, traffic congestion is often caused by abrupt changes in speed such as “rubbernecking” [17]. Rubbernecking refers to the sudden slowing down of vehicles
when drivers turn their heads to catch a glimpse of roadside accidents in passing (refer to Figure 1 below).

Figure 1. Illustration of a rubbernecking incident via South Carolina Department of Public Safety Twitter.

Therefore, improving the predictability of cars should in turn not only dramatically reduce the frequency of car accidents, but also improve traffic congestion, reducing commute times for everyone on the road. Additionally, if drivers hypothetically did not need to provide their undivided attention to operate an automobile, the time devoted to commutes could be spent focusing on other things and improving productivity as well as comfort. Populations of people that normally are unable to drive or have difficulties driving, such as the elderly and the disabled,
could also see improvements to their quality of life because autonomous vehicles could aid the mobility and independence of these people.

1.2. Ethical Considerations

The negative consequences of autonomous vehicles should also be addressed. The most obvious of these negative consequences are the ethical considerations. One such ethical dilemma is the universal accessibility of these autonomous vehicles. When developing new technologies, it is of paramount importance for engineers to ensure that the benefits of these new technologies are available to diverse communities. Personal vehicles are expensive and self-driving vehicles are more expensive. As a result of the price tag, autonomous vehicles may be unaffordable to most people and disproportionately benefit the wealthy populations that can afford them.

Another ethical consideration is the automation of labor. Lots of people make their living by driving. Currently, there are 2,350,464 drivers employed in the United States [7]. According to the British Broadcasting Corporation, robots are expected to replace 20 million workers before the year 2030 [16], and autonomous vehicles would only contribute to the number of displaced jobs. Finally, personal vehicles are bad for the environment. Personal vehicles emit, on average, close to 4.6 metric tons of carbon dioxide annually [10]. In addition to autonomous vehicles, it is important for engineers and infrastructure to invest equal resources in mass transportation, and push back against the widespread use of personal vehicles in the United States.

2. Theory

2.1. Cameras and Image Formation

Autonomous and semi-autonomous vehicles rely on perception information from a suite of sensors [1, 11]. This generally includes some combination of cameras, radar, LIDAR, etc.
Cameras are one of the favorite types of sensors employed by self-driving cars. For a low cost, cameras produce high-resolution, colored images. Cameras typically employ in some fashion a lens and a sensor. When a camera is aimed at a scene, light rays are reflected off of objects in the scene and travel through the lens [4]. The lens directs light to a sensor, which captures the resulting image which represents a 2-dimensional mapping of light in the image plane. Figure 2 below visualizes the intrinsics of a simple camera.

*Figure 2. The camera has a lens that redirects light in the image plane as it passes through the lens to form an image on a sensor. Image from [4].*
Figure 3. The image formation process involves a mapping of light in the world to some camera frame in the pose (the perspective coordinate system) of the camera. Image from [14].

The quality of these images makes cameras excellent at performing tasks related to object detection, recognition, and tracking. An example of the strength of cameras is the ability to identify and differentiate stop signs or other vehicles. However, cameras are not without drawbacks. They rely on wide-open, well-lit, sightlines. Cameras are susceptible to interference from weather conditions and can be distracted by rain, fog, or snow. They are also vulnerable to occlusion. Occlusion refers to sightlines being obstructed by objects such as buildings, trees, or
other vehicles. These factors make the effective range of cameras short in comparison to the range of other sensors such as radar.

2.2. Depth Estimation from Cameras

Using cameras for depth estimation is also difficult. Because images are 2-dimensional, they do not contain information about depth. It is possible to estimate the depth in images from multiple images taken of the same scene. This is known as stereo depth estimation. For the simple case of two images, that is a left and a right image, the calibration of the positions between two cameras is the pose. This pose consists of a rotation and a translation of the two camera views. By employing feature matching techniques to identify points of interest that appear in both the left and right camera views, known geometry can triangulate the 3-dimensional position of points of interest.

Monocular depth estimation, or depth estimation from one image, is technically impossible because a single image does not contain any information about 3-dimensional depth. In practice, however, monocular depth estimation can be implemented from monocular cues. These cues include the relative size of the image, color, and shape from shading (contours from shadows cast on objects in the image). To the human eye, these cues can be intuitive. Objects that are closer usually appear larger and brighter. The width of a car is usually around two meters. If a dog and a car were to appear side-by-side in an image, with similar pixel sizes, then the dog is probably closer to the point of reference (assuming it is not an abnormally large dog).

