Supplemental Material to “Drive&Act: A Multi-modal Dataset for Fine-grained Driver Behavior Recognition in Autonomous Vehicles”

Manuel Martin[*1] Alina Roitberg[*2] Monica Haurilet[2] Matthias Horne[1]
Simon Reiss[2] Michael Voit[1] Rainer Stiefelhagen[2]

[1]Fraunhofer IOSB, Karlsruhe [2] Karlsruhe Institute of Technology (KIT)
[*] equal contribution

Abstract

This supplemental document describes additional details about our dataset, baseline methods and evaluation results. It is structured as follows:

Section 1 *presents additional details about our data collection setup.*

Section 2 *describes the parametrization of our baseline models.*

Section 3 *shows sample images of our dataset and their annotation.*

1. Experimental setup

Although we collected our dataset in a driving simulator we aim to keep the surroundings as close to the real driving experience as possible. To achieve this our simulator is equipped with a real car instead of a simplified mockup. This section provides additional details about this setup and the collected data.

1.1. Driving simulator

Figure 1 shows the surroundings of our simulator. The simulator vehicle is an Audi-A3. It is surrounded by three projection screens achieving a simulated field of view of about 200 degrees for the driver of the simulator. In addition we position a flat screen behind the vehicle to facilitate simulation of the center mirror and the view back through the center of the vehicle. The outer mirrors are also modified with displays to enable simulation. Steering wheel simulation and feedback is provided by an electric motor on the drive shaft. All parts of the interior are modified to function properly in the simulation. In addition the dashboard is heavily modified to maximize screen space for future experimentation (see Figure 2).

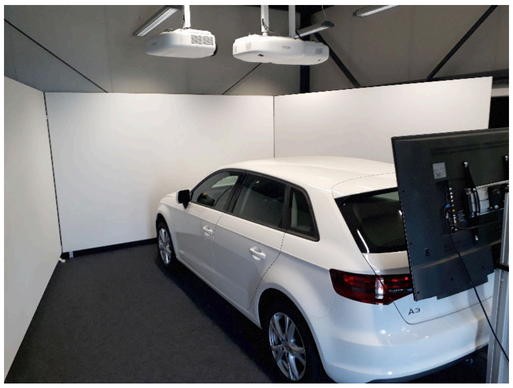


Figure 1: Overview of the simulator setup including the projection screens.

1.2. Camera setup

Figure 2 shows the interior of the simulator annotated with the position of all cameras used in the experiment. We designed the camera setup this way with different goals in mind. The setup contains a close approximation of all views suitable for monitoring the entire driving area both with regard to the viewing angle as well as integration of a future miniaturized commercial product. This enables investigation of each view separately to achieve the best possible result as well as experimentation for generalization to multiple views. In addition we also provide a detailed view of the drivers face with a camera mounted behind the steering wheel because this view is very popular for facial analysis and eye gaze estimation but provides challenges for action recognition because of its limited field of view. We also wanted to use triangulation to determine the 3d body pose, which is possible with this setup.

All views are at least equipped with a NIR-camera. We add active illumination via infrared light at $850nm$ and equip each camera with a narrow bandpass filter for this wave to be as independent as possible from external illumination. In addition we use a Microsoft Kinect v2 on the

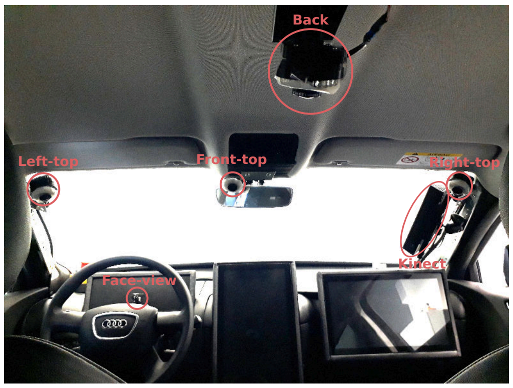


Figure 2: Interior of the simulator depicting the modified dashboard and camera positions.

