The HawkEye Radar Synthesizer Code Documentation

Junfeng Guan

September 28, 2020

Abstract

A documentation of the HawkEye radar data synthesizer Matlab implementation. The documentation includes a description of the files, the input parameters, the functions.

1 Introduction

Collecting real-world Millimeter-wave radar data for training can be very time-consuming. To address this, we build a synthesizer to generate 3D mmWave heatmaps of cars from 3D CAD models. Our synthesizer is designed to create 3D point reflector models of cars and then simulate mmWave radar signals.

2 License

3 Overall Structure

File	Description
main.m	main radar data simulation function
variable_library.m	library of various variables
variable_library_radar.m	radar configuration related variables
remove_occlusion.m	remove occluded body of the car from the scene
model_point_reflector.m	model radar point reflectors in the scene
simulate_radar_signal.m	simulate received FMCW radar beat signals
radar_dsp.m	radar signal processing to generate 3D heatmaps

Table 1: Matlab Files in the Source Code

4 Instructions to Use Codebase

4.1 Synthesizer Setup

Before running the program, we need to setup the synthesizer configurations in the 'variable_library.m' file, and the simulated mmWave radar configurations in the 'variable_library_radar.m' file. We describe the available parameters in Table 2 and Table 3

Parameter	Description
N_CAD_car	number of CAD models of cars (in ./Synthesizer/CAD)
N_placement_car	number of placements of each car
rotate_ang	set of rotation angles of the car
translate_lim	limits of the translation of the car along the x and y axis
translate_x_res	resolution of translation along the x axis, unit: mm
translate_y_res	resolution of translation along the y axis, unit: mm

Table 2: Synthesizer Configuration

4.2 Synthesizer Input & Output

Input: HawkEye synthesizer takes input of the 3D CAD models of cars. The CAD models are stored in the '/Synthesizer/CAD' folder in the '.mat' MATLAB binary data container format. Every file contains the CAD model of a car, and is formatted as a N by 3 array, where N is the number of points on the surface of the car, and the three columns represent the x,y,z cartesian coordinates of the points respectively (unit: mm). The center of the horizontal (x-y) plane of the

Parameter	**Description**
radar_FoV_deg	azimuth and elevation field of view
radar_FoV_rho	range field of view
N_phi	number of azimuth bins in the output radar heatmap
N_theta	number of elevation bins in the output radar heatmap
fc	radar wavelength
BW	FMCW radar waveform bandwidth
Ts	FMCW sweep time
Rs	Baseband sampling rate
TX_pos	TX antenna position
array_size	number of antenna elements on x and z axis
array_gap	antenna element spacing on x and z axis

Table 3: mmWave Radar Configuration

car is placed at the origin, while the lowest point (with minimal z coordinate) is placed on the z=0 plane.

Data Structure for the Car Point Clouds: We store the point cloud of each car in the 'car_v_struct' data structure as shown in Table 4.

Parameter	**Description**
CAD_idx	CAD model index
N_pt	number of points in the point cloud
cart_v	cartesian coordinates vector of each point
sph_v	spherical coordinates vector of each point
lim	cartesian coordinates limits
bbox	cartesian coordinates of the 8 vertices of the car bounding box
rotate	degree of rotation of the car
translate	translation of the car along the x,y,z axis

Table 4: Car Point Cloud Data Structure

Output: The output of HawkEye synthesizer program is the simulated 3D radar heatmap and the ground truth point cloud of visible (not occluded) surface the car.

The output 3D radar heatmap is stored in the 3D array 'radar_heatmap', where the three dimensions correspond to range (ρ), azimuth (ϕ), and elevation (θ) respectively. The values in the array represent the measured radar reflection power from the corresponding voxel in the space. The size of each dimension is determined by the radar field of view and resolution configurations.

The ground truth point cloud of visible (not occluded) surface the car is stored in the 2D array 'visible_cart_v'. Similar to the input point cloud of the car, it is also a N by 3 array, where N is the number of visible (not occluded) points

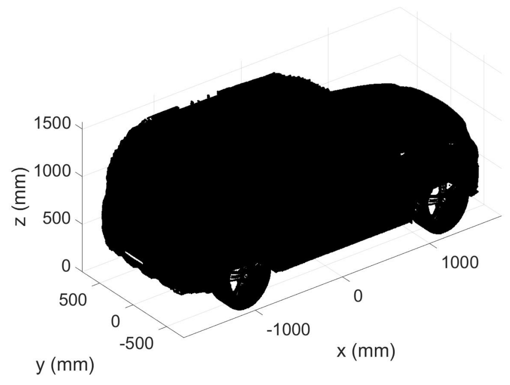

Figure 1: Input point cloud of the CAD model of a car.

on the surface of the car to the radar. The three columns represent the x,y,z cartesian coordinates of the points in meters.

4.3 Running Procedure

Once the synthesizer and radar parameters are configured, we can run the program by running the 'main.m' function. It loops through all CAD models of cars, and with each car it creates a number of scenes with different placements of the car.
After creating the scene, it first removes the occluded body of the car ('/functions/remove_occlusion.m'), and model the remaining visible surface of the car into radar point reflector models taking into consideration of the specularity at different parts of the car ('/functions/model_point_reflector.m').
Finally, we simulate the received reflected FMCW radar beat signals ('/functions/simulate_radar_signal.m'), and use it to generate the 3D radar heatmap ('/functions/radar_dsp.m').

