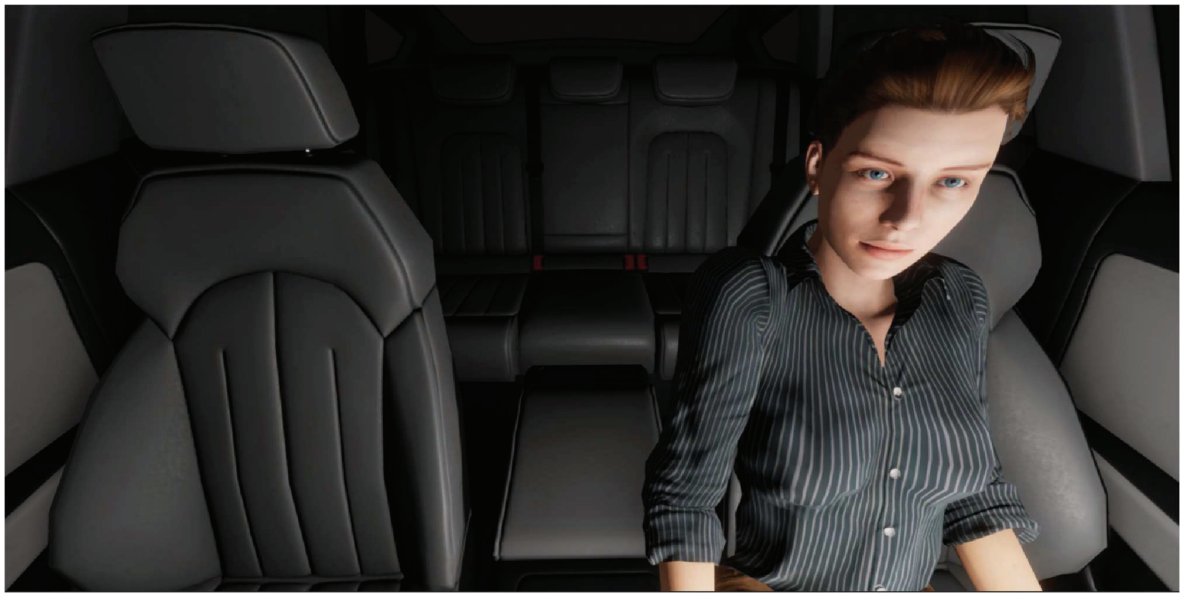
DRAMA-2

Publik rapport



Författare: Boris Duran, Martin Torstensson, Thanh Hai Bui, Svitlana Finer, Henrik Lind, Lorenzo Mirante, Mattias Torstensson, Jonathan Bergqvist, Ted Gustafsson
Datum: 20230130
Projekt inom Elektronik, mjukvara och kommunikation

FFI Fordonsstrategisk Forskning och Innovation
VINNOVA Energimyndigheten TRAFIKVERKET FKG SCANIA VOLVO

Innehållsförteckning

1 Sammanfattning

DRAMA-2 bygger på de erfarenheter som erhållits från DRAMA-projektet där tillämpning av Machine Learning (ML) -algoritmer användes för att övervaka beteendeen och interaktioner. En av de tydliga flaskhalsarna som identifierades i det första DRAMA-projektet var svårigheterna kopplade till att samla in tillräckligt mycket kvalitativ data för att träna nätverken. Dessa utmaningar är direkt kopplade till mängden tid, resurser, och personal som behövs för att samla in dataset från verkliga miljöer.

Syftet med detta projekt är att skapa en simuleringsmiljö där högkvalitativa (fotorealistiska) bilder kan samlas in från en simulator och på så sätt kan vi kringgå de hinder som finns när man samlar in data i den verklig miljö. Projektet kommer också implementera en kinematisk beteendemodell över hur människor rör sig inne i fordonet.

2 Executive summary in English

DRAMA-2 builds upon the experiences obtained from the DRAMA project[1] on the application of Machine Learning (ML) algorithms for monitoring the behavior of vehicle occupants. One of the major bottlenecks observed in the DRAMA project was the difficulty of collecting enough amount of high-quality data for training these networks. The reasons for these problems are directly related to the amount of time, funds, equipment and personnel required for collecting these datasets in real environments.

DRAMA-2 creates a simulated environment where high-quality photo-realistic images can be generated and collected without the constraints of the real world in a more efficient way. The proposal also wants to implement an inverse kinematics model of human motion for the generation of in-cabin movements by the occupants of the vehicle.

3 Background

Vehicle automation through improved ADAS and implementation of different SAE levels calls for strong and accurate interplay between the driver and the vehicle. Situations where the responsibility of driving is transferred between driver and vehicle are particularly challenging. Recognition of human behavior inside vehicles is becoming increasingly important. Paradoxically, the more control the car has (i.e. in terms of ADAS), the more we need to know about the person behind the wheel[2] especially when he/she is expected to take over control from autonomous modes in vehicles. DRAMA-2 is supporting the perception part of this understanding by laying the framework for using mixed training using real images with images from a synthesized environment.

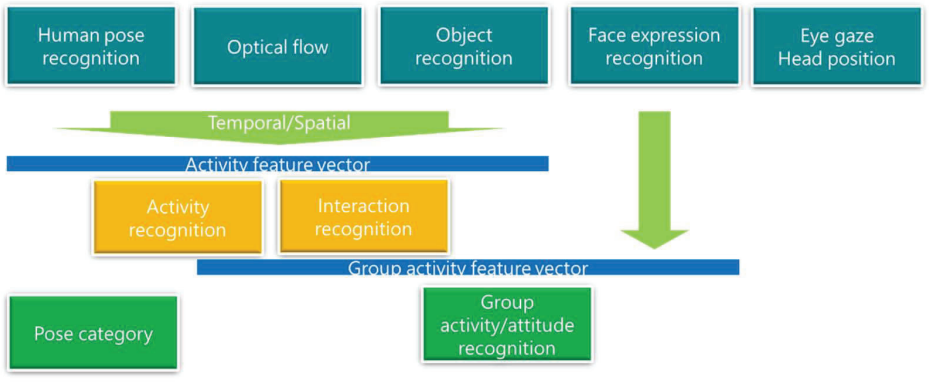


Figure 1: DRAMA activity recognition framework

In the specific case of perception, it is always desirable to be able to obtain a large number of images representing objects, people and the events describing their interaction. When talking about mapping the activities happening

inside the cabin of a vehicle, the correct classification of these events would allow us to, for example, decide when it is necessary and suitable to activate certain safety protocols, evaluate the emotional state of driver and passengers, or adapt aspects of human-machine interaction. Tracking face expressions, gestures and body position allows us to estimate the emotional state and responses of the driver and/or passengers which in turn can be used to evaluate the automated vehicle's actions in traffic. All these functions can improve the interaction between vehicle and driver. The better the vehicle knows its occupants, the safer and more comfortable the travel can be.

In DRAMA project[1], we developed a framework (Figure 1) and Proof-of-Concept (PoC) prototypes using multiple visual sensor settings together with deep learning techniques to detect and understand driver/passenger ongoing activities insides a car cabin. The framework acts on timeseries of activity sub-features e.g. face landmarks/expression, human body skeleton, objects, dense trajectories data was collected and annotated describing the activities.

Within this framework, activity sub-features are selected as the "spanning" set of high-level features that can be extracted from the input sensor data streams, forming a basis for understanding the entire space of activities/behaviors.

However, employing this framework in practical automotive applications shall require application-dependent specifications of activities in focus and their related sub-categories, as well as the corresponding annotated datasets for training, validation, and supervision.

Obtaining high quality and quantity of the data for training deep learning (DL) algorithms is arguably the biggest challenge in this domain. Currently there are only a few publicly available datasets that can be used to train deep neural networks (DNNs) and most research groups used them as benchmarks for comparing their results[3]–[8]. Since the focus list of activities and sub-categories are dependent on specific application requirements, it is often difficult to find public datasets that correspond to the various setups of sensors and use-cases, consequently, it is also a diligent job to collect data that represents the infinite amount of varieties according to the selected activities and subcategories [9].

4 Purpose, research questions and methods

In DRAMA-2, we will investigate a more efficient methodology to automatically collect large amounts of data representing a variety of objects, occupants, and their interactions. Figure 2 shows a block diagram summarizing the main differences between a conventional way of collecting data for training DL-based functions versus the approach in this project.

The methodology is aiming at generating training/validation data with configurable distributions (allowing us to enrich the data with situations that are rare, but important for the model to learn) while still maintaining the same properties of the real-word data space. The first component is:

- Generation of configurable realistic video data that simulate the sensor specifications and different environmental conditions (weather, illumination, different settings of cabin interior, …).

To model human activities and in-cabin objects, we use a motion capture system and extract the basic motion primitives and corresponding objects (holding a cup, eating a sandwich, etc.). The in-cabin simulator can then combine the primitives to generate realistic variations of human activities, i.e. the second component is thus:

- Generation of human activities by modelling body motions from basic motion primitives.

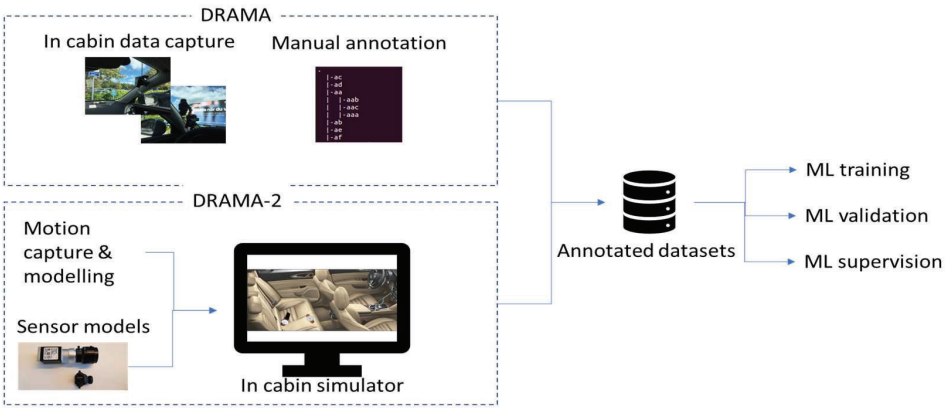


Figure 2: Diagram for conventional way of creating annotated data and DRAMA-2 approach

The following research questions are targeted within the project:

- RQ1: How can we generate sensor realistic datasets for training in-cabin human activity recognition system from synthetic data simulators?
- RQ2: How can we model in-cabin human body motion with regards to the specified activities? Can we use a 3D body to train the network with 2D image inputs to estimate accurate 3D joint positions in space?
- RQ3: How can we evaluate the quality and completeness of generated datasets?
- RQ4: How can we combine real-world and simulated datasets for training human activity and object detection?

5 Goal

The aim of DRAMA-2 is twofold (a) explore the possibility of synthetically generating annotated training datasets used in deep learning algorithms and (b) improve the human activity recognition models developed in DRAMA, as well as object detection models.

6 Results and Deliverables

6.1 Methodology for generating synthetic datasets for training cabin monitoring algorithms

The lack of real data is a challenging problem that is present most of the time during the training of complex ML systems. One of the goals of this project is to study how synthetic data can help to solve this issue.

In this section we discuss methodology for generating synthetic datasets for training cabin monitoring algorithms assuming that the simulator software that is used to create synthetic data is already chosen, set up and its requirements and limitations are known.