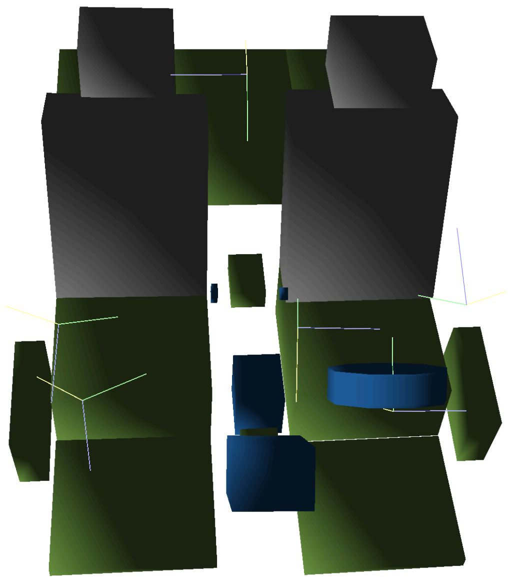


Figure 3: The interior model of the car. Green denotes storage areas, blue denotes car controls and gray remaining regions. Camera positions are depicted as coordinate systems.

co-driver side. We chose this depth camera because of its high depth resolution. However, integrating it without obstructing the drivers view is challenging because of its size and small viewing angle. The sensor is therefore positioned at the point furthest away from the driver that still provides a good view of the area.

1.3. 3D pose representation

Figure 4 show the body joints that are reliably provided in our dataset. The used detector OpenPose provides a more detailed representation of the whole body but because of occlusion the respective body parts are never visible or only visible for a short time. We provide all data generated by OpenPose but we report only reliably detected joints.

1.4. 3D interior representation

Figure 3 shows the interior model of the car provided with the dataset. It is built out of cylinders and cubes so it is easy to use and reason about compared to a more detailed representation with a 3D mesh. We built this model by using the point cloud of the Kinect as a guide to manually position each primitive. Movable parts like the seats are adapted for each sequence. Overall our model consists of 21 named primitives. It contains all parts of the car interior with semantic meaning for the actions in our dataset.

2. Model Parametrization

In the following we describe the structure and training parameters of all our baseline models. We train separate models for each of the three annotation levels. The annotation of atomic actions consists of triplets. We train three separate networks, one for each part of the triplets, because we found that this improves results compared to training a single model with three output layers.

2.1. End-To-End Models

We train our networks end-to-end using stochastic gradient descent (SGD) with momentum, employing early stopping on the validation set with the maximum of epochs set to 150. The video streams of all views and modalities are first rescaled to 252×256 pixel, and further augmented with random cropping to obtain the resolution needed for the specific network, as done in [4]. The resolution of the final crop used as input to our network is 224×224 for I3D, 160×160 for P3D Resnet and 112×112 for C3D. For testing, we follow the standard practice of replacing the pre-processing pipeline with a center crop[4]. The processing of the temporal dimension was adapted based on the capabilities of the respective model (16 frames for C3D and P3d ResNet and 64 frames for I3D). Frames are randomly selected from the provided three seconds long video samples (see Section 3.3 of the main paper). The network architecture-related parameters (e.g.dropout, input size) were set according to the original architectures[4, 8, 7]. We apply a weighted sampler to balance the dataset classes for training.

Inflated 3D ConvNet We use the Pytorch [6] implementation of the Inflated 3D architecture (I3D) [4] with its pre-trained weights on the Kinetics dataset provided by [2]. We set the initial learning rate to 0.01, and divide it by a factor of 10 after 50 and 100 epochs. We use momentum of 0.9, weight decay of 1e-7 and a mini-batch size of 8. During training, temporal data augmentation samples clips of 64 frames and spacial data augmentation computes random crops of size 224×224.

