A Recreation of Through Fog High Resolution Imaging Using Millimeter Wave Radar

ECSE 6560 Modern Communication Systems
Project Final Report | 12/15/25
[Aidan Rosenblatt, rosena10@rpi.edu]
[Gabriela Crother-Collado, crothg@rpi.edu]

1 Abstract

The goal of this project is to simulate the effects of fog on millimeter wave radar sensing and recreate the realistic radar receive data in "Through Fog High Resolution Imaging Using Millimeter Wave Radar" by Guan, Madani, Jog, Gupta, and Hassanieh [1].

We accomplish this by studying and independently recreating the preprocessing portion of the paper. First, we use the MATLAB libraries included in the paper's GitHub repository to simulate the preprocess/data synthesization portion of the original paper and generate 3D point clouds and heatmaps. Then, we create our own MATLAB scripts that produce similar point clouds/heatmaps to compare and learn about the data and signal processing portions of the paper.

Our results aligned closely with the original results for this initial portion of the paper. This shows the accuracy of our simulation and displays realistic receive data from the radar.

2 Introduction

With the rise of autonomous vehicles, the importance of accurate and efficient sensing is more prevalent than ever. The most common types of sensors are LiDAR (Waymo cars) and optical sensors (Tesla self-driving cars). However, they both face similar problems when it comes to low-visibility situations. Unlike LiDAR and cameras, millimeter wave radar can penetrate through visibility impairments such as fog and still receive useable data. However, it suffers from low resolution, secularity, and noise sensitivity.

"Through Fog High Resolution Imaging Using Millimeter Wave Radar" aims to solve this issue by passing the low-resolution radar receive data through cGAN based deep learning architecture Hawkeye to recover high-resolution data. To do so, they need to produce accurate low-resolution training data. Their work takes models of cars and simulates how a millimeter wave radar would detect these cars with the presence of fog. They then generate 3D point clouds and heatmaps displaying what the radar would detect. This preprocessed data is what they pass through the Hawkeye program, and what we recreate in this project.

3 Related Work

Aside from [1], there is other work exploring the idea of using millimeter wave radar for autonomous vehicle sensing and sensing in general. [4] examines various implementations, their signal processing techniques, communication integration, and usage of deep learning. [5] explores the integration of various sensing methods, specifically LiDAR and radar, together to produce a multitarget-tracking method for autonomous vehicles. By doing so, they can produce accurate sensor data without using deep learning programs like most other single sensor tracking methods.

4 Design

As seen in Figure 1, we plan to replicate the data processing steps from the paper in the hopes of recreating their radar receive training data. The data processing procedure consists of 4 steps: construct and transform a full body point cloud, remove occluded points from the radar's POV, simulate the specularity of visible points, and construct a specularity heatmap. The point of this whole process is to make the heatmap as accurate to what the radar would actually read as possible.

Defining a coordinate system where the radar sits at the origin, we first translate the car randomly in the horizontal plane. Then we translate it down 1.5 meters to simulate the radar's actual height in a real-life experiment. The occluded points simulation removes non-visible points from the point cloud and leave behind 3D points that are visible from the radar's position at the origin. The specularity calculations tell us how well each point will reflect transmissions from the radar.

Figure 1: Project Block Diagram

All the data we process in this project is taken from a dataset containing CAD models and photos of various cars at various angles provided in HawkEye's GitHub repository [2].

5 Implementation

This project uses MATLAB to run existing scripts from the paper's GitHub repository; then we attempt to recreate this preprocessing independently using the same dataset used by the authors. To benchmark our results, we compare our intermediate and final results with the paper's to determine the accuracy of our recreation. Specifically, we tested our results with a baseline transformation of 45 degrees of rotation and a translation of (1500, 6000, -1250) millimeters.

The first step in the preprocessing pipeline is to analyze the full body. We accomplish this by transforming the CAD models into point clouds of 3D points, which we can more easily manipulate and transform for our purposes. We use MATLAB's load and sprintf functions to open the CAD file and then pass the file's cart_v field into the pointCloud object type constructor. Once we have the Nx3 matrix of coordinates that correspond with the surface of the car, we construct a bounding box whose corners lie at the maximum x, y, and z coordinates of the car itself. Then, we apply a rotation to the point cloud by generating a random angle from 0 to 360, constructing a rotation matrix, and multiplying the x and y coordinates of the point cloud by the rotation matrix. Finally, we translate every point in the cloud randomly in the x-y plane and by -1250 mm in the z-direction to replicate the relative position of the car to the radar in real life.

The next step in the process is to remove the occluded points (remove any points on the face of the car that are not accessible/visible to the radar). The is accomplished by converting the 3D cartesian coordinates of the point cloud into spherical coordinates, (x-y-z) → (φ-θ-r), using MATLAB's cart2sph function, and then sorting through the 3D points and only keeping the closest point to the origin for each viewing angle.