2.3. Monocular Depth Estimation via Machine Learning

Convolutional neural networks are a special type of neural network that are particularly useful for extracting features from images. The most straightforward method of training these deep learning models is to use images that are labeled with ground-truth depth information.
Ground-truth depth information can come from manual annotations to the images, or sometimes LIDAR data. Predicted depth maps can be then compared to ground-truth labels to minimize losses.

Figure 4. Monocular Input image versus machine learning generated depth maps. Image from [9].

Ground-truth depth labels can be difficult to obtain or expensive in the case of LIDAR. Using a mix of unlabeled data and some labeled data for training is called *semi-supervised*
machine learning. Using only unlabeled data is called self-supervised machine learning. When ground-truth depth labels are not available, stereo-image pairs can be used to train instead. When these pairs have a constant rotation and translation, known geometry can be used to generate depth labels (stereo depth estimation). However, even the pose can sometimes be difficult to obtain. Self-supervised and unsupervised machine learning models go a step further and use neural networks to predict pose. Pose predictions can then be used to triangulate 3-dimensional depth using unknown geometry.

2.4. Monodepth2 Architecture

The codebase we chose to use as a starting point for the project is Monodepth2. Monodepth2 is an open-source self-supervised machine-learning model for monocular depth estimation that was developed in 2018 [9]. Monodepth2 relies on monocular images and consecutive video frames to train its convolutional neural networks. Rotations and translations between consecutive frames are predicted from a convolutional neural network. A monocular depth decoder uses a convolutional neural network to generate a corresponding depth map for each individual monocular image. These pose and depth map predictions are used to warp an image to the frame of a nearby image. The photometric error loss is computed by comparing the brightness or intensity of each corresponding pixel in the warped synthetic image to its target image and the neural networks are trained to minimize these losses.
Figure 5. Monodepth2 Architecture. A target image $I_1$ is input to a depth decoder (neural network) and a pose decoder. Predictions of depth, rotation, and translation are used to warp a nearby image $I_2$ to the target image to form $\hat{I}_1$. Losses are computed from $I_1$ and $\hat{I}_1$.

To avoid matching occluded pixels, the losses are calculated using the minimum of pixels across images instead of the average.

Figure 6. A depth decoder and a pose decoder are used to respectively generate depth maps and pose predictions. Minimum losses across image sequences for pixels are recorded to improve the performance of the neural network. Image from [9].
3. Procedures and Accomplishments

The objective of our E90 project was to develop a platform that can produce 3-dimensional depth maps using input from 2-dimensional images. The input images would not contain ground-truth depth labels. Instead, we employ a self-supervised machine learning model.

3.1. Loading and Verification of Image Data

The KITTI Vision Benchmark dataset contains 180 gigabytes of self-driving car data, collected from a sensor-equipped station wagon driving around the mid-sized city of Karlsruhe, Germany [8]. The KITTI dataset was downloaded and stored in Amazon Simple Storage Service, otherwise known as Amazon S3.

The coding environment we chose to work in was Paperspace Gradient. Gradient is a virtual environment that is used for running machine learning experiments. Gradient was selected for its wide selection of powerful GPUs and performance. Data transfer from S3 to Gradient is read-only. Python code was written to verify that the full KITTI dataset was available in S3 and data transfer was possible. Python was also used to resolve JPEG versus PNG discrepancies that surfaced in the image file types.

3.2. Selecting Images for Training and Validation

Python code was written to generate the dataset eigen_chien by selecting images from the KITTI dataset. The Monodepth2 codebase uses consecutive video frames to train its convolutional neural networks. For each image selected, the code selects both the preceding and succeeding images as a set. Therefore, the first and last frame from any individual drive was never selected. Left and right images were treated as unrelated monocular images so that images from both cameras could be used for training. In total, 91,771 training items were selected.
The validation images were selected in a similar fashion. Validation images were specifically selected from the day with the least amount of driving data in the KITTI dataset (i.e. September 29th, 2011). The set of training items did not contain any of the images that were selected for validation. 3,094 items were selected for validation.

3.3. Training the Model

The Monodepth2 codebase was uploaded to a Gradient notebook with the PyTorch framework. To train the model, a GPU server was necessary. The GPU we selected was a PNY NVIDIA RTX A4000. The model was trained using the eigen_chien dataset. Even with a powerful GPU, the training took 60 hours to complete. Results were plotted and viewed with TensorBoard.