P3D ResNet We use the Pytorch [6] implementation of the Pseudo 3D ResNet (P3D) [7] with its pre-trained weights on the Kinetics dataset provided by [3]. We set the

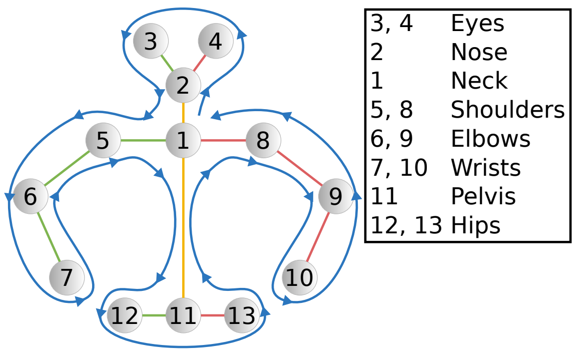


Figure 4: Annotated body joints and joint sequence used for the spatial stream

initial learning rate to 0.01, and divide it by a factor of 10 after 50 and 100 epochs. We use momentum of 0.9, weight decay of 1e-7 and a mini-batch size of 16. During training, temporal data augmentation samples clips of 16 frames at a random position, and spacial data augmentation computes random crops of size 160×160.

C3D For C3D [8], we use the Pytorch[6] implementation with its pre-trained weights on the Sports 1-M dataset provided by [1]. The initial learning rate is set to $3 * 10^{-4}$, and divided by half after every 20 epochs. We use momentum of 0.9 and a mini-batch size of 16. Similar to the original paper [8], we do not use weight decay in this architecture. During training, temporal data augmentation samples clips of 16 frames at a random position, and spacial data augmentation computes random crops of size 112×112.

2.2. Body Pose and Car-Interior Architecture

As described in the main paper the core model for each stream of the body pose based action recognition system is the same. It consists of two recurrent layers with 512 LSTM-units followed by a fully connected layer with Softmax activation. All models are trained with the Adam optimizer [5] using the default parameters of Keras. Models are trained for 100 epochs, employing early stopping on the validation set. For training we use a batch size of 128 samples selected balanced from the training dataset. We use a fixed history of 90 frames. If the annotation is shorter than 90 frames we pad the beginning with zeros. To combine different streams we employ weighted averaging. We compute the weights on the validation set using grid search. In contrast to the end-to-end models no pre-training or data augmentation is used. In the following we describe the input of each stream in more detail.

Temporal Stream The input of the temporal stream consist of the concatenation of the 13 main joints depicted in Figure 4 resulting in a input feature with 39×90 dimensions.

Spatial Stream The joint sequence used for the spatial stream is depicted in Figure 4. For each joint in the sequence we concatenate all joints in the 90 frames window resulting in a input feature with 270×23 dimensions.

Interior Stream The input feature of the interior stream consists of distances of body joints to the surface of the primitives of our interior model. We focus on the body parts that are most important for action recognition namely the hands and the head and compute the distance of all three joints to all primitives of the interior model. The dimension of the input feature of this stream is therefore 63×90.

3. Examples of individual activity classes

In the following, we provide examples of all annotated classes. We first describe all 34 concise fine-grained activities in Section 3.1. Then, we visualize frames with annotations of their atomic action units, which comprise of {*Action, Object, Location*} triplets in Section 3.2(6 actions, 17 objects and 14 locations respectively). Finally, in Section 3.3, we visualize an example segment for each of the 12 possible high-level scenarios/tasks.

3.1. Fine-grained Activities (level 2)

In Figure 5 and Figure 6 we provide an example image for each of the 34 fine-grained activity labels (visualization of Kinect RGB view for different subjects).

3.2. Atomic Action Units (level 3) and complete annotation examples

In Figure 7, we aim at covering all *atomic action units* and provide examples of their combinations, with *complete* annotations (all levels of granularity) The text below each recorded frame depicts our provided annotations: the *first annotation row* represents the *high-level scenario/task* (level 3); the *second row* provides additional annotation of the *driving context* (e.g.*steering left hand*); the *third row* is the annotation of the *fine-grained activity* (level 2); the *fourth, fifth and sixth rows* represent the annotation of *atomic action units*, standing for the *action*, *object* and *location* of the current interaction.

