

Supplementary Material

TICaM: A Time-of-flight In-car Cabin Monitoring Dataset

Jigyasa Singh Katrolia

jigyasa.katrolia@dfki.de

Ahmed Elsherif

ahmad.elsherif27@gmail.com

Hartmut Feld

hfeld@posteo.de

Bruno Mirbach

bruno.mirbach@dfki.de

Jason Rambach

jason.rambach@dfki.de

Didier Stricker

didier.stricker@dfki.de

DFKI

German Research Center for

Artificial Intelligence

Kaiserslautern, Germany

1 TICaM Example Images

In Figure 1 we show example synthetic (top 3 rows) and real (bottom 3 rows) images from TICaM. The images showcase a range of driving scenarios, different body poses as well as various activities performed by people in the scene.

2 Driving Scenarios Captured in TICaM

We record a variety of realistic driving scenarios in TICaM with different seat occupancies as depicted in Figure 2. We also introduce additional variations in the scene. Figure 3 shows a participant in various car seat positions, in different clothing accessories and performing different actions while driving.

3 Training Details for Baseline Evaluation

3.1 YOLACT Architecture and Training Details

We adapt YOLACT implementation from [1] for our depth images for 2D object detection and segmentation. We use ResNet-101 [2] with FPN [3] as the backbone feature extractor. We initialize the backbone network with weights pretrained on ImageNet [4]. The model is trained on two GTX 1080 GPUs for 12 epochs with a batch size of 8 and the performance

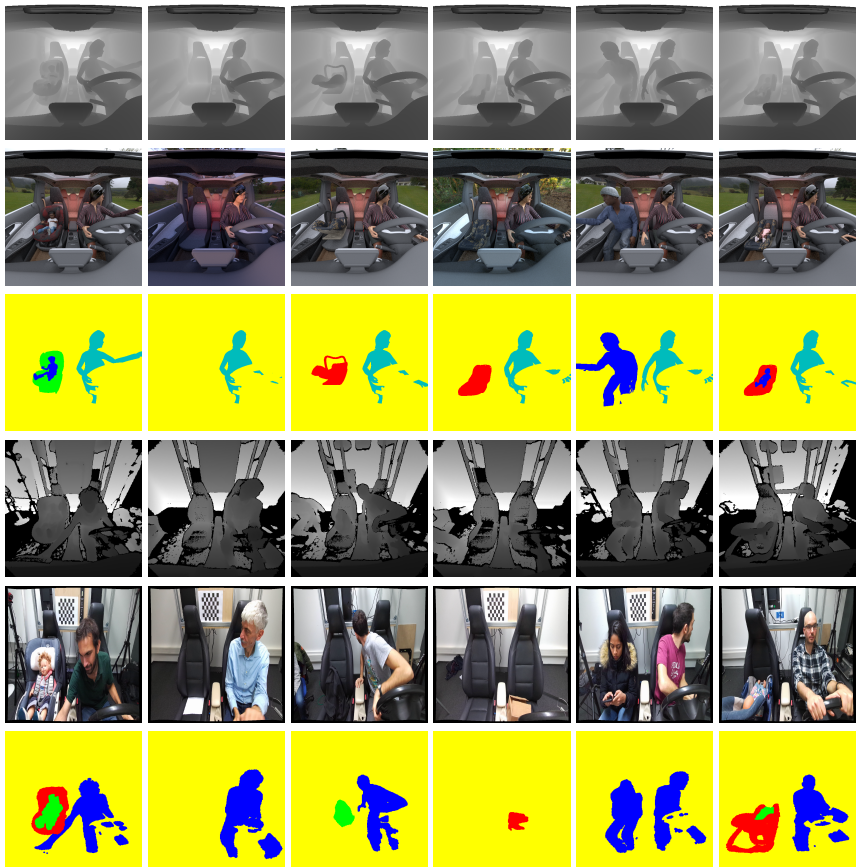


Figure 1: Example depth, RGB and ground truth segmentation mask images from TICaM.

with the best performing checkpoint is reported. We use SGD with initial learning rate of 0.0005, momentum 0.9 and weight decay of 5×10^{-4} . During finetuning, we first train the model on synthetic imageset using same settings for 12 epochs, then freeze the first 7 blocks of ResNet architecture and train again for 12 epochs on real images. We use the provided images as they are without any hole filling.

3.2 Faster R-CNN Architecture and Training Details

We adapt PyTorch implementation of Faster R-CNN from [14] for our dataset. We use VGG-16 [15] network as the backbone feature extractor. We initialize the backbone feature extractor with weights pretrained on ImageNet [16] and train the network for 10 epochs on single GTX 1080 GPU with a batch size of 4. The results with the best performing model checkpoint are presented. We use SGD optimizer with initial learning rate of 0.0001, momentum 0.9 and weight decay of 5×10^{-4} . During finetuning, we first train the model on synthetic imageset using same settings for 10 epochs, then freeze the first 24 layers of the VGG network and train again for 10 epochs on real images.

