Supplementary Materials for Through Fog High Resolution Imaging Using Millimeter Wave Radar

Junfeng Guan   Sohrab Madani   Suraj Jog   Saurabh Gupta   Haitham Hassanieh
University of Illinois at Urbana-Champaign

Abstract

This supplementary material presents additional qualitative results from HawkEye. It also provides more detailed descriptions of HawkEye's GAN architecture, data collection hardware platform, and how the quantitative metrics are extracted.

Appendix A: Additional Qualitative Results

We present additional qualitative results from HawkEye in three categories. First, we show an example of HawkEye's performance on the synthesized test dataset along with the original 3D CAD model we use in our data synthesizer (section 5) in Fig. 1. We only show one example of HawkEye's output on the synthesized test dataset, because HawkEye can accurately recreate all detailed features as in the synthesized ground-truth most of the time. Hence, we focus on results from the real test dataset.

Second, we show HawkEye's performance with rain in the scene in Fig. 2, which is also part of our controlled experiments (section 6). We use a water hose to emulate rain around the object of interest (the car), and use HawkEye to image through the emulated rain. We observe that despite not being trained with such examples, HawkEye still generalizes well with rain in the scene and captures the location, orientation, size, and shape of the car. As we have discussed in section 6.2, due to FCC regulations, we are constrained to build our experimental setup at the 60 GHz unlicensed spectrum, which suffers from higher attenuation from water particles. We believe HawkEye's performance would further improve in inclement weather conditions by implementing HawkEye's mmWave radars at the 77 GHz frequency band, which is allocated specifically for automotive radar applications.

Finally, we also show additional *randomly sampled* qualitative results from HawkEye on the real test dataset in Fig. 6 and Fig. 7. We can see that, in most cases, HawkEye accurately depicts the location, orientation, size, and shape of different car models in various background environments and viewpoints.

Appendix B: Network Architecture and Implementation Details

HawkEye's **generator** network (section 4) follows standard encoder-decoder architecture.

The **encoder** starts with one channel of 3D input (ϕ, θ, ρ) of size $1 \times 64 \times 32 \times 96$. There are 6 3D convolution layers with kernel sizes=6 and strides=2 on all three dimensions. With every convolution layer, the number of channels increases, while the 3D feature map size decreases by half in every dimension. We use a BatchNorm layer followed by a Leaky-ReLU layer after every 3D convolution layer. The encoder outputs a $2048 \times 1 \times 1 \times 1$ dimensional z-vector, which is then squeezed to $2048 \times 1 \times 1$. At each layer we appropriately zero-pad the features in order to get the desired input and output sizes.

The **decoder** begins with the $2048 \times 1 \times 1$ dimensional z-vector, and it contains 8 2D deconvolution layers. The first deconvolution layer has a kernel size=$(4, 3)$, and stride=$(2, 1)$ in order to get an output of size $(2, 1)$. The subsequent deconvolution layers have kernel sizes=4 and strides=2 on both dimensions. Hence, each layer doubles the feature map sizes and we also decrease the number of channels. We use a BatchNorm layer followed by a ReLU layer after every deconvolution layer. Again, at each layer we appropriately zero-pad the features in order to get the desired input and output sizes. This results in a output feature map of size $1 \times 256 \times 128$. We also use *skip connections* between the encoder and the decoder. Using the 3D heatmap to 2D depth-map mapping defined in section 4, we obtain 8 channels of 2D feature maps of size 64×32. We concatenate them with the feature maps at the 6^{th} deconvolution layer along the channel axis. The concatenated feature map then goes through the last deconvolution layer and a fully connected layer. Finally, we apply the hyperbolic tangent function alongside a linear transformation that maps the final output between 0 and 1. This leads to the 256×128 generator output $G(x)$.

The **discriminator** network (section 4) adopts a two-stream architecture that uses two separate encoders to map 3D radar heatmaps and 2D depth-maps to 1D feature vec-

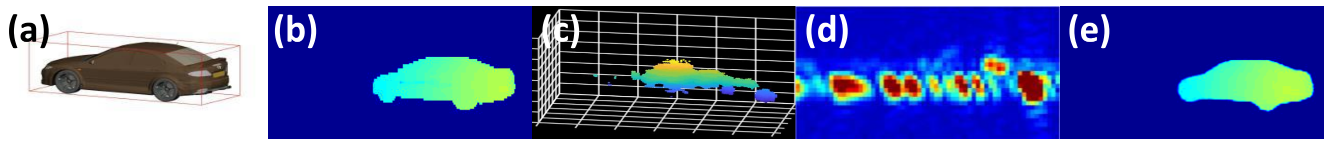

Figure 1: HawkEye's performance on synthesized test data. Column (a) shows the original 3D CAD model of car. Column (b) shows the corresponding synthesized ground-truth depth-map. Column (c) and (d) show the synthesized radar heatmap in the form of 3D point-cloud and 2D front-view projection respectively. Column (e) shows the HawkEye output.