3.3. Scenarios/Tasks (level 1)

We further visualize example progression for each of the twelve high level scenarios/tasks (Figure 8 and Figure 9). Due to the longer duration and compositional nature of the tasks, we visualize multiple steps that took place and provide the corresponding annotation of the fine-grained activity of this step. The way of completing the task and therefore the fine-grained activities that took place during its execution, was left to the subject. Fine-grained activities and their order in Figure 8 and Figure 9 are therefore just examples of how the person decided to accomplish the task.

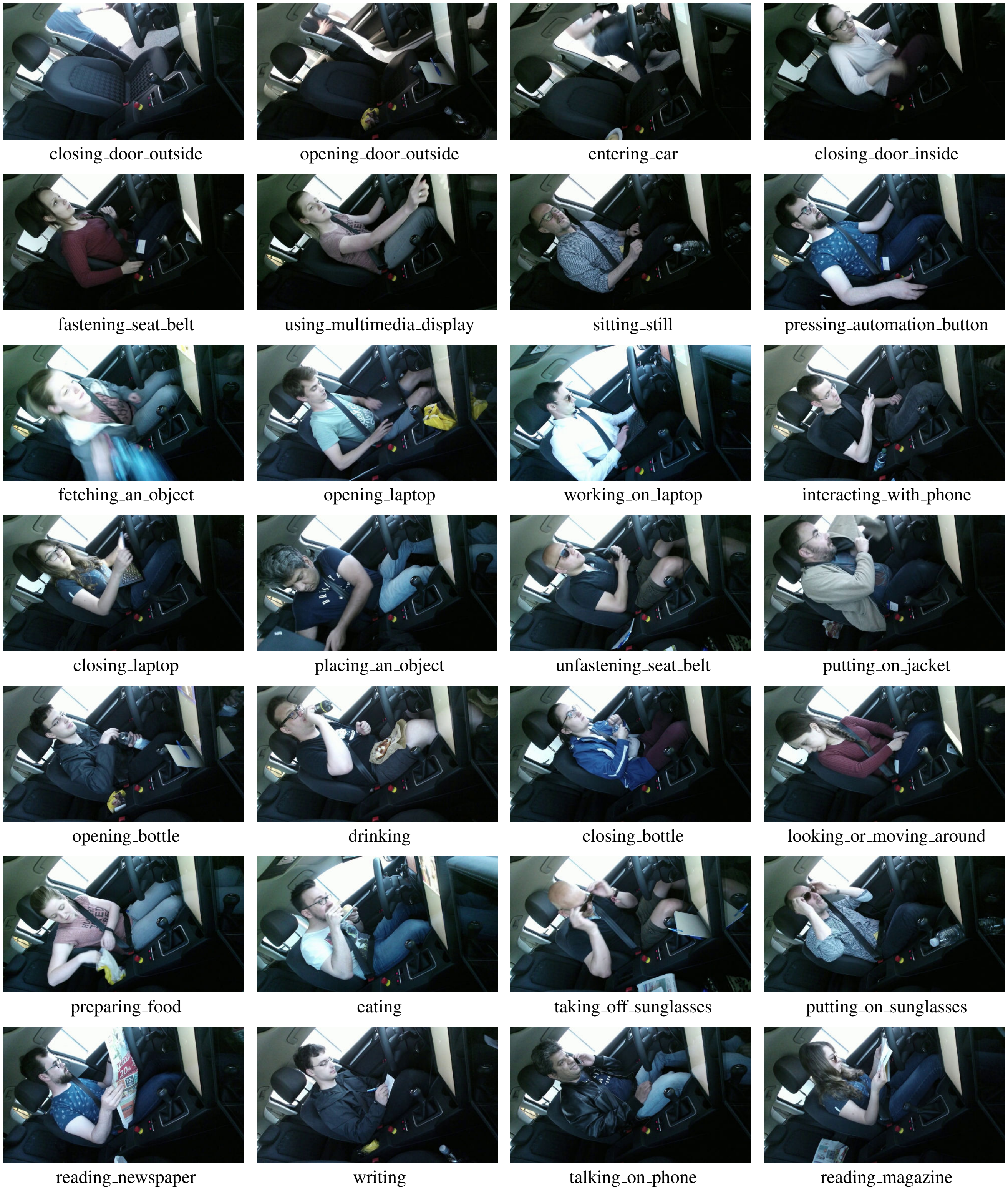


Figure 5: Sample images extracted from video files, displaying all 34 *fine-grained activity* classes.