As any other task, creating synthetic dataset starts from **the specification of the synthetic dataset requirements**. First step is to decide what ML system will it be used for, what are the goals of that ML system, what tasks it should be trained for and what results we expect to obtain. Then, the description of the environment is prepared, to name a few: light conditions, outdoor landscape(s), vehicle type(s), vehicle interior, etc. Next, one should prepare information about the focal length, sensor size, distortion, and noise of the camera to create realistic synthetic images. One should also describe number of cameras and camera position(s), as well as what kind of light should

the camera(s) capture (ambient or infrared - IR). Once the environment is prepared, and the next step is to prepare objects and people in the vehicle. Besides deciding the number of the objects and people, one should prepare description of how they should be placed and interacted if any. Should the objects be placed on the seats, on the floor, in the hands of people, etc.? Should people sit in the seats and have different poses or the same ones, should they hold objects or be empty handed, should they interact with the vehicle, etc.? It should also be decided whether to store single images or series of images. Another important step is to decide whether the synthetic data should be created as a standalone dataset used for the ML system or should it be combined with the datasets collected in real life (hereafter referred to as real datasets)? Once this is clarified, the available real data is studied. In this process, it is important to look for imbalances in the information given by the real data, and for edge case scenarios that are hard to reproduce when capturing the real dataset. The purpose of synthetic data should be to fill these “holes” in the real dataset, to help the ML system to learn its tasks, by maximising the domain related “information content” of the data. In addition to the data specification, one must prepare storage space where the synthetic data will be temporary or permanently stored.

Once the specification of the synthetic dataset requirements is done and the space is prepared, starts **the synthetic dataset creation and annotation**. Even though the same simulator software could be used to create the synthetic data the pipeline could be specific to different datasets. With that in mind, the pipeline is checked and set up. Creating data in a simulated environment means that the user has complete control over the data and its distributions. Therefore, all the possible variables related to the data are known and can be stored as metadata together with the data. This allows us to reproduce the exact same data or tweak certain parts of it while keeping all others the same, as well as to create detailed data annotations. This data specification is crucial to control performance of supervised ML systems, which are sensitive to the training datasets and require extensive annotations.

Once the data is synthetized, the next step is **the quality check and adjustment of the synthetic dataset**. There are different methods that can be used. The simplest one that can be a good start is, of course, visual inspection. If there is a real dataset that the synthetic dataset should be combined with, then a comparison of the likeness of the two could be useful.

Once this is ready, **the specification document of the synthetic dataset** should be prepared. After that this synthetic dataset can be released with the version number and used in ML system.

Usually, creation of the synthetic dataset is not a one-way process that goes straight from the request of the data to the use of that data in the final training of the ML system. Many iterations of the synthetic data are made over time improving different aspects of it, based on the feedback obtained during the training of the ML system.

6.2 Simulator for generating synthetic datasets

6.2.1 Base simulator

Object detection datasets and action detection datasets were generated in separate simulators, but they have a common base, as will be described in this section.

There are multiple car interiors available that can all be assigned probabilities of being the active car in each data point. In addition to swapping out entire car interior designs, the front seats can be replaced with seats from the other car models, and their position can also be adjusted. Each car also allows for the definition of camera placement positions so that the camera can be in a similar spot, such as on the dashboard, no matter how much the cars' interiors differ.

The simulator allows for multiple cameras to be active simultaneously and it will do one set of renders for every camera allowing for multiple different perspectives of the same scene. Blender's cameras can approximate real camera distortion allowing for the creation of a dataset that has multiple different intrinsic camera parameters. Camera mounting location and projection angle can also be randomized to simulate variance in the exact placement of real cameras. Along with the customizable camera there's also the ability to customize and randomize lights tied to the camera that function as a camera flash.

Global illumination is simulated by randomly choosing a real 360° high dynamic range photo as the background for the scene, this creates realistic looking lighting conditions and defines the environment outside the car's interior. By varying illumination sources, intensities, further variance is acquired.

With the car and environment set up, objects and people are positioned inside of the car.

Objects can be placed onto empty seats, or they can be spawned from customizable volumes inside the car. After spawning all objects, a rigid body simulation is run in order to place all objects neatly onto the surfaces inside the car. The probability for every object being put in the car can be customized, allowing for customized probabilities for each object class label.

People are placed differently for the object detection and for the action detection simulators.

To add more variance to the simulator it is possible to assign materials to any surface (mesh) in the scene, and it is possible to randomly change the appearance of these materials.

Simulators for generating both object and action detection datasets initially produce RGB images that are then postprocessed to recreate specific camera properties (such as sensor noise and lens blur). Unity (SW used for generating object detection datasets) has the option to add noise proportional to the lighting of the image. In Blender (SW used for generating object detection and action detection datasets) the noise is added during the postprocessing phase, randomizing it equally all over the image. In this project synthetic datasets are combined with the real datasets, so the synthetized images for the object detection simulate NIR light and fish-eye lens effect, while the synthetized images for the action detection simulate RGB light and no fish-eye lens effect.

Along with RGB images simulators also produce metadata files that contain object segmentation masks of every object and person in the scene with a better accuracy than manual annotation. These files contain class labels and the objects'/person’s places in 3D space and in 2D space (in the form of bounding boxes). These annotations can be later used as a ground truth for the development of the DNN algorithms.

6.2.2 Simulator for creating object detection dataset

One of the differences between the simulator for generating object detection datasets and simulator for generating action detection datasets is the addition of people to the vehicle environment.

In the simulator for generating object detection dataset, people are randomly picked from a library and then randomly assigned to empty seats inside of the active car. Which seats and people that are available as well as the probability of using them is customizable. The peoples' meshes are fully rigged and get assigned randomized poses, moving all limbs while remaining seated. The people can also be assigned to hold objects in either of their hands, with the simulator automatically posing the characters’ fingers so that they connect with the surface of the object and grip it. Both 2D and 3D body pose data is created for every active character.

In this project 2 different software were used and compared for generating synthetic datasets of images for the object detection, namely Unity and Blender. Blender is a free open-source software, while Unity has free and paid versions, later with more features. Table 1 outlines the main differences between the two types of the synthetic datasets produced by the respective software, as well as the real datasets. While the Unity dataset simulate real world actions, and places objects in the hands of people, the Blender dataset has a focus on a better photorealism for the lighting conditions, the models of the objects and the people in the environment. Blender dataset has more variety of the objects, and in addition to the objects belonging to the detected classes it includes objects that do not belong to any of the detected classes. This variety of objects allows object detection algorithm to get more information and become more robust.

Table 1: Dataset feature comparison

Dataset feature	Unity	Blender	Real
People	Yes	Yes*	Yes
Detected objects	Yes	Yes*	Yes
Not detected objects / Objects categorized as “Other”	No	Yes	Yes
Vehicle model and interior variation	Yes	Yes	Yes
Environment outside of the vehicle variation	Yes	Yes*	Yes
Light variation	Yes	Yes	Yes
Objects held in hand by people	Yes	No	Yes
Simulation of actions	Yes	No	Yes
Photorealistic lighting	No	Yes	Yes
Random noise proportional to the environmental light	Yes	No	Yes

* Additional object models are available compared to the other synthetic dataset.

6.2.3 Simulator for generating action detection dataset

6.2.3.1 Realistic movement Part 1 (use body key joints from real movements)

Datasets for the action detection were created in Blender.

The simulator for generating action detection dataset renders image sequences of a single person in the car's driver seat performing various actions. The action movements are the recreations based on the 3d body pose data extracted from recordings of real people performing those actions (MOCAP- motion capture), which are then applied onto the synthetic characters. Through this method one can map a single real action's recording onto an arbitrary amount of synthetic people, where the synthetic people also have arbitrary features such as body shape, clothing, etc. Objects used to perform the action in real life, such as drinking from a cup, in the synthetic dataset can be substituted with other objects, such as a soda can or a water bottle.

6.2.3.2 Realistic movement Part 2 (use basic motion primitives to model body motions)

6.2.3.2.1 Introduction

One of the main goals of this project is the generation of relevant and high-quality training datasets which in turn will be used in the development of machine learning algorithms for activity recognition. To better understand the different modules of the proposed methodology we will make use of the terminology used in software development for the creation of front- and back-end components.

In the context of this project, we could call front-end development to the steps taken for generating the final images and sequences which are going into the training and testing of ML algorithms. The creation and properties of these datasets are well described in the previous sections however it is important to remember that all those images should contain samples of humans, objects, and their interactions in “realistic” contexts. In other words, all the images should represent situations or events that are as close as possible to situations and events happening in the real world. For example, an image of a person reaching for an object should depict the reaching arm with the respective natural limitations of length and rotations that we find in real humans. To achieve this, it is necessary to control the way all joints move individually and as a whole.

The back-end part of this development focuses on the methodology for controlling the motion of these virtual characters. Since we are only interested in the visual representation of human motion, we will limit the scope of our

work to the development of kinematic models and will not include dynamic models which take into consideration more complex interactions of forces and torques.

Kinematic models for human-like robotic manipulators have been studied for a long time and from different research areas. The task of reaching different positions in 3D space with a multi-link serial manipulator can be solved analytically and algorithmically without many problems in today's computational capabilities. However, these methodologies do not take into consideration the quality of motion, i.e. how similar to the trajectories performed by real humans both in space and time. In order to achieve human-like natural movements we divide this problem into two parts: finding an inverse kinematics solution and recreate natural movements.

There are a limited number of activities that can be studied inside the cabin of a vehicle, a short list with some examples of these activities is presented below:

- Steering wheel
- Gaze phone
- Gaze food/drink
- Blink
- Sleep
- Reach phone
- Reach food/drink
- Bring phone
- Bring food/drink
- Gesture
- Texting

Motor primitives are simple building blocks that can be combined and transformed to create larger sequences of more complex actions. From the list above, activities such as "reach phone" and "reach food/drink" share the same primitive "reach"; the same applies for "bring" and "gaze". The methodology proposed for modelling human activities in this project starts by modelling motor primitives; since the same approach is used for all of them, we decided to work on the "reach" motor primitive to prove the concept.

6.2.3.2.2 The Kinematic Chain

In order to better understand the implementation of an inverse kinematics model for reaching behaviors in the current use case, we decided to simplify the kinematic chain to be studied. All virtual characters built inside the simulator contain 26 joints; however, each joint represents a maximum of three different rotations, also known as degrees-of-freedom (DOF), around local reference frames. For example, the elbow represents a 1-DOF joint whereas the shoulder can be considered as a 3-DOFs joint. Reducing the number of DOFs in the kinematic chain

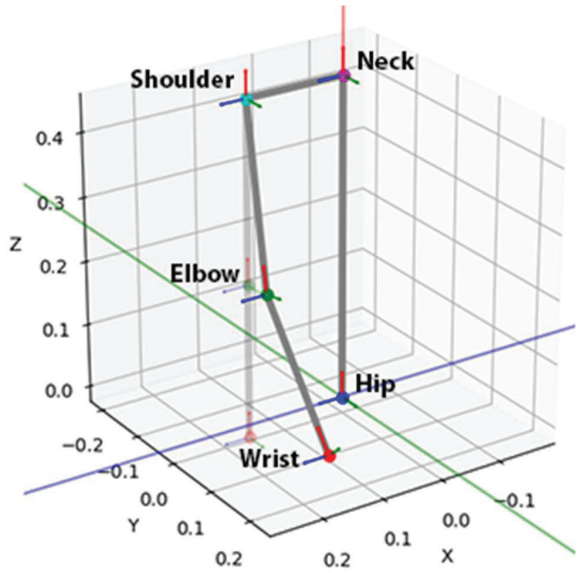


Figure 3: Diagram for the kinematic chain used in the study of inverse kinematic models for the generalization of reaching behaviors.

allows us to focus on developing a solid understanding of how a ML model can be trained for interacting with objects in a reduced space such as the interior of a vehicle. A virtual character placed in the driver seat will mainly use the right arm; therefore, we will study the IK problem for a 5-joint kinematic chain: hip (world/fixed reference frame), neck, right shoulder, right elbow, and right wrist, Figure 3.