After removing the occluded points, we simulate the effects of fog and specularity of each point in the visible point cloud. To find the specularity of a point, p, we first compute the distance between p and the four edges of the bounding box to figure out which edge of the box p is closest to. Since we cannot know the actual orientation of the surface a point lies on, we use the orientation of the nearest bounding box edge as an approximation. Then, we use the angle function to determine the relative angle between the tangent of the nearest bounding box edge and the viewing angle of p from the origin, as shown in Figure 2. Then, we subtract 90 from the relative angle to compute the relative angle between the radar and the *normal* vector of the reflective surface.

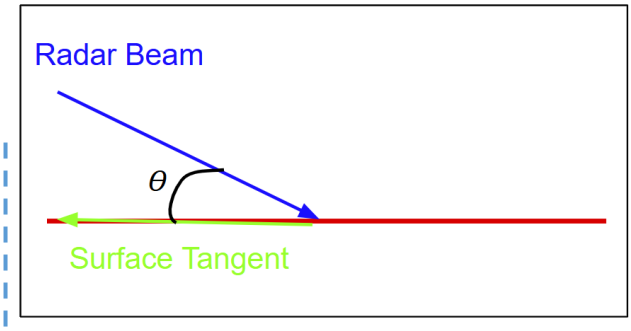

Figure 2: Relative Angle between Radar Beam and Surface Tangent

Importantly, our code does not actually calculate the real specularity of the points on the car, but we perform an approximation by using the relative orientation of the surface normal and the incoming radar beam. So, if a point has a relative angle of 0, that point lies on a surface perpendicular to the radar and will more strongly reflect any signals aimed at it.

Once we calculate the relative normal angles of each point's surface, we select all the points whose relative orientations are below a certain threshold and construct "blobs" of reflective points around them. These reflective blobs are most reflective at their centers and become less reflective towards their edges, approximately 0.3 meters away. The blobs are meant to model the reflected readings that the radar actually receives in a fog-covered test.

After determining the specularity, we create a heatmap displaying this radar receive data, which could be passed through the Hawkeye program. We compare this and our intermediate point clouds to the original results to determine our accuracy.

6 Evaluation

We chose to verify our data synthesizer by comparing the output at each step in Figure 2 with the output from [2] for a set car orientation. We conduct our comparisons after rotating the point cloud by 90 degrees about the z-axis, and translated by [1.5, 6, -1.25] meters. Overall, our results match relatively closely with the paper's, aside from a few implementation differences. With some minor changes to follow the original code, we were able to produce very similar results.

6.1 Full Body Analysis

The first step of the data synthesization pipeline is straightforward, mainly consisting of matrix multiplication and addition to rotate and translate the point cloud created from the CAD model. As expected, our resulting point cloud matched the control point cloud perfectly, as shown in Figure 3.

6.2 Point Occlusion

The point occlusion portion performed as expected, producing nearly identical results to the paper, as shown in Figure 4. Our synthesizer successfully removes the non-visible points and leaves only the points the radar can read.

6.3 Reflectivity

While our implementation did not produce identical results for the reflectivity portion (shown in Figure 5), the data is very similar and useable for the next step. The discrepancies are likely due to slightly different methods for determining specularity and the angle of the point to the radar.

6.4 Radar Simulation

The final step is simulating the recieved data from the radar and producing a heatmap to display the data. Our synthesizer produces fairly different results for this step, as shown in Figure 6. This is likely due to the compounding differences in reflectivity as well as implementation of the radar simulation. Upon altering our version to more closely align with the paper's, we are able to produce more similar results. However, we could not produce these results independently.

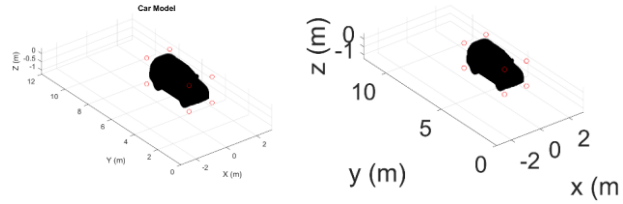

Figure 3: Student-Made Car Point Cloud (left) and HawkEye Point Cloud (right)

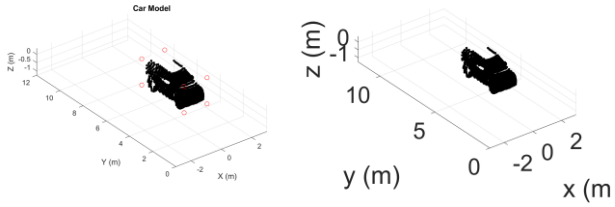

Figure 4: Point Cloud w/out Occluded Points, Student-Made (left) and HawkEye (right)