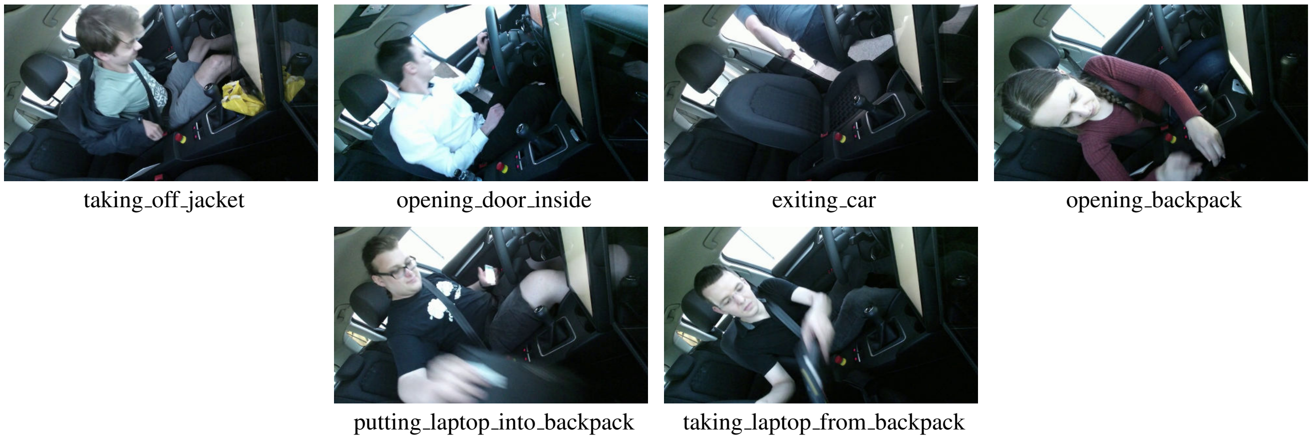
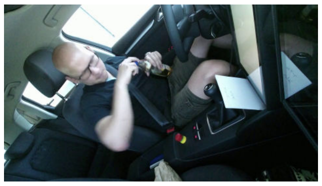
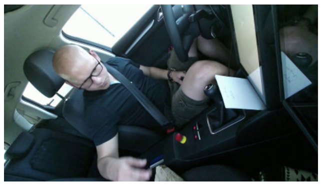
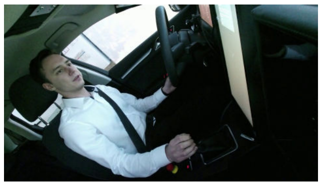
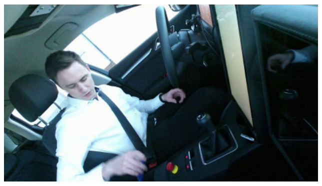