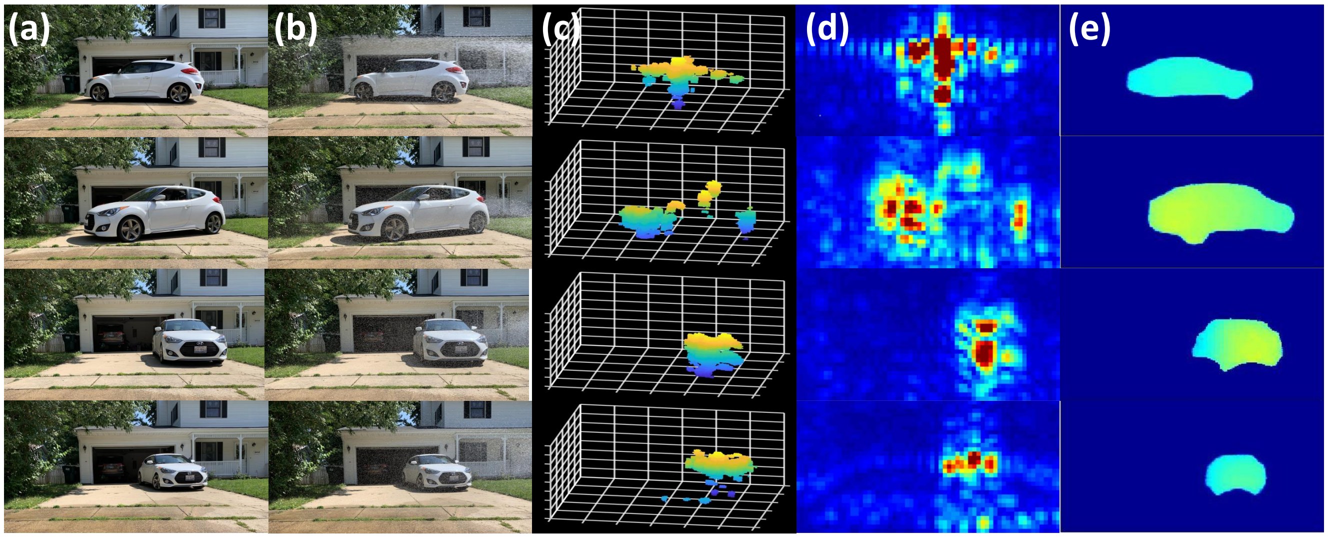

Figure 2: HawkEye's performance with rain in scene. Column (a) shows the original scene. Column (b) shows the scene with rain. Column (c) and (d) show the radar heatmap in the form of 3D point-cloud and 2D front-view projection respectively. Column (e) shows the HawkEye output.

tors. The encoder for 3D radar heatmaps comprises the generator's encoder architecture resulting in 512 dimensional output vector z'. The encoder for 2D depth-maps takes the ground-truth y or the generator output $G(x)$ as input and also outputs a 512 dimensional vector z''. It has 8 2D convolution layers with kernel sizes $= 4$ and strides $= 2$ on both dimensions with appropriate zero-padding. Each convolution layer is followed by a BatchNorm layer and a Leaky-ReLu layer. z' and z'' are then concatenated and fed into two fully connected layers with ReLu and Dropout layers in between. Finally, we use the sigmoid activation function to get the output probability of the discriminator.

To calculate the **perceptual loss**, we feed HawkEye's output $G(x)$ and the corresponding ground-truth y into a pretrained VGG16 [5] model, by replicating $G(x)$ and y to three channels. We obtain outputs of the VGG model at the 3^{rd}, 8^{th}, 15^{th}, and 22^{nd} layers for $G(x)$ and y. Then we compute the L1 difference of the outputs at each layer and normalize them to get the perceptual loss $\mathcal{L}_p(G)$. We set the weight for $\mathcal{L}_1(G)$ and $\mathcal{L}_p(G)$ to be $\lambda_1 = 1000$ and $\lambda_p = 20$.