3.4. Radar Prep-Work

After obtaining preliminary results, we began to prepare radar data from the NuScenes dataset for image projections. The KITTI dataset contains annotations for objects [6] in the images such as class names (e.g. person, car, road sign), dimensions, and bounding boxes. NuScenes contains annotations [15] in a format that is different from the KITTI format. Before attempting to project the radar to image planes, it is important to make sure the NuScenes data is converted to the KITTI format to avoid compatibility issues. Python code from the NuScenes dev-kit (export_kitty.py) was adapted to successfully perform the conversion for a mini-sized dataset. This conversion was performed on a local machine. The program was not successfully adapted to execute the conversion on a virtual machine.

3.5. Challenges and Setbacks

Several unexpected challenges were encountered throughout the semester, which ultimately hindered our ability to train the model with radar data. First, the Monodepth2
codebase was uploaded to a Gradient notebook with TensorFlow. After encountering library compatibility issues with TensorFlow that we could not solve, we were forced to start the project over and use PyTorch instead.

After successfully executing the initial training of the model, we realized that it would be far too expensive to continue training in Gradient. Due to the size of the datasets, the data transfer costs from AWS to Gradient were too high and cost us hundreds of dollars to train our model. We, therefore, opted to pivot from Paperspace Gradient to a different coding environment, Amazon SageMaker.

SageMaker is a virtual machine learning environment that is native to the AWS ecosystem. The hourly computing costs for SageMaker were slightly higher than in Gradient, but we would save much more on data transfer costs. We had to revise our code when Sagemaker introduced a new set of library compatibility issues.

It was also more difficult to access GPU servers with SageMaker than Gradient. Access to GPU servers in SageMaker required express approval from an AWS representative, which we were only able to obtain after iterating back and forth with the representative.

4. Results

4.1 Accuracy of Model

The results of our initial training using image data only are viewable with TensorBoard.
Figure 7. The accuracy of predictions increases with training.

TensorBoard shows that the accuracy of our model improves with time spent training.

The accuracy of our predictions can be scored by comparing the similarity of the model’s predicted synthetic images to the target images that they were warped to.
4.2. Losses of Synthetic Imagery

![Figure 8. Losses of synthetic image warpings decrease with training.](image)

The losses are specifically the photometric error loss, that is, the per-pixel minimum reprojection loss. The per-pixel photometric loss at each pixel is the minimum photometric error over all source images. As expected, the more that the model is trained, the better it performs at the task of predicting synthetic images.
### 4.3. Error of Validation

![Figure 9](image.png)

*Figure 9. The error of validation images decreases as the model is trained.*

Finally, the error is evaluated by comparing depth maps from the validation images to ground truth data. That is opposed to the losses, which are computed by comparing warped training images to target training images. Error and loss are distinguished in this way. The error that we achieved training with the *eigen_chien* outperforms publication baselines for absolute relative error, root mean squared error, log root mean squared error, and relative squared error. We found absolute relative error close at levels below 0.14 (baseline at 0.140), root mean
squared error below 5 (baseline at 5.512), log root mean squared error below 0.22 (baseline 0.223), and relative squared error below 1 (baseline 1.6).

(a.)

(b.)

Figures 10a and 10b. Early synthetic image warpings (10a) show significant blurring and ghosting of features especially along the perimeters, but features of predictions become more discernible as more training is completed (10b).
5. Future Work

5.1. Radar Background Information

Radar is another sensor that shows promising results for self-driving applications [2]. Radar is much better than cameras at detecting the depth of objects. Radar works by emitting radio waves to generate 3D mappings of objects in the scene. These radio waves can penetrate through objects so radar is not so vulnerable to occlusion. Radar is effective at longer ranges than cameras and isn’t limited by weather and light conditions. However, data is low-resolution and not ideal for recognition tasks. The method of combining data from multiple sensors to overcome their individual limitations and get a better picture of the surrounding environment is called sensor fusion [1].
Figure 11. Radar systems send incident radio signals into the environment and locate other objects by detecting the reflected signal. Image [2].