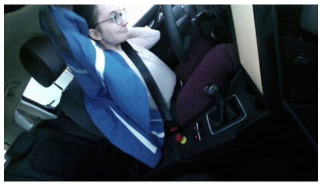
Figure 6: Continuation of figure 5

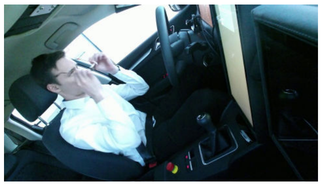

eat_drink
–
closing_bottle
closing
front_area
bottle

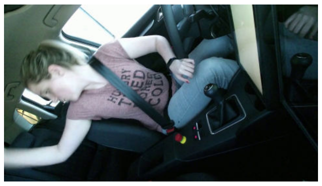

eat_drink
–
fetching_an_object
reaching_for
codriver_seat
food

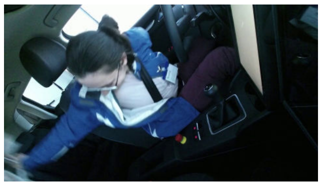

driving_preparation
steering_left_hand
–
interacting
–
gearstick

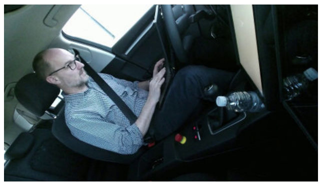

–
–
placing_an_object
placing_moving_to
center_console_back
pen

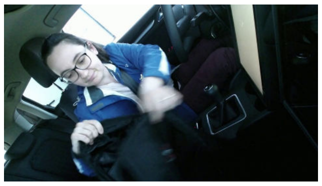

put_on_jacket
–
putting_on_jacket
interacting
front_area
jacket

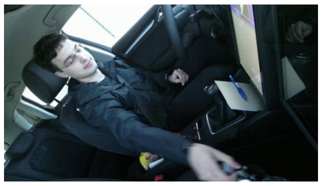

put_on_sunglasses
–
putting_on_sunglasses
placing_moving_to
head
glasses


read_magazine
–
fetching_an_object
reaching_for
right_backseat
writing_pad


read_newspaper
–
fetching_an_object
retracting_from
left_backseat
newspaper


work_on_laptop
–
working_on_laptop
interacting
lap
laptop


work_on_laptop
–
opening_backpack
opening
codriver_seat
backpack


watch_video
–
placing_an_object
placing_moving_to
codriver_footwell
bottle

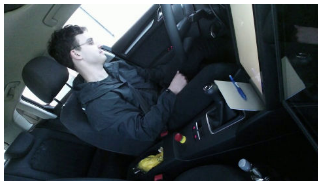

put_on_sunglasses
–
fetching_an_object
retracting_from
driver_door
glasses_case

Figure 7: Sample images covering different instances of *atomic action units*, provided with **complete annotations**. Text below the images are annotations for each level. **First row:** high-level *scenario/task* (level 3); **Second row** : additional annotation of the *driving context* (e.g.*steering left hand*); **Third row**: *fine-grained activity* (level 2); **Fourth, fifth and sixth rows**: *atomic action units*, standing for the **action**, *object* and *location* of the current interaction. Note, that - depicts no present activity at the corresponding level.