We train HawkEye with the Adam optimizer and batch size=4 on an Nvidia Titan RTX GPU. We start with a learning rate=10^{-4} for the first 100 epochs. We the linearly decay the learning rate to zero for another 70 epochs. Lastly, in the fine-tuning stage, we use a learning rate of 10^{-5}.

The runtime for HawkEye's 3D mmWave heatmap to higher-resolution depth-map translation is 23ms per image on a single Nvidia Titan RTX GPU.

Appendix C: Data Collection Platform

We collect our own dataset, which includes 3D mmWave radar heatmaps of cars and the corresponding stereo camera depth-maps. We built a mmWave imaging system using 60 GHz radios and a SAR (Synthetic Aperture Radar) platform. We send standard FMCW radar waveform generated by our custom-build FMCW circuit.

We leverage SAR to emulate a 2D antenna array by mechanically scanning a single mmWave radio. We opt for SAR-based implementation because of the limited availability of 2D phased arrays with hundreds of antennas like [4, 7] in commercial systems and the high flexibility of SAR. SAR provides us with a reconfigurable antenna array for a wide range of frequencies and aperture sizes, which allows us to generate radar heatmaps with different resolutions. We build a 2D SAR platform shown in Fig. 3 using three FUYU FSL40 linear sliders [6] with sub-millimeter

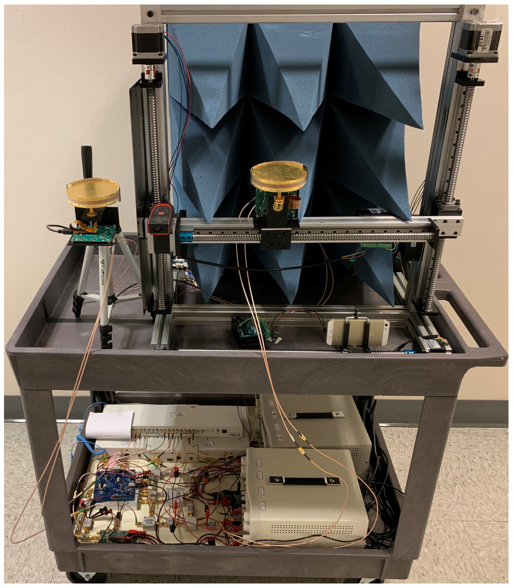

Figure 3: Data Collection Platform

accuracy. We mount a Pasternack 60 GHz radio front-end [2] on the SAR platform as the receiver, and another radio on the side as the transmitter. We use omni-directional antennas for both the transmitter and receiver to have a maximum field-of-view of 180° in azimuth and 35° in elevation. We also place RF absorbers and shields around the antennas to eliminate the direct path leakage and unwanted reflections from the backside. The horizontal slider scans the mounted receiver radio along the X-axis, while two vertical sliders scan along the Z-axis. In HawkEye, only a fraction of $20cm \times 20cm$ area is scanned to emulate a 40×40 array at 60 GHz, which provides $\sim 8°$ angular resolution along azimuth and elevation axes. The scanning time is 5 minutes and reduces to 90 seconds for a 20×20 array.

For the mmWave radar circuit, we implement a heterodyne architecture, as shown in Fig. 4. We first generate the same FMCW waveform at baseband using ADF4159 PLL (Phased Locked Loop) [1], with a bandwidth of 1.5 GHz sweeping from 0.1 GHz to 1.6 GHz. Then we up-convert it to have a center frequency of 60.15 GHz using quadrature modulation. The resulting signal sweeps from 59.4 GHz to 60.9 GHz with the other unwanted sideband suppressed. The FMCW receiver has a reciprocal architecture. The reflected signals at 60 GHz are first down-converted to the baseband through quadrature demodulation to get 90° phase-shifted I and Q channels. Then we feed them separately into RF mixers along with the original baseband FMCW waveforms to extract the beat signal, whose frequency is proportional to the time-of-flight of the radar waveform in the air. We sample the I and Q components of the complex beat signal with two N210 USRP software-defined radios [3] for direct phase measurement.