The hip joint is designed to rotate the link between the hip and the neck around the x-axis (pitch) and y-axis (roll); i.e., 2 x DOFs. The neck joint rotates the shoulder-neck link around the y-axis (roll) and the z-axis (yaw). The shoulder joint rotates the shoulder-elbow link around all 3 axes (pitch + roll + yaw). Finally, the elbow joint rotates the elbow-wrist link around the x-axis (pitch). In total, the kinematic chain to be used in the current model contains 8 DOFs, 4 links and 5 joints if we include the end-effector (wrist), see Figure 3.

6.2.3.2.3 Inverse Kinematics (GMM)

Virtual characters inside the simulator are designed with anthropomorphic configurations, i.e., links (torso, upper-arm, forearm, etc.) serially connected through joints (shoulders, elbows, wrists, knees, etc.). The task of finding the coordinates of joints in space given a set of pre-defined angles is known as Forward Kinematics (FK). However, the current use case attempts to solve the opposite problem, i.e., trying to find the angles that each joint needs to rotate in order to take the end-effector in reaching distance of pre-defined coordinates in the scene; this is known as the Inverse Kinematics (IK) problem.

The known variables are the positions of objects (a telephone, a book, food, the panel, etc.) inside the cabin of the vehicle where the virtual characters (kinematic chain) need to reach. The unknown variables are the angles that each joint in the kinematic chain needs to rotate in order to take the end-effector in reaching distance of those positions. For simplification purposes, the wrist of the virtual characters will be considered as the end-effector of the kinematic chain from now on.

The solution of the FK problem is straightforward assuming that the information about the kinematic chain is known from the beginning. This information is easily extracted from the type of joints and dimensions of the virtual characters inside the simulator. However, the IK problem can have an infinite number of solutions since there are an unlimited number of joint configurations and trajectories that the end-effector could follow to reach the desired coordinates.

This section describes the use of Deep Neural Networks and Gaussian Mixture Models (GMM) for solving the IK problem in the context of the current project. The model developed by [10] has the advantage of taking into consideration the hierarchical nature of anthropomorphic kinematic chains. Since small changes in rotation of those joints located further away from the end-effector in the kinematic chain create larger deviations, the model works sequentially by conditioning the movement of each joint until the one connected to the end-effector.

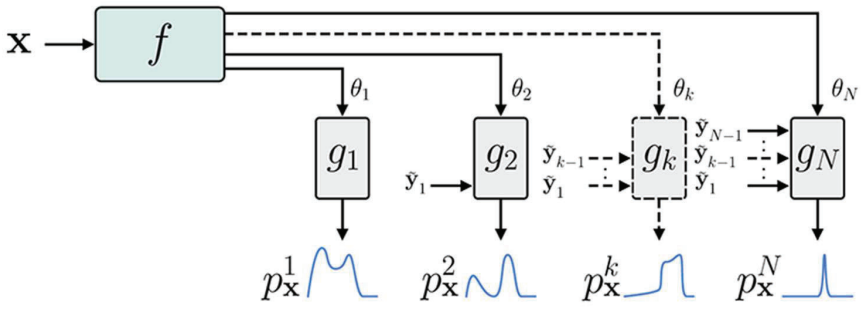


Figure 4: The Neural Inverse Kinematics model proposed by [9]. The hypernetwork f maps the desired position of the end-effector x (3D coordinates) to a set of parameters for the N primary networks representing each joint of the kinematic chain

The model is implemented as a hypernetwork [11] and can be divided into two main blocks: a hypernetwork f that maps the desired location in space for the end-effector; and a group of primary networks $g_1, g_2, \ldots, g_N$ representing GMMs for the joints in the kinematic chain, Figure 4.

Each sample in the training dataset is composed of two vectors $\{(x^{(i)}, y^{(i)})\}$ where $y^{(i)}$ represents the distribution of the joint angles that would take the end-effector of the kinematic chain to its target position $x^{(i)}$. The goal is to map the vector x representing the final position of the end-effector to a conditional probability distribution P_x such that the likelihood of the vector $y \in S_x$ is high, where S_x is a set of valid joint configurations for that specific end-effector position. The hierarchical structure of the kinematic chain is created by parametrizing the distribution P_x in sequential order:

$$\widetilde{y_1} \sim p_x^1 \coloneqq p_x(y_1)$$

$$\widetilde{y_2} \sim p_x^2 \coloneqq p_x(y_2 \mid \widetilde{y_1})$$

…

$$\widetilde{y_k} \sim p_x^k \coloneqq p_x(y_k \mid \widetilde{y_1}, \ldots, \widetilde{y_{k-1}})$$

… and

$$P_x \coloneqq [p_x^1, p_x^2, \ldots, p_x^N]$$

Here, p_x^k is the GMM distribution parameterized with a three-dimensional vector m_x^k capturing the mean, variance, and mixture coefficients:

$$[\theta_1, \theta_2, \ldots, \theta_N] = f(x)$$

$$m_x^1 = g_1(\theta_1)$$

$$\widetilde{y_1} \sim p_x^1$$

$$m_x^2 = g_2(\theta_2, \widetilde{y_1})$$

$$\widetilde{y_2} \sim p_x^2$$

…

$$m_x^N = g_N(\theta_N, \widetilde{y_1}, \widetilde{y_2}, \ldots, \widetilde{y_{N-1}})$$

$$\widetilde{y_N} \sim p_x^N$$

6.3 Datasets

The datasets are divided between the real data collected in vehicles and synthetic data created using Unity and Blender simulators.

6.3.1 Real dataset for object detection

The dataset contains scenarios with multiple test participants seated inside of the different vehicles. There are both scenarios with the vehicle driving around a parking lot, as well as with the test participants performing actions such as talking on the phone, eating, and drinking in a stationary vehicle. Objects are both held in hand and placed in random locations within the cabin. The camera is mounted in the roof of the vehicle (above RVM).

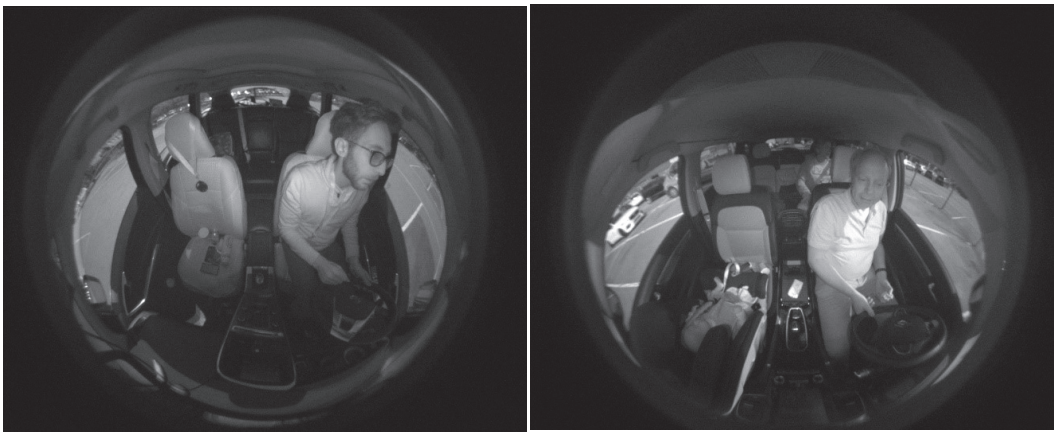


Figure 5: Real in cabin dataset

Table 2: Distribution of real dataset

Objects and people	Percentage of the class distribution	Total number per class
People	37.62%	12418
Mobile phones	27.61%	9115
Apples	15.44%	5098
Cups	19.32%	6378
Total	100%	33009
Classes per image in average (Total amount of bounding boxes / Total amount of images in the dataset)	33009/4150=7.95	

6.3.2 Synthetic dataset created for object detection

6.3.2.1 Dataset created in Unity

The synthetized datasets by Unity reproducing the real dataset collected are illustrated in Figure 6, with the data distribution described in Table 3.

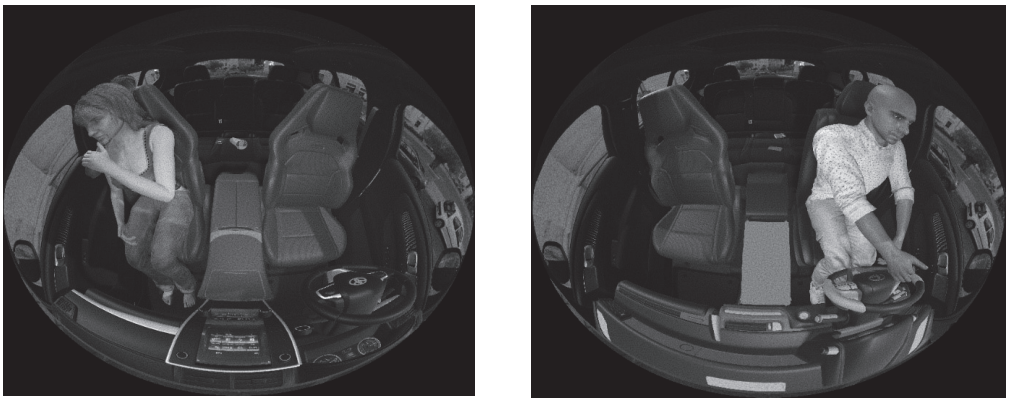


Figure 6: Synthetic images created using Unity

Table 3: Distribution of synthetic dataset generated by Unity

Objects and people	Percentage of the class distribution	Total number per class
People	33%	60474
Mobile phones	33%	59948
Apples	24%	44177
Cups	10%	18415
Total	100%	183014
Classes per image in average (Total amount of bounding boxes / Total amount of images in the dataset)	183014/30000=6.10	

6.3.2.2 Dataset created in Blender

The synthetized datasets by Unity reproducing the real dataset collected are illustrated in Figure 7, with the data distribution described in Table 4.

Figure 7: Synthetic images created using Blender

Table 4: Distribution of synthetic dataset generated by Blender

Objects and people	Percentage of the class distribution	Total number per class
People	37%	44899
Mobile phones	11%	14006
Apples	12%	14596
Cups	40%	49355
Total	100%	122856
Classes per image in average (Total amount of bounding boxes / Total amount of images in the dataset)	122856/30000=4.10	

6.3.3 Real dataset for action detection

The real dataset used for the action recognition in the project is the Drive & Act dataset [12]. It contains a total of 12 h videos of annotated actions captured in cars, with 15 people participating. The actions have been separated into smaller segments corresponding to one out of 83 classes.

6.3.3.1 Structure

Out of the 83 potential activities a subset of five was chosen, eating, drinking, working on laptop, talking on phone and interacting with phone. This subset results in a very small portion of the dataset being viable for this specific task. In total 118324 images were used and set up as sequences. The distribution of sequences is based on the evaluation run and is described in Table 8.

The videos were separated into frames using ffmpeg [13] and categorized into a hierarchical folder structure similar to the one in the synthetic dataset. The intention was to create the same format for both datasets to allow the same algorithms to run on either one. For the same reason the framerate of the real dataset was changed from 30 to 15 Hz by removing every other image.