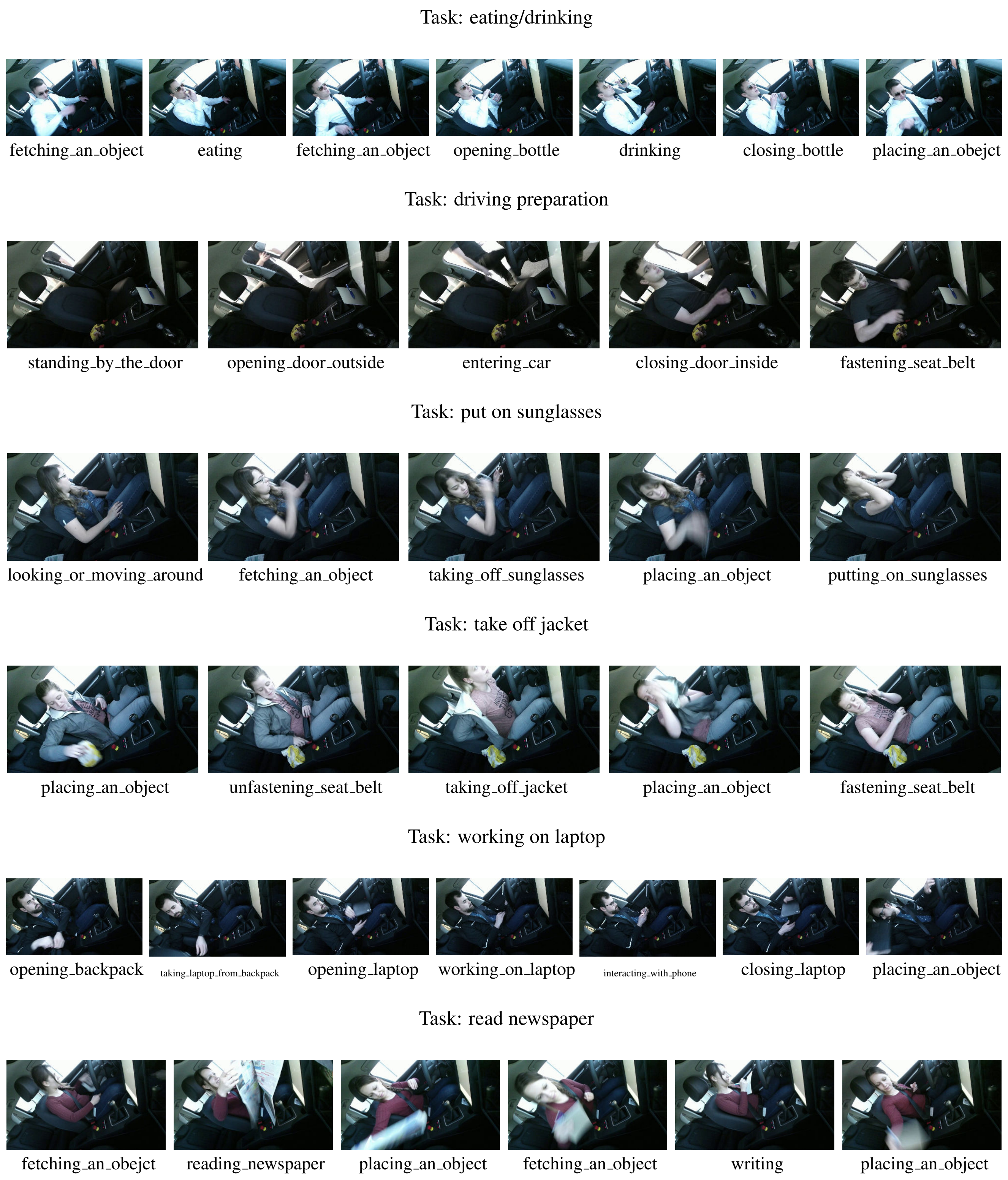
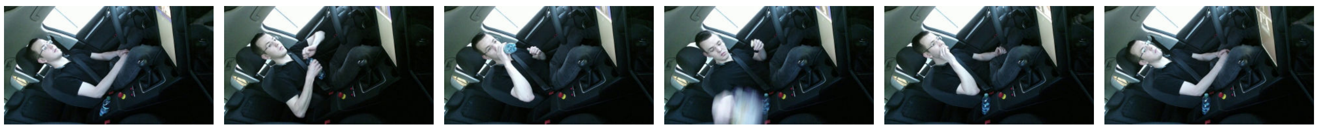
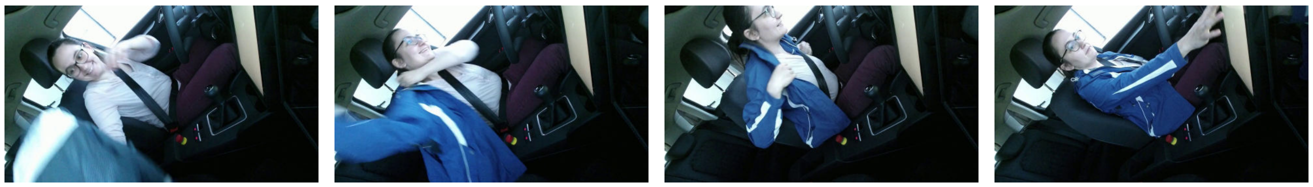


Figure 8: Example sequences taken from different subjects solving each of the twelve *tasks*. Caption over the sequence row depicts the current task, while annotations under the images show the *fine-grained activities* that occurred at this step.