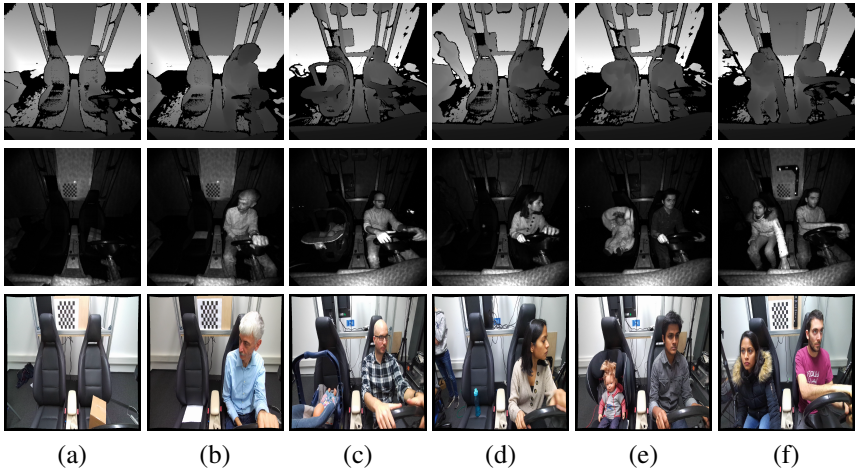


Figure 2: Depth and IR images of different driving scenarios provided in TICaM (a) Object on driver seat (b) Only driver (c) Driver and infant in RF (d) Driver and object (e) Driver and child in FF (f) Driver and Passenger

4 Additional Qualitative and Quantitative Results

We adapt PyTorch implementation of Faster R-CNN from [10] for our dataset. We use VGG-16 [8] network as the backbone feature extractor. We initialize the backbone feature extractor with weights pretrained on ImageNet [9] and train the network for 10 epochs on single GTX 1080 GPU with a batch size of 4. The results with the best performing model checkpoint are presented. We use SGD optimizer with initial learning rate of 0.0001, momentum 0.9 and weight decay of 5×10^{-4} . During finetuning, we first train the model on synthetic imageset using same settings for 10 epochs, then freeze the first 24 layers of the VGG network and train again for 10 epochs on real images.

We present in table 1 Average Precision (AP) for each class for the boxes and masks predicted by YOLACT. As mentioned in the main paper, finetuning the model on real data after pretraining on the synthetic imageset yields slightly lower performance, and this loss is mainly due to classes 'FF' and 'object'. Table 2 presents per class AP of the detected boxes by Faster R-CNN for the three training strategies described in the main paper.

Figure 4 and 5 present YOLACT's mask and Faster R-CNN's box predictions respectively when trained on *train_real* (column 1), *train_syn* (column 2) and finetuned on *train_real* (column 3) imagesets.

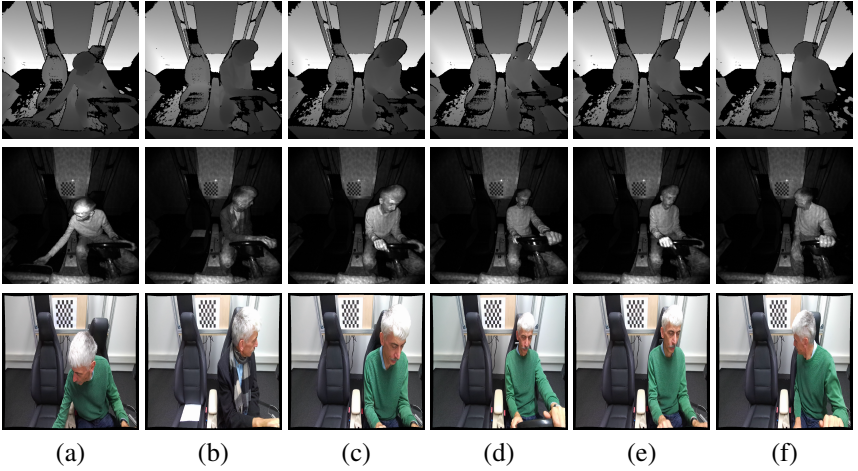


Figure 3: Figure 2(a) and 2(b) show a person in different clothing accessories. 2(c) and 2(d) show driver in different seat positions. Different driving actions are shown through Figure 2(a)-2(f).

Table 1: Per class Average Precision AP at IoU=0.5 for predicted masks and boxes by YOLACT for different training splits.

	Box mAP			Mask mAP		
	<i>train_real</i>	<i>train_syn</i>	<i>finetuned</i>	<i>train_real</i>	<i>train_syn</i>	<i>finetuned</i>
Person	98.93	87.23	98.96	99	85.44	99
Object	75.95	48.18	53.52	71.83	0	64.19
Infant	72.76	0	75.24	85.21	2.94	86.23
FF	100	15.09	95.08	59.7	11.52	68.15
RF	98.97	2.10	99.92	98.97	2.16	99.92
Child	100	13.46	100	100	12.35	100
mAP	91.11	27.67	87.12	85.78	18.99	86.25

Table 2: Per class Average Precision AP at IoU=0.5 for predicted boxes by Faster R-CNN for different training splits.