5.2. Improvements to Model with Radar

For this sensor fusion, the code to convert NuScenes data to the KITTI format needs to be adapted to work correctly in the virtual environment (SageMaker). Then, the radar data from NuScenes can be used to supplement the image data for training the model. Radar is used to generate “point clouds” for objects in the scene [11, 12]. These point clouds can be expanded into infinitely tall pillars to represent the nature of regions that cannot be traveled through [12]. Then, the radar can be projected into the image plane. Training the model with radar-supplemented data should yield better results than images alone.
Figure 12. Expansions of radar point clouds to 3-dimensional pillars can be projected to image planes. Image from [12].
6. Satisfaction of ABET Criteria for Design

In order to prove that my senior design project satisfies ABET’s definition of a design project, I will address the following questions:

1. *What constraints govern your design problem?*

2. *What requirements will you (did you) develop?*

3. *How will you (or show that you did) evaluate your solution against the requirements?*

4. *What professional standards and codes, if any, govern your design (and if none, state why or that you were unable to find any)?*

6.1. *What constraints govern your design problem?*

The design of our originally planned methods was constrained by the sensors—no depth from the camera, no elevation from the radar. Also limiting us is the limited data with ground truth depth, which is difficult to obtain in the real world. This is why it is practical to apply instead monocular depth algorithms and extend radar point clouds as infinitely high pillars.

We did not eventually include radar in our models, that is left for future work. The image formation process is still a limiting factor. The quality of image data depends on an assortment of variables, such as occlusion, general visibility, and the resolution of cameras. Furthermore, even the best image in the world taken from the best camera is still just a 2-dimensional image with no way to theoretically perceive depth. The design of a model needs to cleverly overcome the limitations of the image formation process by using monocular cues and machine learning techniques.

We are also faced with computing constraints. Enormous datasets like KITTI and NuScenes require a lot of cloud storage. The intensity of training models with large datasets has
huge demands for computing power. Even Swarthmore’s computer cluster was not powerful enough, so we opted for virtual environments instead to utilize powerful GPU servers.

6.2. **What requirements will you (did you) develop?**

As it relates to requirements, we are NOT requiring real-time computing at this stage, but we need the model to outperform baseline monocular depth estimations. Additionally, the neural network needs to work in the computing framework we have (i.e. the neural network can be big and slow but not TOO big and slow).

6.3. **How will you (or show that you did) evaluate your solution against the requirements?**

Our end product can be evaluated against these requirements by comparing the error of our results to publication baselines for depth estimation. Greater detail is described in section 4, results.

In terms of actually making sure the neural network works in the computing framework we have, we were able to actually train our model in Paperspace Gradient with image data. Training took 60 hours (the neural network was indeed big and slow) but it was successful. We were also able to initiate training in SageMaker, although we did not complete a full training with just image data.

In the future, we will ideally be able to run our model with radar data in SageMaker.

6.4. **What professional standards and codes, if any, govern your design (and if none, state why or that you were unable to find any).**

Finally, professional standards and codes for our project are not so obvious. We don't have any professional code to judge by—we are going by publication standards of using the various metrics (e.g. percent of depth error less than the threshold, mean absolute error, mean
relative error, root mean squared error, etc). As stated in section 4 results, our results outperform the baseline model trained on the KITTI dataset.

8. Acknowledgements

With immense gratitude, this work was completed for E90, the Senior Design Project. The E90 provided me with the extraordinary opportunity to explore what it means to be an engineer beyond the conventional classroom setting, from the creative yet meticulous design process to all of the challenges we encountered along the way. I would also like to thank the Swarthmore College Engineering Department, whose esteemed faculty have consistently challenged me throughout my four years of college to think critically, solve problems across disciplines, and harness my potential as an engineer. Moreover, I am incredibly grateful for my E90 advisor, Professor Stephen Phillips, whose constant mentorship and guidance have inspired me to explore the exciting field of machine learning.
9. References…

https://www.thinkautonomous.ai/blog/aurora-deep-learning-sensor-fusion-motion-prediction/

https://doi.org/10.1109/TITS.2016.2632309


https://crashstats.nhtsa.dot.gov/Api/Public/Publication/812115

https://docs.nvidia.com/tao/tao-toolkit/text/data_annotation_format.html#:%3A%2F%3A%2Ftext%A2KITT%20format%20label%20file,Parameter%20name

https://www.zippia.com/driver-jobs/demographics/
https://doi.org/10.1177/0278364913491297

https://doi.org/10.48550/arXiv.1806.01260


https://www.epa.gov/greenvehicles/greenhouse-gas-emissions-typical-passenger-vehicle

https://www.thinkautonomous.ai/blog/aurora-deep-learning-sensor-fusion-motion-prediction/