6.3.3.2 Preprocessing

One difference between the synthesized data and the real data is the length of the sequence. In the case of synthetic data there is one long sequence per person and action, while in the real data they are divided up into more but smaller sections. This has a large impact on the final size of the dataset as the action recognition algorithm requires a set of starting images to act as a history. The result is that a smaller portion of every sequence is effectively not useable for training or evaluation purposes. In the case of synthetic data there is a very small impact as there are few sequences on the real data, however, the choice of how large the moving window size (number of frames) has a large impact on the real data with many smaller sequences. This is a limiting factor to how long history can be used in the processing for both the synthetic and real datasets as it is important to keep it the same between the two datasets for comparison and combination.

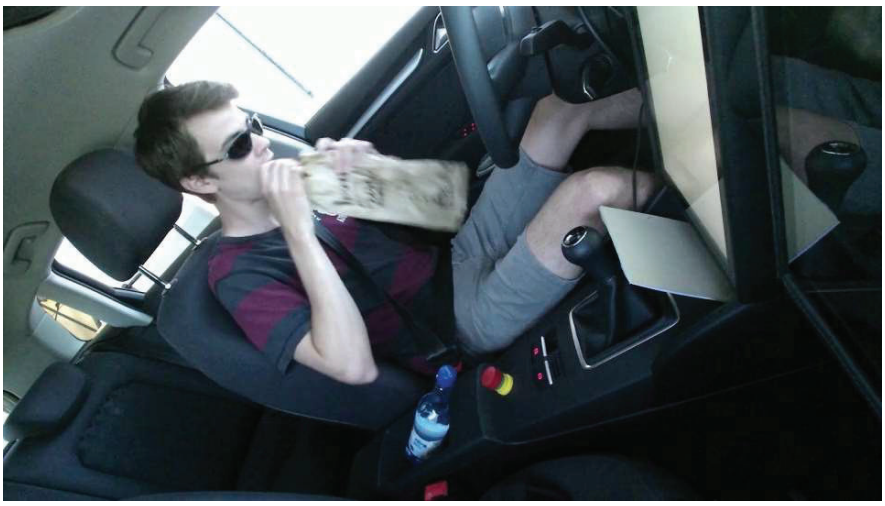


Figure 9: Example image of the eating class in the Drive & Act dataset [11]

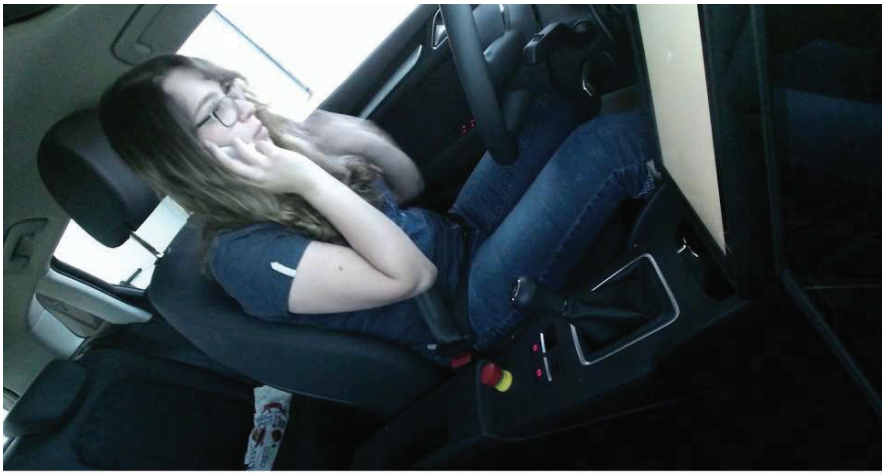


Figure 8: Example image of the talking on phone class in the Drive & Act dataset [11]

6.3.3.3 Balancing datasets

To avoid bias between the synthesized dataset and the real dataset they should contain the same number of images. With two equally sized datasets the evaluation can be performed by sampling a percentage of each dataset and combining them into a new dataset with the contributions from real/synthetic respectively: 100/0 %, 80/20 %, 60/40 %, 50/50 %, 40/60 %, 20/80 % and 0/100 %.

6.3.4 Synthetic dataset created for action detection

6.3.4.1 Realistic movement from Part 1

This dataset contains image sequences of people performing actions created using recordings of 6 real people eating an apple, drinking from a cup, and talking on their phone with left and right hands. The pose data from these recordings were then applied to 4 female and 4 male synthetic characters. In contrast to the original recordings the cup drinking sequence was rendered with the characters drinking from both a water bottle and a soda bottle instead. Similarly, the apple eating sequence was rendered with the characters eating an apple, as well as eating an apple

that was partially wrapped in a bag. Lastly, a fully synthetic action where the characters interact with a laptop on their knee was created without a reference recording. Some basic statistics for images can be seen in Table 5.

Table 5: Basic statistics for images.

Statistics per Unit	**Unit name**	**Number of images per unit**
Actions	Eating with the right hand	10464
	Eating with the left hand	10368
	Drinking with the left hand	9120
	Drinking with the right hand	8160
	Talking on the phone with the right hand	8784
	Talking on the phone with the left hand	6912
	Texting on phone with the right hand	4800
	Texting on phone with the left hand	4992
	Working on the laptop with the both hands	5232
	Total	68832
Synthetic characters	Male 1	8604
	Male 2	8604
	Male 3	8604
	Male 4	8604
	Female 1	8604
	Female 2	8604
	Female 3	8604
	Female 4	8604
	Total	68832
Real people	Male 1	11472
	Male 2	11472
	Male 3	11472
	Male 4	11472
	Female 1	11472
	Female 2	11472
	Total	68832

Figure 10: Image sequences synthetized with realistic movements from Part 1 (left: drinking sequence, right: talking on phone sequence)

6.3.4.2 *Realistic movement from Part 2*

In the study case presented in section 6.2.3.2.2 with a simplified kinematic chain, the value of N is 8. Training samples were originally planned to be extracted from MoCap systems for this part of the project; however, the GMM-based IK model needs a large training dataset, so it was decided to generate artificial samples instead. A total of 20000 samples were generated for training and 1000 extra samples were used for validation. All of these

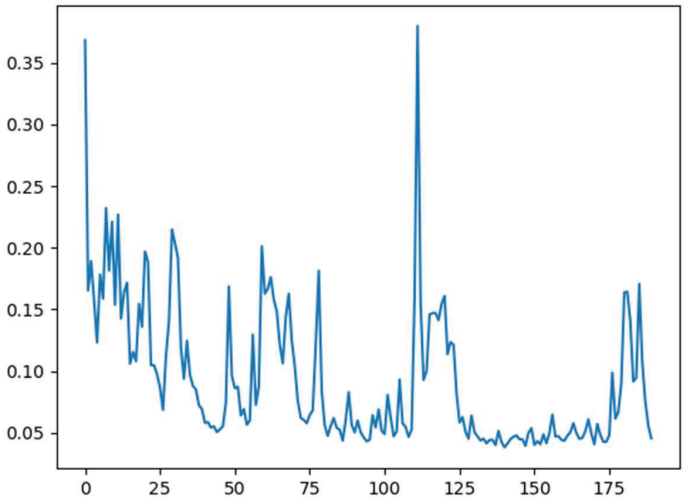


Figure 11: Testing results after 200 epochs. The best network configuration was observed at epoch #140

samples were generated randomly within the reaching space of the kinematic chain but constraining the rotation of each joint to be that of human-like limitations. The training was done for a total of 200 epochs, but the best network configuration was obtained at epoch 140 observing the best testing results, Figure 11.

The results of the training process can now be deployed in the simulator and the virtual characters developed and presented in the previous sections of this document. A simple test on the inference process for a random position in the space of a virtual driver can be seen in Figure 12. The left image contains 10 different inverse kinematic

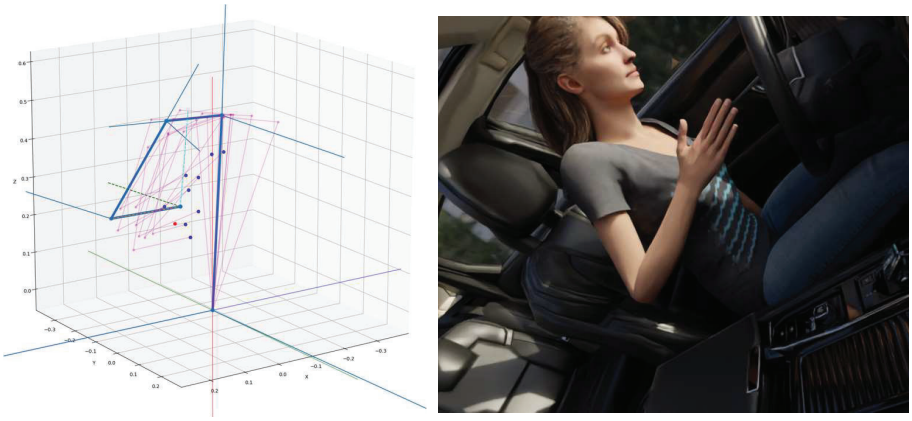


Figure 12: Left, an example of the solutions obtained from the GMM-based network for a random location in space (red dot). Right, the implementation of the best solution on a virtual character inside the simulator

solutions that approximate the position of the end-effector to the target marked as a red dot. The best solution is

selected as the one with the shortest distance from the end-effector to the target position, i.e., the kinematic chain marked in blue. All other possible solutions are depicted in magenta, and they show the advantage of this approach with respect to other methods since there are no “unnatural” body configurations which are usually found in analytic approaches. In this case, it is possible to see that all solutions are close to each other in the same skeletal configuration. The image on the right is the final output of the presented methodology; or using the suggested terminology, the front-end implementation of an inverse kinematic algorithm.

6.4 In-cabin action/object recognition algorithms

6.4.1 Object detection algorithm

The object detection model used in all experiments was YOLOv3[14], which is a single-stage convolutional neural network (CNN). The model that takes an image as input and outputs bounding box predictions for objects in three different scales – small, medium, and large. Each bounding box has a confidence score, a class label, a position as well as height and width. The network was trained from randomly initialized weights until convergence for all experiments.

6.4.2 Action detection algorithm

The action recognition network architecture used is based on the one developed in the DRAMA[1] project (Figure 1). However, within DRAMA2 project, several adaptations/improvements have been performed to meet the current project settings and to update the relevant components from the latest state-of-art developments in the field. The network architecture is decomposed into four interconnected components. Three of these are used extract action related features from the input images. These modules consist of object detection, optical flow, and body posture recognition. The fourth module is a temporal network model that use time sequences of concatenated features from the other modules as its input and provide the predictions as detected activities. This component is built as a combination of LSTM and dense layers. The module addresses the time dimension of the actions into consideration while the previous 3 modules are focusing on image-based features.