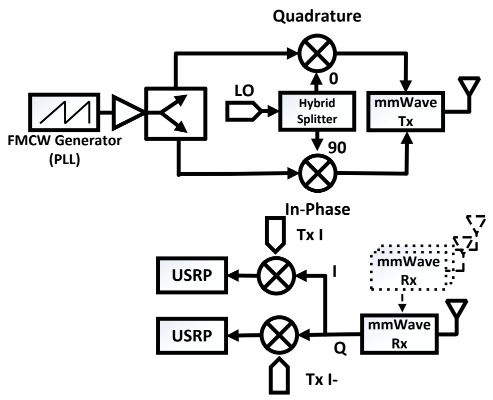

Figure 4: Millimeter Wave Imaging Radar Circuit Diagram

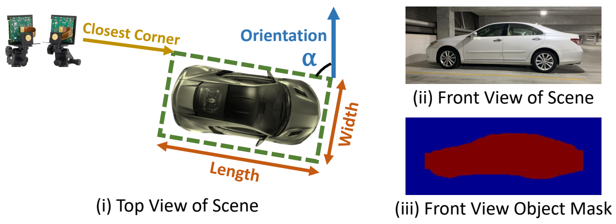

Figure 5: Quantitative Metrics

We use a common clock to enforce frequency and phase synchronization in the radar circuit.

We then align the continuously sampled complex beat signal to the antenna positions in the array. In this process, we track the SAR trajectory by leveraging the phase shift in the residual direct path leakage. We then apply Fast Fourier Transform and conventional beamforming in sequence to estimate the reflected signal power from every voxel $x(\phi, \theta, \rho)$ to generate the 3D mmWave radar heatmap.

Appendix D: Extracting Quantitative Metrics

We evaluate on range, size (length, width, height), and orientation of the car, as they represent the contextual information of the car in the scene (shown in Fig. 5(i)). We define the distance to the closest corner of the car as the range, and orientation as the angle between the longer edge of the car and the 0° azimuth of the mmWave heatmap. First, we convert depth maps into 3D point clouds in the camera frame based on the mapping from pixel values to metric depth as shown in the scale bars. Then we estimate the bounding boxes of cars by projecting the point clouds onto the horizontal plane and fitting the points into either a 90-degree corner or a straight line. For radar heatmaps, we manually annotate the bounding boxes in the 2D top-view projections, similar to Fig.3(c). Finally, we fill up the occluded parts of the bounding boxes and extract the metrics.