	<i>train_real</i>	<i>train_syn</i>	<i>finetuned</i>
Person	91.30	43.78	91.4
Object	62.11	0	49.9
Infant	71.16	2.70	100
FF	100	27.34	97.2
RF	98.13	21.64	100
Child	99.94	2.64	70.5
mAP	87.1	16.4	84.8

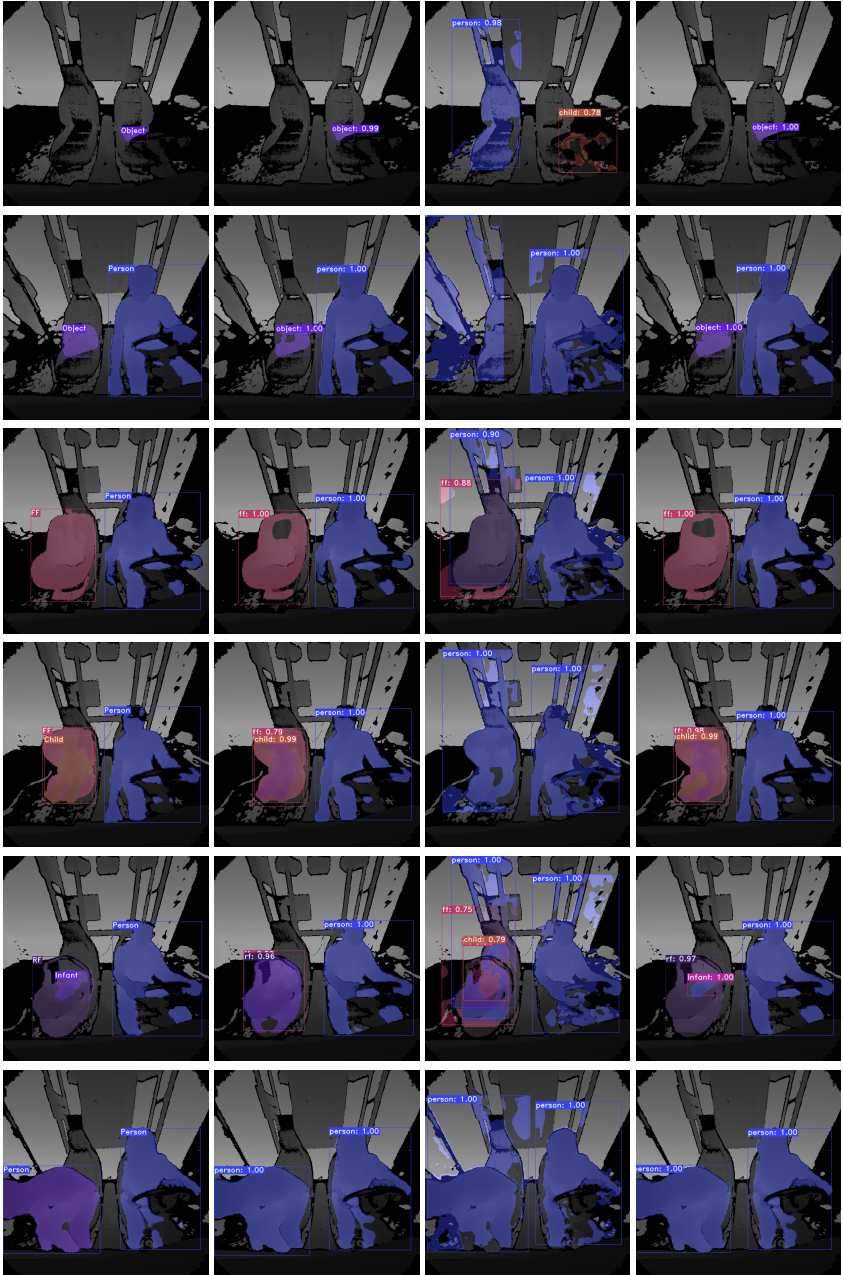


Figure 4: Depth images showing masks predicted by YOLACT when trained on depth images from *train_real*, *train_syn* and finetuned on *train_real* imagesets.



Figure 5: Depth images showing boxes predicted by Faster R-CNN when trained on depth images from *train_real*, *train_syn* and finetuned on *train_real* imagesets.

References

- [1] Faster r-cnn pytorch implementation. <https://github.com/jwyang/faster-rcnn.pytorch/tree/pytorch-1.0>.
- [2] Yolact pytorch implementation. https://github.com/feiyuhuahuo/Yolact_minimal.
- [3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. pages 770–778, 06 2016. doi: 10.1109/CVPR.2016.90.
- [5] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. 12 2016.
- [6] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.