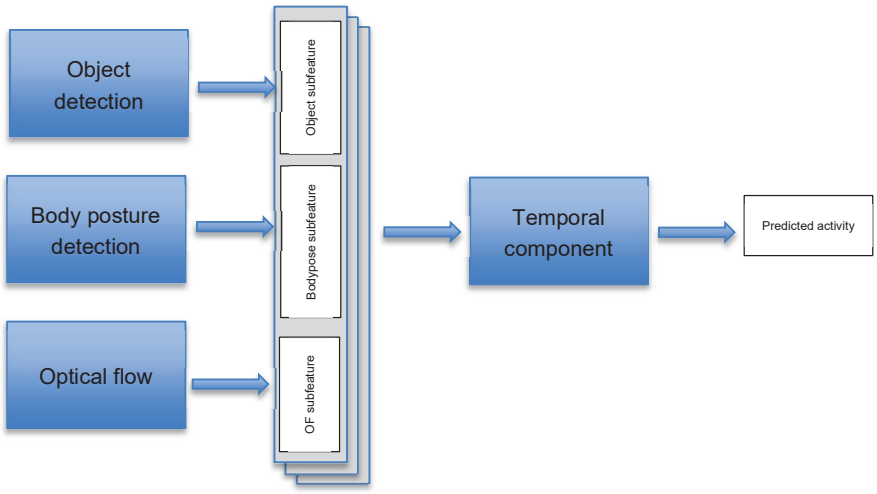


Figure 13: DRAMA activity recognition network architecture

6.4.2.1 *Object detection submodule*

The original module for object detection in DRAMA was a Yolov3 [14] (Figure 14). This network was, however,

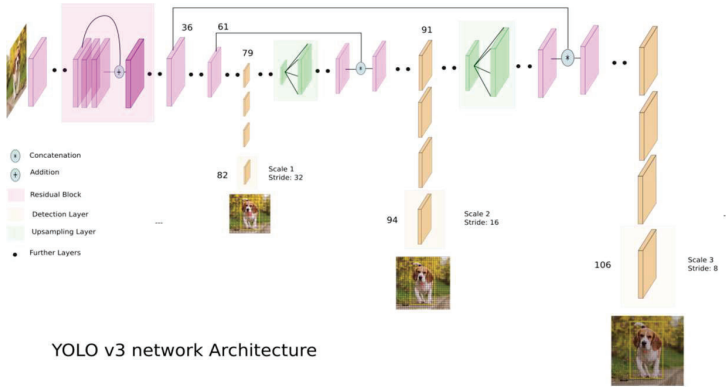


Figure 14: YOLOv3

delivered poor performance in recognizing the objects in either the real or synthesized datasets used within DRAMA-2. Therefore we selected a Swin transformer[15] (Figure 15) as the object recognition component. As illustrated in the top left of the figure, Swin transformer uses a hierarchical approach where patches of pixels are combined to represent higher hieratical levels. This allows for the usage of methods such as feature pyramids and U-nets. The intention with the network is to allow for the transformer architecture commonly used for natural language processing to be applied in image analysis. The output from the network is a mask with image

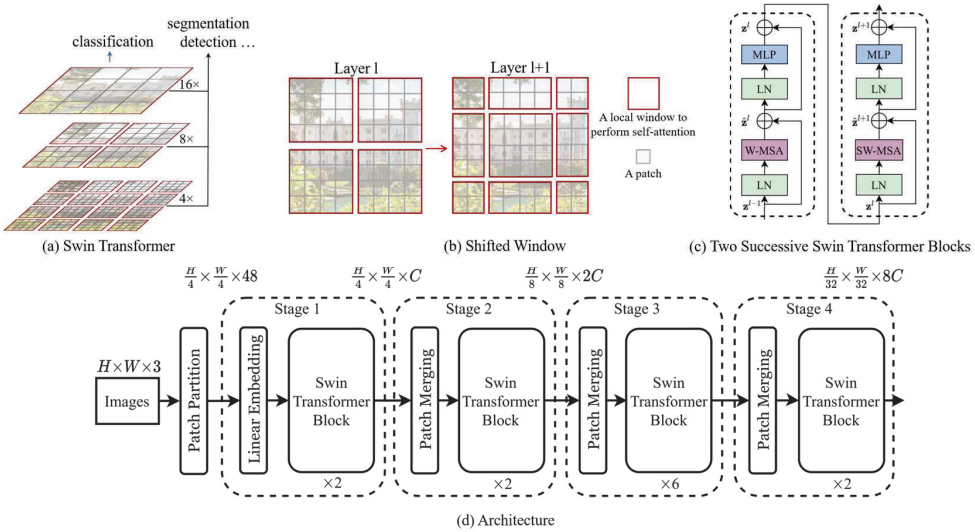


Figure 15: An illustration of the Swin transformer architecture and function [15].

segmentation and a list of bounding boxes with the corresponding classification score for the object in the box. In order to keep the number of features in the output from the object detection low, only the bounding boxes and detection scores of the first few objects have been used. This results in fixed length object feature vectors.

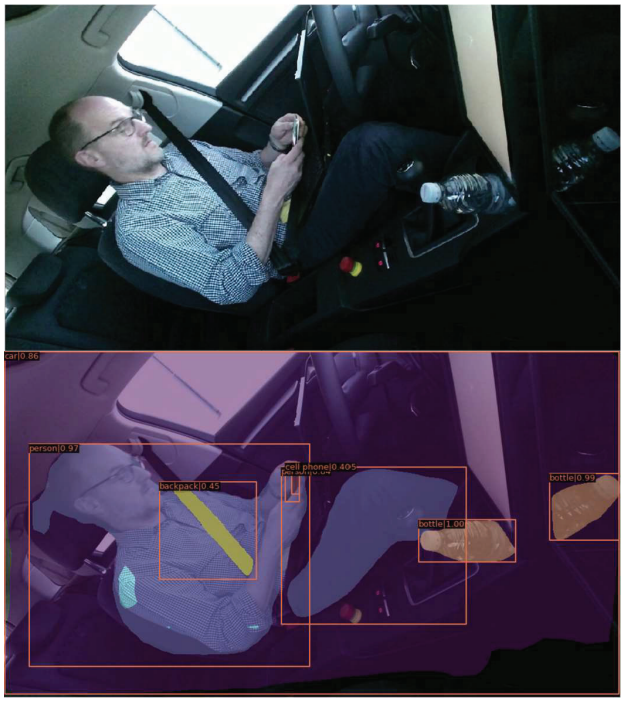


Figure 16: The upper image is the result of running the version of YOLOv3 used in DRAMA and the lower image is the results with the SWIN transformer utilized in the DRAMA II project.

6.4.2.2 Optical flow

The next feature extraction module is an instance of Farneback optical flow[16] to create a dense optical flow between every pair of consecutive images. This captures the pixel wise movement patterns. To reduce the feature dimension, a region-based histogram was used. In each region the vectors were divided into eight different bins based on the direction of the vector. The combination of all features from all regions are used as optical flow based feature vector.

6.4.2.3 Body posture

A very important set of information in order to recognize activities is the body posture of the person. To extract this information a machine learning based body posture recognition network has been used. In the previous DRAMA project, a network based on Posenet[17] was used for this. However, this network has poor performance when working with highly rotated images in the real and synthetic datasets used within this project. Therefore within DRAMA-2, this component is replaced with the latest SoA mmpose[18]. This selection resulted in significant improvement of the pose recognition performance and consequently higher quality of body pose subfeature.

6.4.2.4 Temporal component

To get a greater perspective of how actions are composed over time a temporal component is added to the network. This module, as in the previous DRAMA project, is based on combinations of dense and LSTM layers. After the feature vectors (extracted from the other three modules) have been concatenated into a larger combined feature vector per image, a limited length time series of concatenated feature is used as input to the temporal component. The length of the series is possible to change, but a limiting factor is the length of the series of images constituting an action in the dataset. Based on how long history is used a piece of the input series needs to be dedicated to

history and therefore reducing the total number of datapoints in the dataset. The choice of length of the history is thus limited by how many datapoints are necessary from the dataset.

6.5 Evaluation methodology

This section aims to describe the evaluation methodology used to produce the evaluation results.

6.5.1 Realism of the synthetic datasets

The goal of the synthetic datasets is to be interchangeable to real datasets. Thus, the aspect of realism is of importance in this project. To measure this realism, a metric called Fréchet inception distance (FID)[19] is used for evaluation with respect to this aspect.

FID measures the overall quality and diversity of a set of synthetic images. It is the most commonly used metric to evaluate images from generative models. The method is based on calculating the *Fréchet distance* (sometimes called *Wasserstein-2 distance*) between two image datasets. One of these datasets is the “real” or reference dataset, the other dataset is the “synthetic” or target dataset. Note that the distance is calculated on statistics obtained from the distributions of the datasets, such that no pairwise comparisons are needed. It is assumed that the goal is for the synthetic/target dataset to be similar to the “real” or reference dataset, if they are similar and the quality is good, then the FID score is small. It should also be noted that FID is shown to agree in most cases with human perception [19].

To calculate this distance, comparisons of images are needed. However, comparison in the pixel representation is not semantically meaningful. Thus, before comparisons are made, images are embedded using a pre-trained feature extraction network. The most commonly used feature extraction tool is the neural network Inception-v3[20]. More specifically, deep feature maps from the embedding network are used. These features from the images are then assumed to follow a multidimensional Gaussian distribution, where the mean and the covariance matrix are being used to calculate the actual distance.

FID is mathematically defined as:

$$FID = ||\mu_s - \mu_r||_2^2 + Tr\left(\Sigma_s + \Sigma_r - 2(\Sigma_s \Sigma_r)\right)^{1/2}$$

Where μs stands for the mean of the synthetic dataset and μr stands for the mean of the real dataset. Similarly, Σs, Σr are the covariance matrices of the two datasets. Finally, the Tr(*) stands for the trace function, the sum of the diagonal of a matrix. For this project, the torchmetrics implementation of FID is used[21].

6.5.2 Object detector evaluation

This chapter describes the methodology for evaluating the object detection model trained on synthetic and real data.

6.5.2.1 Mean Average Precision (mAP)

The performance of the object detection model was evaluated with the mean average precision (mAP) metric. It combines the *precision* and *recall* for all bounding box classes into a single metric for the model's overall performance. More specifically, assume a test dataset of images each containing a varying number of annotated ground truth bounding boxes. The ground truth bounding boxes each have a position, width, height, and a class label from one of *N* classes. The mAP is then computed in the following way: Each image is given as input to the model which outputs a set of predicted bounding boxes, each with the same properties as above as well as an additional confidence score indicating the reliability of the prediction. The predicted bounding boxes are filtered based on a confidence threshold θ, such that boxes with a confidence lower than θ are removed.

The predicted bounding boxes are then compared pairwise to the ground truth bounding boxes to find potential matches that can be counted as "positive" detections. If no match can be found, the predicted box is counted as a

"negative" detection. The matching is done using the intersection-over-union (IoU), which is a number between 0 and 1 for how well a predicted bounding box overlaps with a ground truth bounding box. If the IoU is above a certain threshold and the class labels are the same for a pair of predicted and ground truth boxes, it is counted as a positive detection and the ground truth is removed for consideration for all remaining predicted bounding boxes. In our experiments, an IoU threshold of 0.5 was used.

To compute the mAP score, the confidence threshold θ is varied from 0 to 1, resulting in varying amounts of positive and negative detections. For each θ, the positive and negative predictions are used to compute the corresponding precision and recall. These are given by the following formulas:

$$P = \frac{TP}{TP+FP} \qquad R = \frac{TP}{TP+}$$

where TP = True Positive, FP = False Positive, TN = True Negative and FN = False Negative.

The precision values for each recall value are averaged together to form the *average precision* (AP). This is done for each of the *N* class label separately, giving *N* AP scores for each class label. These are finally averaged to give the mean average precision, mAP.

6.5.2.2 Experiment scheme

For each of the two synthetic datasets, Unity and Blender, the model was trained in 25 experiments that investigated how the synthetic data is best combined with real data in order to increase the model's performance on a fixed test dataset, consisting of real images selected from 20% of the real dataset. The images were selected from different recordings compared to the training set, since frames from the same recording appearing in both the train and the test set would give the model an unfair advantage. The performance of the object detection model was evaluated using Mean Average Precision (mAP) in all experiments.

Short description of the experiment setups can be found in Table 6.