5 Example Run

Here, we demonstrate a radar heatmap synthesizing process. Figure 1 shows the input CAD model of a car. After rotation and translation, the scene created with this model is shown in Fig. 2. Then we show the modeled radar point reflectors in the scene in Fig. 3. Finally, we simulate the FMCW beat signals in the receiver array and recreated the 3D radar heatmap. We show the top view (range-azimuth plane projection) of the 3D radar heatmap in Fig. 4 and the front view (azimuth-elevation plane projection) of the 3D radar heatmap in Fig. 5.

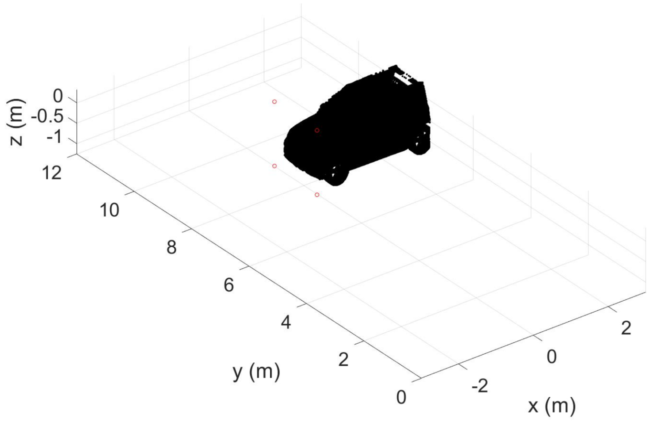

Figure 2: Scene created through rotating and translating the car model.

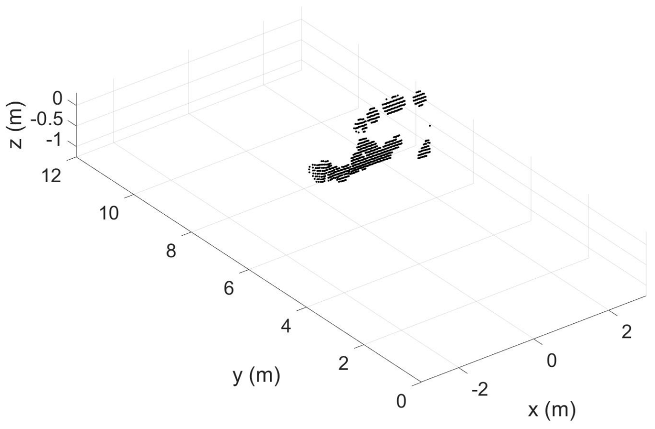

Figure 3: Modeled radar point reflectors in the scene.

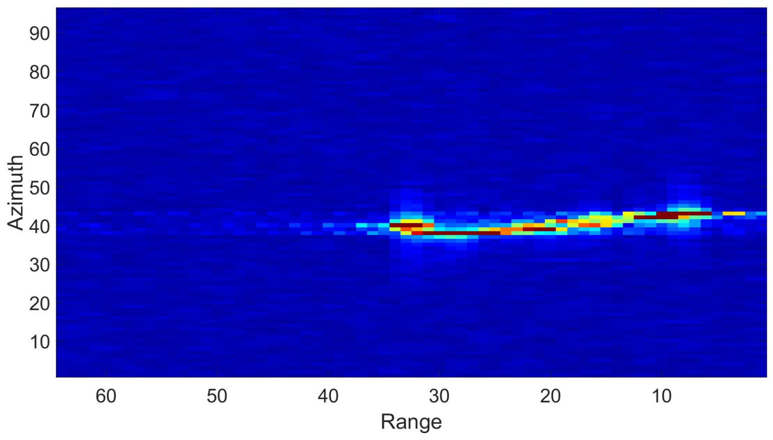

Figure 4: Synthesized radar heatmap top view.

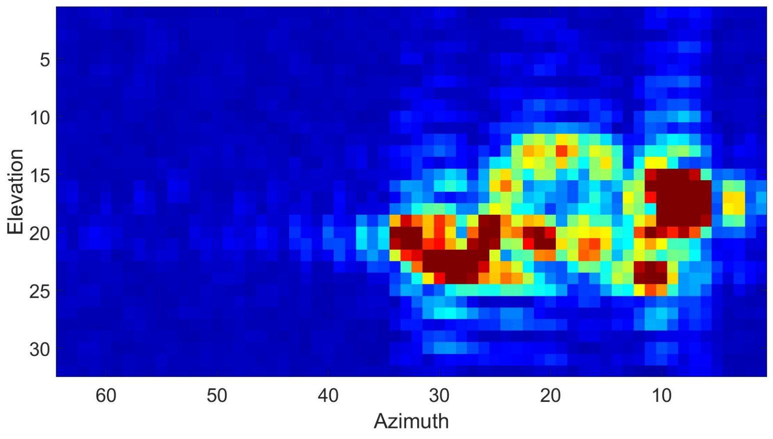

Figure 5: Synthesized radar heatmap front view.