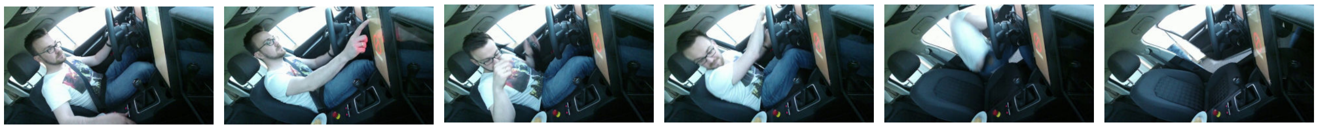
Task: watch video



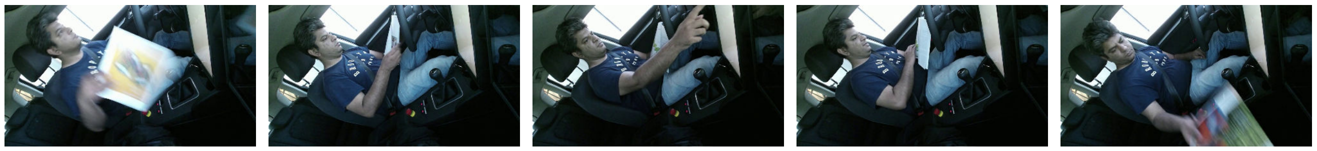
Task: put on jacket



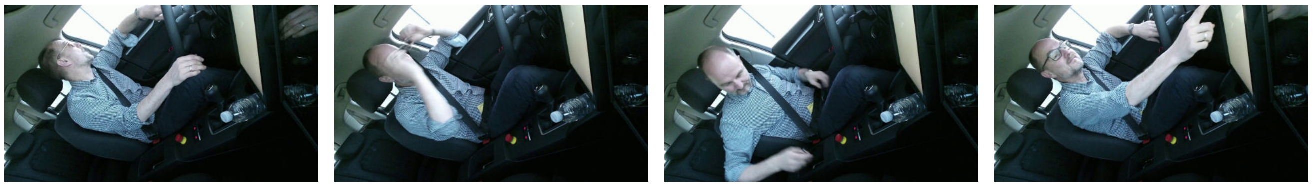
Task: park car and exit



Task: read magazine



Task: take off sunglasses



Task: take over steering

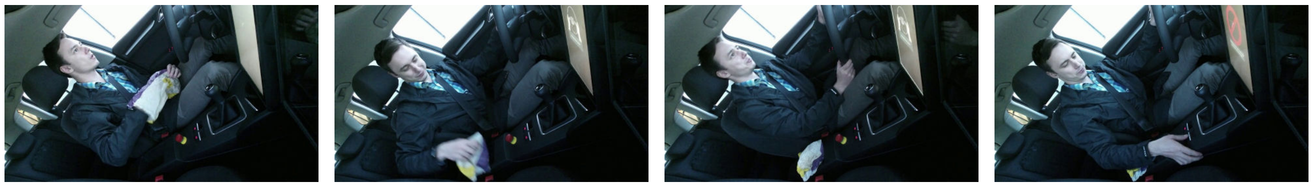


Figure 9: Continuation of Figure 8. For the task *take over steering* the *fine-grained activity* is left blank as steering is mentioned under *additional annotations*.

References

[1] C3D implementation in Pytorch. https://github.com/DavideA/c3d-pytorch. 3

[2] Inflated 3D Network (I3D) implementation in Pytorch. https://github.com/hassony2/kinetics_i3d_pytorch. 2

[3] Pseudo 3D Resnet implementation in Pytorch. https://github.com/qijiezhao/pseudo-3d-pytorch. 2

[4] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 6299–6308, 2017. 2

[5] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*, Dec. 2014. arXiv: 1412.6980. 3

[6] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 2, 3

[7] Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *Proceedings of the International Conference on Computer Vision*, pages 5533–5541, 2017. 2

[8] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the International Conference on Computer Vision*, pages 4489–4497, 2015. 2, 3