Table 6: Short description of the experiment setups

Experiment name	Training Dataset	Augmentation type
001-Real	Real	*NA
002-Real	Real	*Affine
003-Real	Real	*Full
004-Synth	Synthetic (~10k)	*NA
005-Synth	Synthetic (~10k)	*Affine
006-Synth	Synthetic (~10k)	*Full
007-Mix	Real - Synthetic (~10k)	*NA
008-Mix	Real - Synthetic (~10k)	*Affine
009-MIx	Real - Synthetic (~10k)	*Full
010-Mix	Real *ups - Synthetic (~10k)	*Full
011-Mix	Real *ups - Synthetic (~20k)	*Full

Experiment name	Training Dataset	Augmentation type
012-Mix	Real *ups - Synthetic (~30k)	*Full
013-Mix	Real - Synthetic (~10k)	*Best
014-Mix	Real - Synthetic (~20k)	*Full
015-Mix	Real - Synthetic (~30k)	*Full
016-Mix	10% Real *ups - Synthetic *ba	*Full
017-Mix	20% Real *ups - Synthetic *ba	*Full
018-Mix	50% Real *ups - Synthetic *ba	*Full
019-Mix	80% Real *ups - Synthetic *ba	*Full
020-Mix	90% Real *ups - Synthetic *ba	*Full
021-Mix	10% Real - Synthetic *ba	*Full
022-Mix	20% Real - Synthetic *ba	*Full
023-Mix	50% Real - Synthetic *ba	*Full
024-Mix	80% Real - Synthetic *ba	*Full
025-Mix	90% Real - Synthetic *ba	*Full

*NA=No Augmentation, *Affine=Shift, crop, scale, flip, *Full=Affine + blur, contrast, brightness aug.

*ups = with upsampling 50/50, *ba = best amount

6.5.2.2.1 Experiments 001-009

These experiments investigated the effect of applying image augmentations during training to address the domain gap problem. The rationale is that randomly augmenting the training data, both real and synthetic, reduces the model's tendency to overfit on features particular to either domain, thus improves its ability to generalize to new real data.

Three different sets of images augmentations were tried – none, affine only and full (affine + optical). Each of these were used in the following three sets of experiments: Experiments 001-Real, 002-Real, and 003-Real using real data only to train the model, experiments 004-Synth, 005-Synth and 006-Synth using synthetic data only and experiments 007-Mix, 008-Mix and 009-Mix using a mix of real and synthetic data.

6.5.2.2.2 Experiments 010-015

The purpose of these experiments was to investigate how the size of the synthetic dataset affects the model’s performance, while mixing it with the real dataset held to a fixed size. The model was once again evaluated on the same fixed real test dataset.

Three different sizes of the synthetic dataset were tried: 10 000, 20 000 and 30 000 images. Since the relative size of the synthetic dataset compared to the real dataset may be of importance, the experiments were performed both with and without upsampling of the real data such that it made up 50% of the batches given to the model during the training process, on average.

6.5.2.2.3 Experiments 016-025

Lastly, these experiments investigated the impact of the size of the real dataset, mixed with the synthetic dataset held to a fixed size. The size of the synthetic dataset was chosen based on the results of the previous round of experiments.

Five different sizes were tried for the real dataset: 10%; 20%, 50%, 80% and 90% of the total real dataset, with an original size of 4150 images altogether. Additionally, the experiments were once again performed without and with upsampling of the real data such that it made up 50% of the batches given to the model during the training process, on average.

6.5.3 Action detection algorithm

The metrics used to measure the performance of the action detection is accuracy, confusion matrix and average precision. The confusion matrices and average precision has been calculated using scikit learn[22], without taking imbalance in the class distribution into consideration.

6.5.3.1 Confusion matrix

A confusion matrix is used to show how the predictions relate to the true class labels. It is constructed by creating a matrix with the predicted classes in the columns and the true classes in the rows. The diagonal will in this case correspond to the TPs for all classes. The other elements represent misclassifications, this creates a visualization of how the misclassifications are distributed across classes.

6.5.3.2 Experiment scheme

To perform experiments highlighting the mixing of real and synthetic data for action recognition a process to remove and replace data was set up. First all the images in the real and synthetic datasets were set up as sequences of length 5. Secondly, starting with 100% real data a portion of the data was removed equal to 20%, 40%, 50% and so on following the percentages in Table 8. The choice of which sequences should be removed was made uniformly randomly across all sequences and classes of the real dataset. The same number of sequences that were removed was then replaced with uniformly randomly chosen sequences from the synthetic dataset to fill it up to be the same number of sequences as in the original real dataset. Therefore, all the tests were performed with the same number of sequences, but varying amounts from real and synthetic data.

6.6 Evaluation results

This chapter aims to present the evaluation results obtained during the project.

6.6.1 Dataset realism results

Table 7 contains the result from calculating FID on several target datasets. The reference dataset used for this experiment is Real dataset 1.1. Furthermore, FID is also calculated for a real dataset called Real dataset 1.2 this is done as a baseline, giving a rough idea on what FID values are reasonable. Note that Real dataset 1.2 contains real images. By inspecting the table, Real dataset 1.2 achieves the best FID score of 94.63, however, of the two synthetic datasets, Unity and Blender, the latter outperforms Unity with an FID score of 136.87 while Unity only achieves 162.50. Thus, the Blender dataset is a ~16% improvement over Unity w.r.t. FID. Furthermore, since the Real dataset 1.2 is realistic, its FID value can be used as a baseline to calculate the increase in FID for the synthetic datasets. This gives that the Unity dataset increases FID by ~72%, whereas the blender dataset only increases the FID by ~45%.

Table 7: Realism evaluation results of the synthetic datasets

Reference dataset	Target dataset	FID (Lower is better)	% change in FID compared with baseline
Real dataset 1.1	Synthetic dataset created in Unity with post-processing	162.50	+72%
Real dataset 1.1	Synthetic dataset created in Blender with post-processing	136.87	+45%
Real dataset 1.1	Real dataset 1.2	94.63	Baseline 0%

6.6.2 Object detection algorithm results

This chapter summarizes and discusses the results of the object detection experiments described in section 6.5.2.

6.6.2.1 Results when using only real dataset

Real data only was tried for three sets of augmentation in experiments 001 to 003. The loss curves and mAP score for 001-Real suggest that heavy overfitting occurs, likely due to the small size of the dataset. As affine image augmentations are added in 002-Real, overfitting decreases, and mAP increases significantly. This indicates, as expected, that the variation in image content due to the augmentations helps with generalization. Adding the full augmentations in 003-Real resulted in a slight decrease in performance. This is likely explained by the fact that the real training and test sets already are similar in distribution and hence adding optical augmentations makes the two sets less similar and hence leads to worse performance for the model.

6.6.2.2 Results when using only synthetic dataset

Synthetic data only was tried in experiments 004 to 006 with the three sets of image augmentations. For both the Unity and Blender synthetic datasets, the resulting mAP scores is very low in 004-Synth when no augmentations were used. When affine augmentations are added in 005-Synth, mAP increases for the Blender dataset but remains low for Unity. In 006-Synth, mAP increases for both datasets, again being highest for Blender where mAP is close to that of experiment 001-Real, i.e. real data without augmentation.

The increasing mAP as more image augmentations are used suggests that these helps decrease the domain gap between the synthetic training data and the real test data. The Blender dataset resulted in the highest mAP score for all experiments, indicating a higher level of realism for this dataset. This is also in-line with its qualitative appearance as judged by humans (in this case the researchers in the project). However, mAP scores for both synthetic datasets remain far below those of the real dataset with the same augmentations applied, suggesting that the synthetic data is not sufficiently realistic to replace real data.

6.6.2.3 Results when mixing the real and synthetic datasets

Mixing of real and synthetic data was tried in all three experiment groups, from experiment 007 to 025. Experiments 007 to 009 showed a similar pattern as the previous experiments, with mAP increasing in 008-Mix as affine image augmentations are added compared to using no augmentations in 007-Mix. This occurs for both synthetic datasets and can likely again be attributed to a reduction in overfitting. However, the full augmentations in 009-Mix did not significantly influence mAP or overfitting. Since full augmentations resulted in a decrease in mAP for 003-Real but a increase for 006-Synth, this could potentially be explained by these two effects cancelling out and resulting in a near-zero change in mAP for 009-Mix.

The results of experiments 010 to 012 show that upsampling the small real dataset leads to overfitting. In experiments 013 to 015 when no upsampling is used, less overfitting occurs. Performance also increases as more

synthetic data is added up to a point, after which it decreases again. A potential explanation can be illustrated by considering the limit values. The previous experiments showed that the synthetic data benefits performance when mixed with real data but is not sufficiently realistic to replace it. Having a synthetic dataset size of 0 is hence not optimal as it benefits generalization but having too big of a size is not optimal either as it will completely drown out the smaller real dataset and result in overfitting. Therefore, a maximum occurs for the given amount of synthetic data. 014-Mix with 20k synthetic images resulted in the highest mAP. Given the real dataset size of roughly 4k images, a ratio of 1:5 of real and synthetic data was optimal in this case. However, the exact ratio likely varies depending on the problem at hand.

Lastly, the results of experiments indicate, as expected, that mAP increases with the size of the real dataset as the size of the synthetic dataset is held fixed. However, it seems like there are diminishing returns as the real dataset grows in size. This trend is stronger for the Unity dataset, but present for both. Thus, mixing a small amount of real data with a large synthetic dataset seems to be a fruitful approach for achieving good performance and avoiding the overfitting that would otherwise occur.

6.6.2.4 Propose guidelines on how to combine generated data with real data to increase the robustness of in-cabin monitoring algorithms.

The experiments showed that synthetic data can lead to improvements in performance when mixed with real data during training, given that the synthetic data is realistic enough for the model to generalize to new real data. Given that experiments 004 to 006 showed that a model trained on our synthetic data gave valid predictions on the real test set, this indicates that the synthetic data does not just alleviate overfitting but indeed does contribute to better generalization.

However, the experiments also showed that the synthetic data was not realistic enough to completely replace the real data. Choosing an appropriate mixing proportion, i.e. the size ratio of the real and synthetic datasets, is therefore important. The exact number will depend on the realism of the synthetic data and the given task, but in our experiments a ratio of 1:5 of real to synthetic was found to be optimal.

Image augmentations were also shown to be a good method for reducing overfitting and reducing the domain gap between real and synthetic data and is therefore be recommended to be used.

6.6.3 Action detection algorithm results

The DRAMA network, improved as described in section 6.4.2, was used as a benchmark for action detection evaluation. To eliminate the effect of mixing real and synthetic training datasets mixtures on the performance of the object/body posture detection components (that have been discussed separately in section 6.5.2), the object detection and body posture detection components are using the same pretrained networks (with real data) in this series of evaluation experiments.

The DRAMA network has been trained on different selections of datasets for 3000 epochs and performance metrics were measured. The mixture of real and synthetic datasets was done by selecting random elements from the real dataset to match the percentage being replaced, and the same number of elements from the synthetic dataset were selected and combined, maintaining the total number of elements. Due to this random selection process, the elements used for training may vary between runs and even between different dataset mixes in the same evaluation, leading to variations in performance.

6.6.3.1 *Results when using only real data*

As a baseline the DRAMA action detection model was trained for 3000 epochs on the real action detection dataset[11]. The results are presented in forms of a confusion matrix in Figure 17. The distribution across the classes for the dataset used in training and testing can be found in Table 8. It is not surprising that the performance for the drinking class is the lowest, as it is significantly underrepresented compared to the largest class, eating, in the dataset.