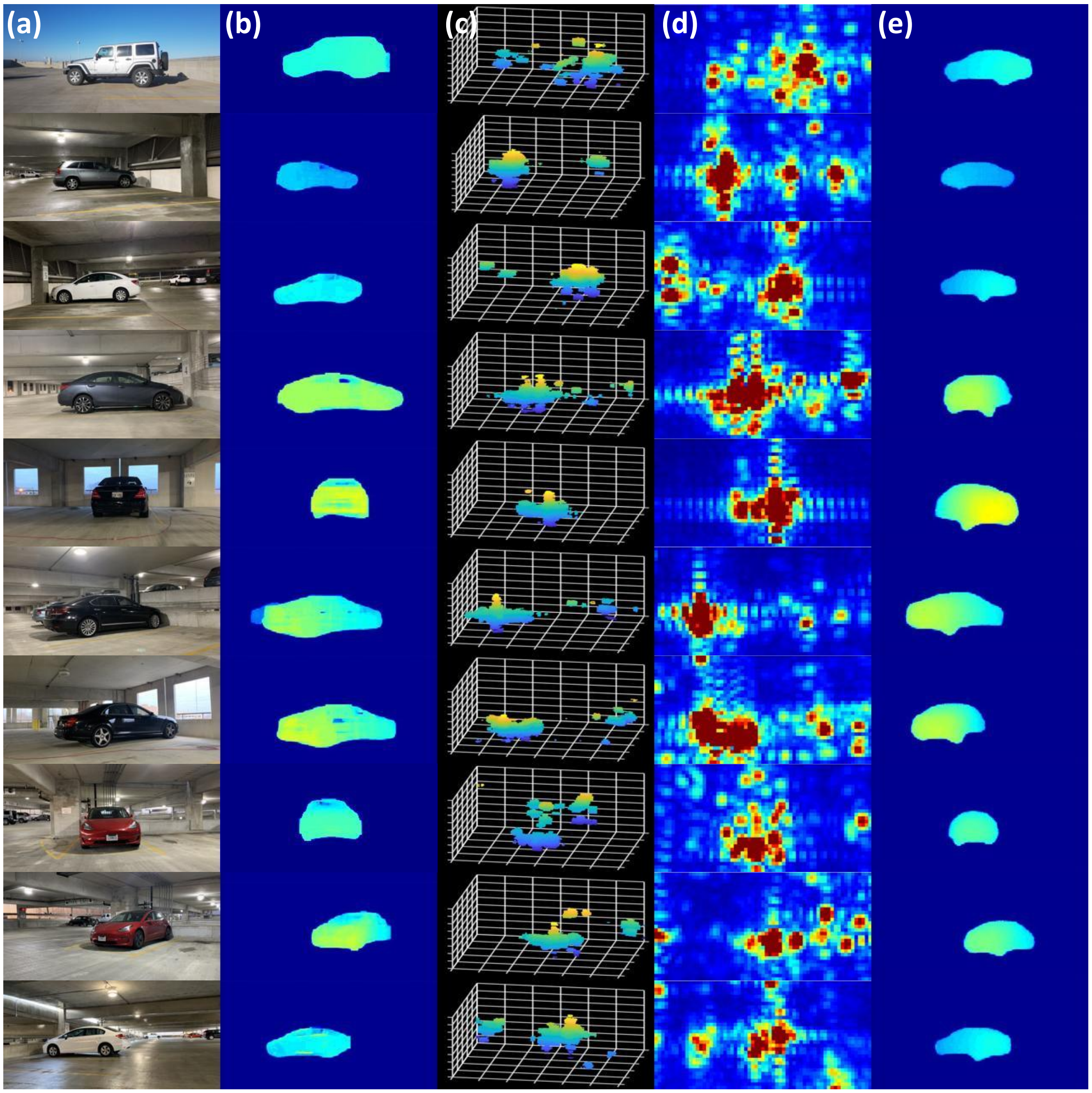

Figure 6: *Randomly sampled* qualitative results from HawkEye. Column (a) shows the original scene. Column (b) shows the corresponding ground truth. Column (c) and (d) show the radar heatmap in the form of 3D point-cloud and 2D front-view projection respectively. Column (e) shows the HawkEye output. Continued on next page.

Range, length, width, and orientation of the cars are computable from the corners and edges of the top-view bounding boxes, while heights are estimated from the 3D points inside the bounding boxes.

We also evaluate accuracy in shape prediction by comparing *(a) % of Car's Surface Missed* (False Negatives) and *(b) % of Fictitious Reflections* (False Positives) in the object masks of HawkEye's output along the front view of the scene as shown in Fig. 5(ii,iii). Note that (a) is indicative of the specularity effects whereas (b) is indicative of artifacts such as multipath and ambient reflections in the image. We extract the object masks from the mmWave heatmap and the outputs of HawkEye and baseline methods separately, and then compute the False Positive Rates (FPR) and False Negative Rates (FNR) against the ground truth object masks. We obtain the ground truth object mask by applying Mack-