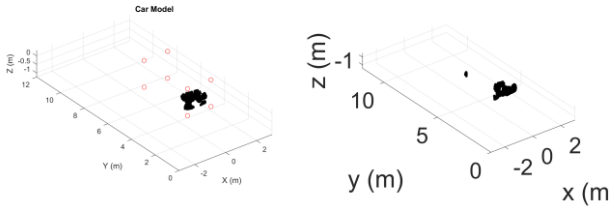

Figure 5: Reflective Points Point Cloud, Student-Made (left) and HawkEye (right)

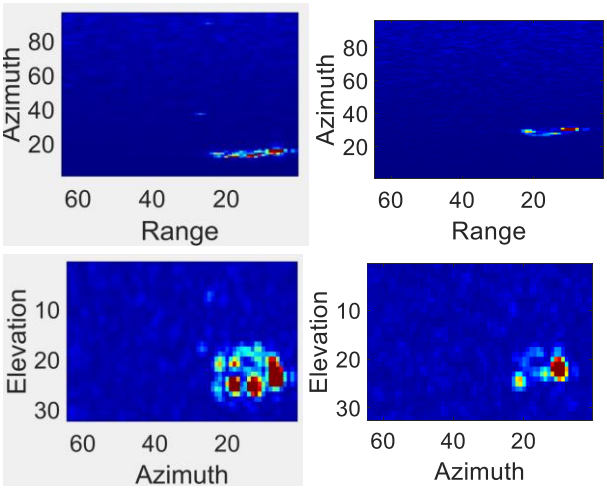

Figure 6: Heatmap, Top View (top) and Front View (bottom), Student-Made (left) and HawkEye (right)

7 Conclusion

This project successfully recreated the preprocessing and data synthesization steps presented in "Through Fog High Resolution Imaging Using Millimeter Wave Radar". By first utilizing the MATLAB tools provided by the authors and then independently developing our own synthesizer, we demonstrate a strong understanding of the paper's implementation and underlying signal processing concepts. Our independently generated point clouds and intermediate results closely matched those produced by the original implementation, validating the correctness of our approach, and confirming that the key preprocessing stages were accurately reproduced.

The full body analysis and point cloud transformations were implemented precisely and matched the reference results exactly. Additionally, the occlusion removal stage performed as expected, producing nearly identical visible point clouds from the radar's perspective. These stages form the core of the preprocessing pipeline and were successfully validated. While the reflective modeling stage produced results that were slightly offset from the original implementation, the outputs remained consistent and usable for subsequent processing. The heat map and radar simulation components were partially implemented but not fully finalized, limiting direct comparison at the final output stage.

Overall, this project provided a valuable learning experience in radar signal processing, geometric transformations, and simulation-based data generation. Future work will focus on improving the reflective model to more accurately capture surface specularity, completing the heat map generation, and implementing a more detailed radar simulation. Additional extensions include incorporating the deep learning model described in the paper and integrating real hardware to move beyond simulation and toward real-world validation. These improvements would further enhance the realism and applicability of the synthesized radar data.

8 Acknowledgement

Concepts from ECSE-6560 Modern Communication Systems were used in the implementation of our synthesizer. Chat GPT was used to refine the grammar and writing of this report.

9 Team Contribution

Aidan Rosenblatt: Full body analysis, point occlusion, reflectivity, radar simulation, presentation, and report.

Gabriela Crother-Collado: Planning, radar simulation, presentation, and report.

10 References

[1] J. Guan, S. Madani, S. Jog, S. Gupta, H. Hassanieh. "Through Fog High Resolution Imaging Using Millimeter Wave Radar." SyNRG. Accessed: Nov. 10, 2025. [Online.] Available: https://jguan.page/HawkEye/

[2] J. Guan, S. Madani, S. Jog, S. Gupta, H. Hassanieh. "HawkEye Dataset & Radar Data Synthesizer." GitHub. Accessed: Nov. 10, 2025. [Online.] Available: https://github.com/JaydenG1019/HawkEye-Data-Code

[3] J. Guan, S. Madani, S. Jog, S. Gupta and H. Hassanieh, "Through Fog High-Resolution Imaging Using Millimeter Wave Radar," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 11461-11470, doi: 10.1109/CVPR42600.2020.01148.

[4] H. Kong, C. Huang, J. Yu and X. Shen, "A Survey of mmWave Radar-Based Sensing in Autonomous Vehicles, Smart Homes and Industry," in IEEE Communications Surveys & Tutorials, vol. 27, no. 1, pp. 463-508, Feb. 2025, doi: 10.1109/COMST.2024.3409556. keywords: {Millimeter wave communication;Radar;Sensors;Surveys;Radar imaging;Radar detection;Deep learning;Millimeter wave radar;wireless sensing;radar signal processing;deep learning;autonomous vehicle;smart home;industry}

[5] J. Shi, Y. Tang, J. Gao, C. Piao, and Z. Wang, "Multitarget-Tracking Method Based on the Fusion of Millimeter-Wave Radar and LiDAR Sensor Information for Autonomous Vehicles," Sensors, vol. 23, no. 15, p. 6920, Aug. 2023, doi: 10.3390/s23156920.