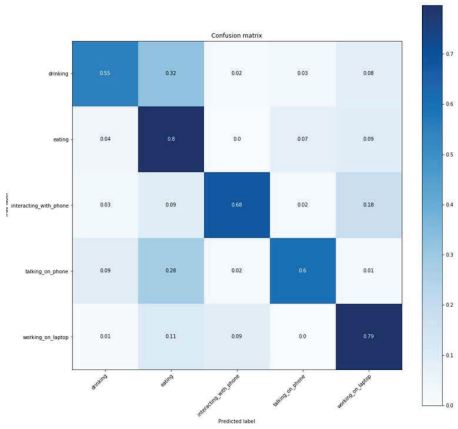


Figure 17: Confusion matrix for DRAMA network trained on 100% real data for 3000 epochs

6.6.3.2 *Results when using only synthetic data*

On the other side of the spectrum the performance of the model trained only on synthetic data can be observed in Figure 18. Most classifications trend towards the drinking class, with lower performance on the other classes being evident.

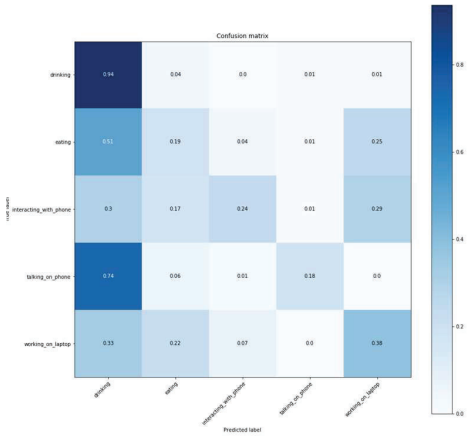


Figure 18: Confusion matrix, trained with synthetic data only, 3000 epochs

6.6.3.3 *Results when using mixture of real and synthetic data*

Similar to the training with 100% of the real or synthetic datasets, mixed datasets are trained for 3000 epochs and tested on 100% real data. The class distributions in mixed datasets can be found in Table 8. As illustrated as configuration matrices in Figure 20 to Figure 24, the results are similar with small variations. The drinking class still has low performance, and the focus is mainly on eating. However, the focus has shifted from working on a laptop to interacting with a phone, which is due to the relatively small number of examples in the synthetic dataset for working on a laptop.

The improvement is noticed at balanced mixes (40%/60% and 50%/50%) of datasets for interacting with phone.

Surprisingly, there is little improvement in the drinking class even though the number of labels increases with more synthetic data.

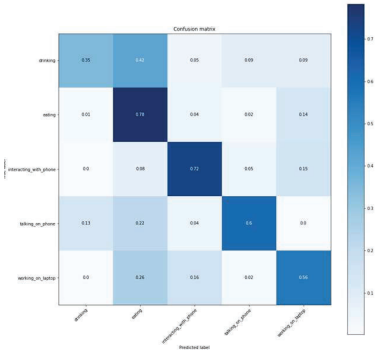


Figure 20: Confusion matrix, 20% real, 80% synthetic data

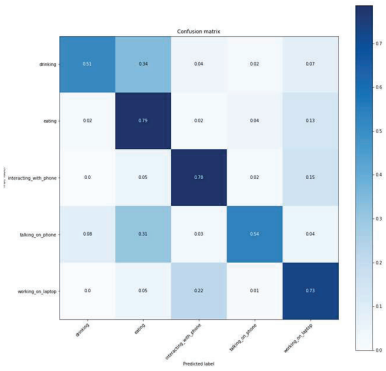


Figure 22: Confusion matrix, 60% real, 40% synthetic data

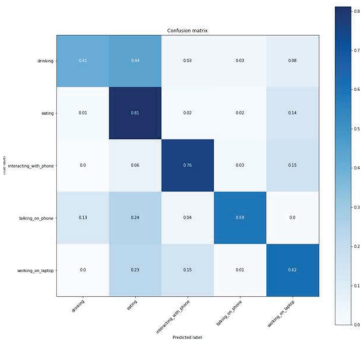


Figure 19: Confusion matrix, 40% real, 60% synthetic data

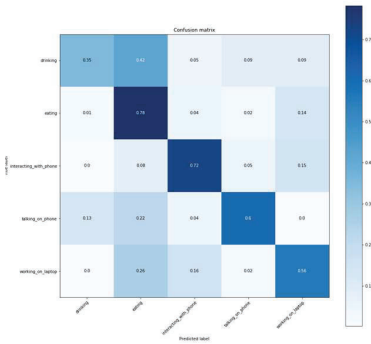


Figure 21: Confusion matrix, 20% real, 80% synthetic data

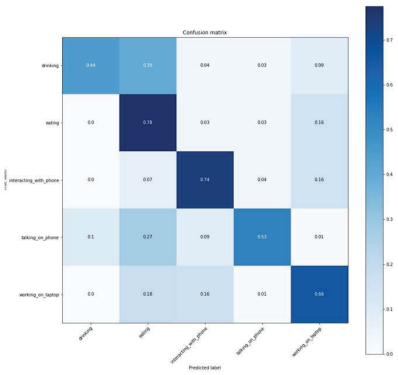


Figure 23: Confusion matrix, 50% real, 50% synthetic data

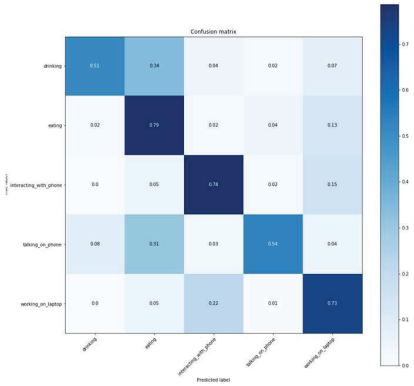


Figure 24: Confusion matrix, 80% real, 20% synthetic data

Table 8: Distribution of classes among sequences used for training and testing

Dataset mix	Drinking	Eating	Interacting with phone	Talking on phone	Working on laptop
100% Real	4762	28841	15227	11327	14271
80% Real	7301	27622	14286	12659	12559
60% Real	9668	26387	13402	14012	10958
50% Real	10978	25863	12971	14565	10051
40% Real	12203	25380	12505	15179	9160
20% Real	14647	24055	11627	16648	7450
0% Real	16432	21840	10192	17080	5488
Test	1068	7760	4003	3991	2236

The test accuracies of different dataset splits per training epoch is shown in Figure 25. Performance remains similar even with medium real data percentages. There is a slight variation between the different mixes without a clear pattern, e.g. the 50% mix performs worse than both the 60% and 40%. This could be due to the randomness in dataset sampling, leading to variations. Interestingly, the 60% real data mix has the best accuracy and could improve further with early stopping, as it declines midway. This suggests supplementing real dataset with synthetic data could lead to better or similar results with a smaller dataset. However, duplicates or similar images could replace real ones during random sampling. The fully synthetic dataset performs poorly, limited by an over-reliance on drinking classifications. It is not the largest class in the synthetic dataset, but larger than in the real dataset, resulting in poor performance because drinking is underrepresented. Figure 25 clearly shows the issue stems from an over-reliance on drinking.

Future work could improve the similarity between the synthetic and real datasets. Currently, the synthetic data is generated from captured real movements, but since the results are based on a test dataset from the real dataset, they lack its advantage. This is because the real dataset is used for both training and testing, assuming that the movements within it are similar, which is a reasonable assumption. However, this can disadvantage the synthetic

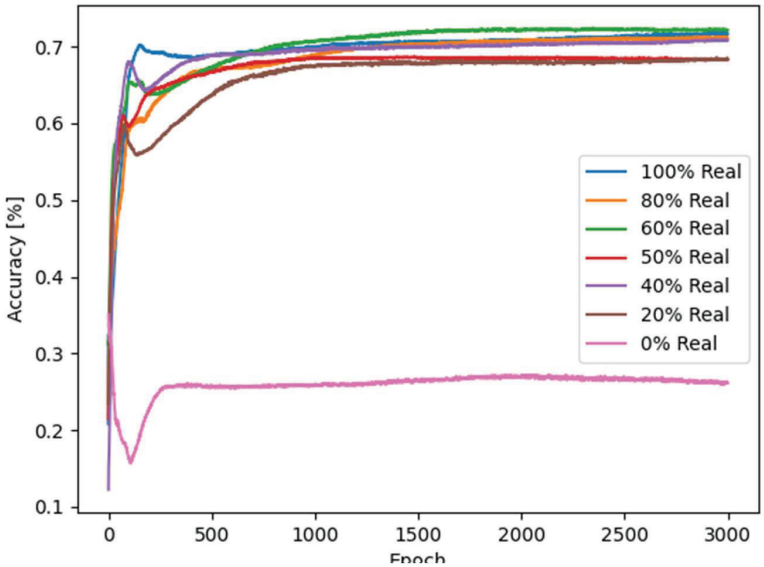


Figure 25: Accuracy on the test dataset for each epoch of training for each training data split.

dataset, as its movements, though realistic, may not match those in the real dataset. Ultimately, this is a question of generality, and the test dataset should be general enough. But this is rarely the case, which highlights the need for better data enrichment methods.

6.6.3.4 *Results when varying inputs and sequence lengths*

To isolate the effects of object detection on the results, the next tests were conducted without optical flow and only used body posture and object detection. The body posture data was normalized based on the bounding box around the detected skeleton points, and the leg skeleton points were omitted as they had low quality due to partial occlusion. A sequence length of 15 frames was set as the standard for these tests.

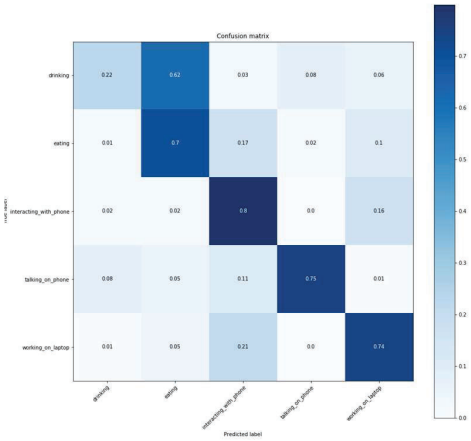


Figure 27:confusion matrix when trained on only the complete real dataset with a sequence length of 15 frames and no optical flow or object detection.

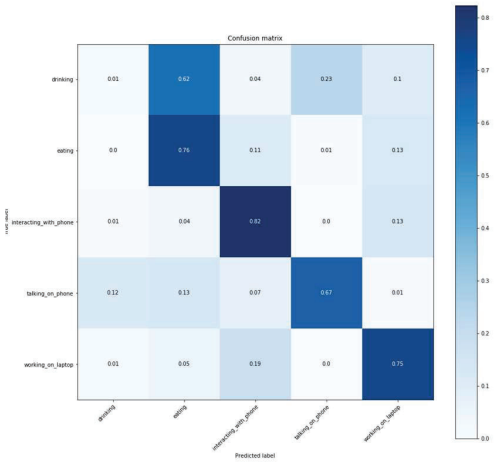


Figure 26:confusion matrix when trained on the complete real dataset and the complete synthetic dataset with a sequence length of 15 frames and no optical flow or object detection.

Figure 27 and Figure 26 show that simply combining both datasets is not effective. Surprisingly, the drinking class performs worse when the synthetic dataset is added, which has more examples of the drinking class than the real dataset. This discrepancy could be due to differences between the synthetic and real testing datasets for the drinking class, making the addition of more synthetic data for that class a disadvantage. To address this, a better merging method that considers the similarities between the datasets and their classes is needed. Additionally, the evaluation of the data is based on people from the real dataset, which may favor the real dataset over the synthetic data. Future work should examine the combination of datasets on other datasets to determine the benefits of adding more information.