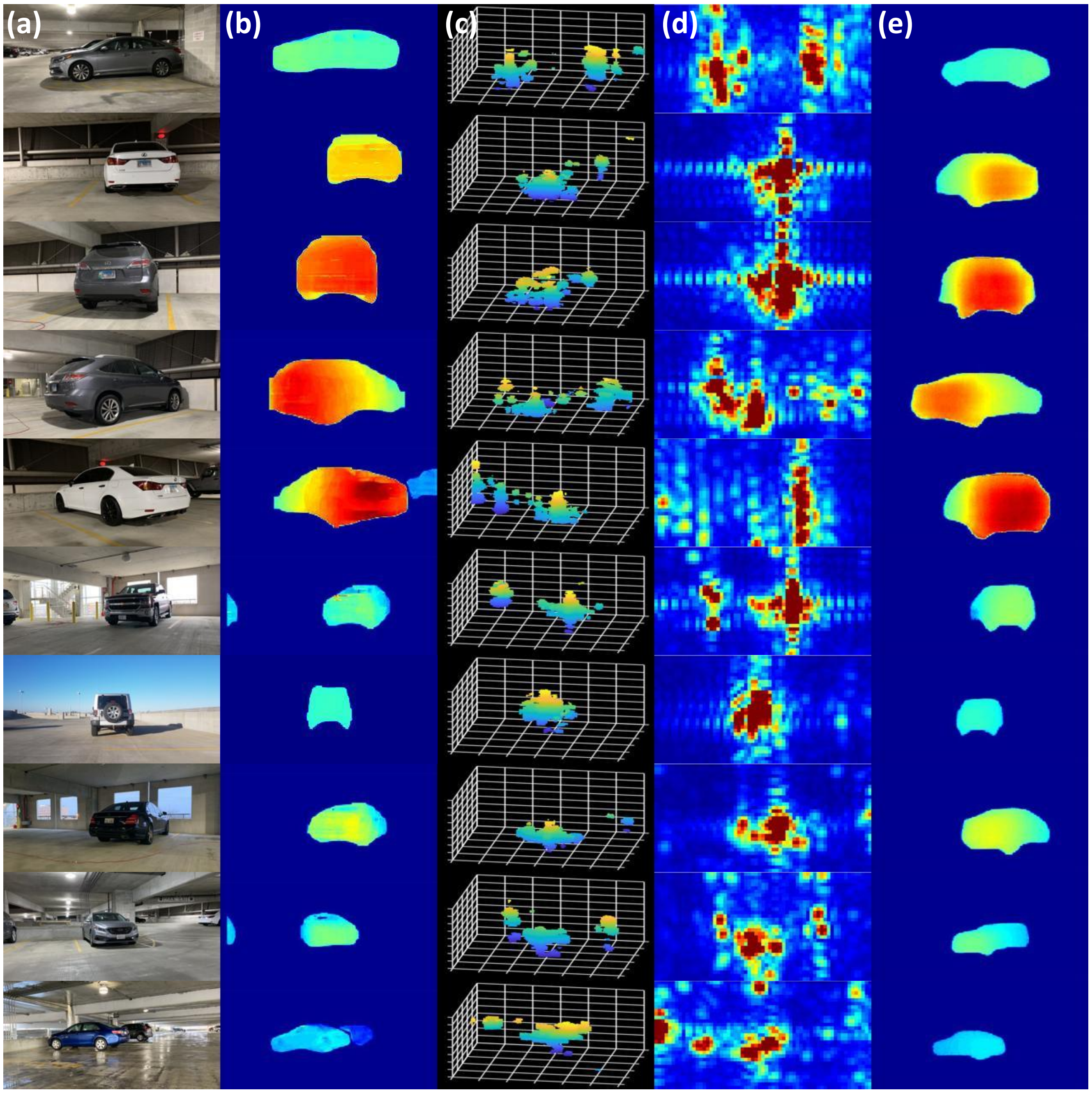

Figure 7: *Randomly sampled* qualitative results from HawkEye. Column (a) shows the original scene. Column (b) shows the corresponding ground truth. Column (c) and (d) show the radar heatmap in the form of 3D point-cloud and 2D front-view projection respectively. Column (e) shows the HawkEye output.

RCNN on the camera image. For the depth map outputs of HawkEye and the baseline methods of L_1 and Nearest Neightbor, we eliminate noise that are far away from the object body using a distance threshold, so that the remaining pixels in the region of the object form the mask. For mmWave heatmaps, we project the 3D heatmaps onto the front-view plane and then select pixels exceeding a power threshold as the object mask. We choose the distance and power thresholds from the ROC (Receiver operating characteristic) curves.

References

[1] Analog Devices. Adf4159 pll. `https://www.analog.com/en/products/adf4159`, 2014.

[2] Pasternack. 60 ghz development system. `https://www.pasternack.com`, 2014.

[3] Ettus Research. Usrp n210. `https://www.ettus.com/all-products/un210-kit`, 2012.

[4] S. Shahramian, M. J. Holyoak, A. Singh, and Y. Baeyens. A fully integrated 384-element, 16-tile, w -band phased array with self-alignment and self-test. *IEEE Journal of Solid-State Circuits*, 54(9):2419–2434, 2019.

[5] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[6] FUYU Technology. Linear motion guide. `https://www.fuyumotion.com/products/linear-motion-guide`, 2020.

[7] S. Zihir, O. D. Gurbuz, A. Kar-Roy, S. Raman, and G. M. Rebeiz. 60-ghz 64- and 256-elements wafer-scale phased-array transmitters using full-reticle and subreticle stitching techniques. *IEEE Transactions on Microwave Theory and Techniques*, 64(12):4701–4719, 2016.