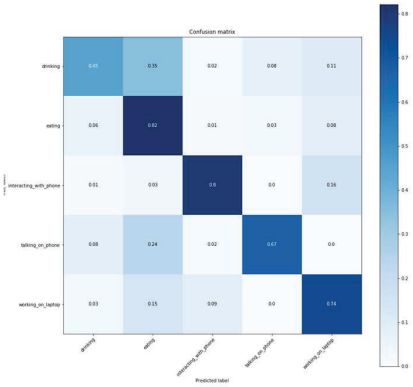


Figure 28:confusion matrix when trained on only the complete real dataset with a sequence length of 15 frames and no optical flow, but with object detection

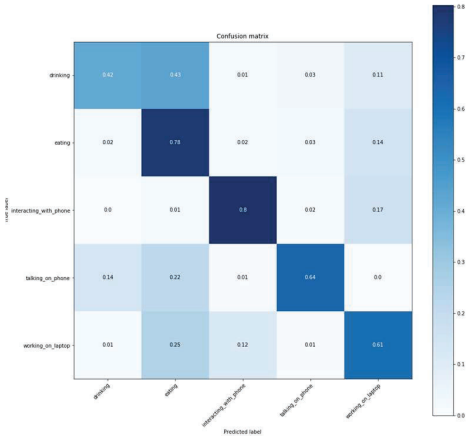


Figure 29:confusion matrix when trained on the complete real dataset and the complete synthetic dataset with a sequence length of 15 frames and no optical flow, but with object detection.

Figure 28 and Figure 29 show lower differences in the drinking class compared to Figure 27 and Figure 26. This may be due to the role of object detection in classifying this class and the closer similarity between synthetic and real data in this aspect. The results improve with the addition of object information, adding more data for the algorithm to utilize and creating a more balanced performance across classes.

Figure 31 and Figure 30 show results when sequence length is reduced from 15 to 5 frames while using object

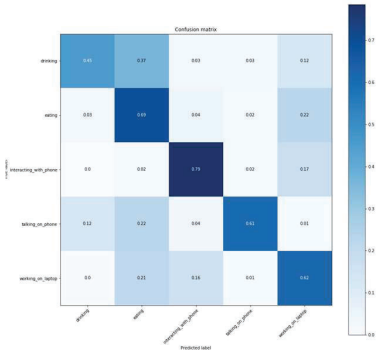


Figure 31: Confusion matrix when trained on both real data and synthetic data with a sequence length of 5 frames with object detection and without optical flow.

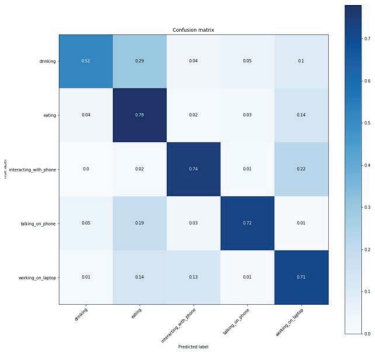


Figure 30: Confusion matrix when trained on only real data with a sequence length of 5 frames with object detection and without optical flow.

detection. Using longer sequences performs better with synthetic data compared to using only real data. This may be due to shorter sequences having more datapoints in training, leading to improved accuracy for real data, which makes synthetic data less competitive.

7 Dissemination and publications

7.1 Dissemination

Hur har/planeras projektresultatet att användas och spridas?	Markera med X	Kommentar
Öka kunskapen inom området	X	Synthetic data generation allows rapid generation of a diversified dataset. The project proves that synthetic data is a valuable contribution to real images in neural network training. Develop knowledge and knowhow on how to use simulator software to generate training datasets
Föras vidare till andra avancerade tekniska utvecklingsprojekt	X	The project result provides a tool for generating datasets that can be used in future research projects allowing early proof of concept without the investment of real data collection. A follow up project DIFFUSE is continuing investigation of synthetic data application in face recognition.
Föras vidare till produktutvecklingsprojekt	X	The project results provide a basis for the development of a production tool for synthetic data generation that will be part of the training pipeline at Smart Eye. This leads to cost and development time reduction as the need for real data collection is reduced.
Introduceras på marknaden	X	Future interior sensing features, targeted to be introduced in cars 2025+, will be developed including the synthetic data generated by the synthetic simulator first version of which was developed in this project.
Användas i utredningar/regelverk/ tillståndsärenden/ politiska beslut		Irrelevant

The approach proposed with this project is also partly adopted in several ongoing research projects (Swedish funding: DIFFUSE, 2DSR. European funding: ROADVIEW), where the developed competence is also reused via the participated RISE and Smart Eye researchers.

7.2 Publications

In the course of the project various internal meetings, workshops, discussions took place. For those meetings various documentation was created to spread the knowledge and document ideas for future development. Among such documentation are Internal technical reports, Internal seminar presentations, Workshop presentations, Synthetic tool manual, etc. In addition to those, OMAD article was prepared and distributed. Additionally, E-BOOK was created: Interior Sensing – The Next Frontier in Improving Road Safety and the Mobility Experience. More publications based on the results received in this project are planned in the future.

8 Conclusions and future research

Our interpretation of the results of all experiments taken together is that the level of realism for synthetic data is an important factor for making the network generalize well to real data. Both synthetic datasets also showed increases in performance when data augmentations mimicking real data characteristics were applied, such as variations in noise, blur and lighting, which increases the realism of the otherwise “perfect” synthetic data and thus further supports our hypothesis.

It should be stated, however, that the results cannot be guaranteed to be attributable to the difference in realism between the two synthetic datasets. They each use their own logic for randomizing the locations of objects and car interiors and use different and varying amounts of 3D models for each object. However, while not being exactly equal in this regard, both datasets contain a fairly large variety of objects and sufficient randomization that we do not expect this to be a likely explanation of the difference in performance.

In this project we considered, used and tested datasets created using Unity and Blender software. In the future one could look into other software used to create synthetic data, such as, for example, Unreal.

In the future, one could also consider other methods used to simulate camera properties, such as noise, lens blur, etc. and their influence on the FID metric results.

One could investigate how results would be affected if one uses bigger real datasets and bigger synthetic datasets.

The experiments with DRAMA activity recognition network show that using normalized body skeleton positions as a sub-feature make the generalization much more complex for the synthetic datasets to improve the performance.

The generated synthetic datasets have already been used to improve Smart Eye Cabin Monitoring System (CMS), however, the evaluations resulted in the following improvement ideas for future research:

- How to eliminate the impact of illumination, color and small texture differences between datasets by using invariant features (e.g. color invariant, illumination invariant,...) in the first part of detection network.
- How to encode body skeleton joints as a vector in Euclidean space: removing the constraints of e.g. length of body parts or rotation angles at joints by excluding these information from the encoded feature vector. An idea of this could be using 3D rotation groups (SO(3)) coefficients to represent a body posture.
- Investigate the use of synthetic datasets for driver readiness recognition.

9 Participating parties and contact persons

RISE Research Institutes of Sweden AB

Lindholmspiren 3A

417 56 Göteborg

Contact: Boris Duran

boris.duran@ri.se

Smart Eye Aktiebolag

Första Långgatan 28 vån 7,

413 27 Göteborg

Contact: Henrik Lind

Henrik.lind@smarteye.se

10 References

[1] “DRAMA - DRiver Activity MApping | Vinnova.” https://www.vinnova.se/en/p/drama---driver-activity-mapping/ (accessed Nov. 20, 2019).

[2] J. Stewart, “Self-Driving Cars Won’t Just Watch the Road. They’ll Watch You, Too,” *Wired*, Feb. 13, 2017. Accessed: Feb. 10, 2020. [Online]. Available: https://www.wired.com/2017/02/self-driving-cars-wont-just-watch-world-theyll-watch/

[3] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context,” *arXiv:1405.0312 [cs]*, Feb. 2015, Accessed: Jun. 25, 2020. [Online]. Available: http://arxiv.org/abs/1405.0312

[4] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

[5] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, “Playing for Data: Ground Truth from Computer Games,” in *Computer Vision – ECCV 2016*, Cham, 2016, pp. 102–118. doi: 10.1007/978-3-319-46475-6_7.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Commun. ACM*, vol. 60, no. 6, pp. 84–90, May 2017, doi: 10.1145/3065386.

[7] H. Idrees *et al.*, “The THUMOS Challenge on Action Recognition for Videos ‘in the Wild,’” *Computer Vision and Image Understanding*, vol. 155, pp. 1–23, Feb. 2017, doi: 10.1016/j.cviu.2016.10.018.

[8] H. Kuehne, H. Jhuang, E. Garrote, T. Poggio, and T. Serre, “HMDB: A Large Video Database for Human Motion Recognition,” p. 8.

[9] Feng Pan, Wei Wang, A. K. H. Tung, and Jiong Yang, “Finding Representative Set from Massive Data,” in *Fifth IEEE International Conference on Data Mining (ICDM’05)*, Houston, TX, USA, 2005, pp. 338–345. doi: 10.1109/ICDM.2005.69.

[10] R. Bensadoun, S. Gur, N. Blau, and L. Wolf, “Neural Inverse Kinematic,” in *Proceedings of the 39th International Conference on Machine Learning*, Jun. 2022, pp. 1787–1797. Accessed: Jan. 05, 2023. [Online]. Available: https://proceedings.mlr.press/v162/bensadoun22a.html

[11] D. Ha, A. M. Dai, and Q. V. Le, “HyperNetworks,” presented at the International Conference on Learning Representations, Jul. 2022. Accessed: Jan. 05, 2023. [Online]. Available: https://openreview.net/forum?id=rkpACe1lx

[12] “Drive&Act: A Multi-Modal Dataset for Fine-Grained Driver Behavior Recognition in Autonomous Vehicles | IEEE Conference Publication | IEEE Xplore.” https://ieeexplore.ieee.org/document/9009583 (accessed Jan. 05, 2023).

[13] S. Tomar, “Converting video formats with FFmpeg,” *Linux Journal*, vol. 2006, no. 146, p. 10, 2006.

[14] J. Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” *arXiv:1804.02767 [cs]*, Apr. 2018, Accessed: May 10, 2019. [Online]. Available: http://arxiv.org/abs/1804.02767

[15] Z. Liu *et al.*, “Swin Transformer: Hierarchical Vision Transformer using Shifted Windows,” *arXiv preprint arXiv:2103.14030*, 2021.

[16] G. Farnebäck, “Two-Frame Motion Estimation Based on Polynomial Expansion,” in *Image Analysis*, vol. 2749, J. Bigun and T. Gustavsson, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 363–370. doi: 10.1007/3-540-45103-X_50.

[17] R. Wightman, “A Python port of Google TensorFlow.js PoseNet (Real-time Human Pose Estimation): rwightman/posenet-python.” May 09, 2019. Accessed: May 10, 2019. [Online]. Available: https://github.com/rwightman/posenet-python

[18] Mmp. Contributors, “OpenMMLab Pose Estimation Toolbox and Benchmark.” 2020. [Online]. Available: https://github.com/open-mmlab/mmpose

[19] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, “GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium,” in *Advances in Neural Information Processing Systems*, 2017, vol. 30. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/8a1d694707eb0fefe65871369074926d-Paper.pdf

[20] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, “Rethinking the Inception Architecture for Computer Vision,” in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 2818–2826. doi: 10.1109/CVPR.2016.308.

[21] “Frechet Inception Distance (FID) — PyTorch-Metrics 0.11.0 documentation.” https://torchmetrics.readthedocs.io/en/stable/image/frechet_inception_distance.html (accessed Jan. 30, 2023).

[22] F. Pedregosa *et al.*, “Scikit-learn: Machine Learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.