Subway Station Hazard Detection

Pascal Fischer, Felix Hoffmann, Edis Kurtanovic, Martin Ludwig, Alen Smajic

September 17, 2020

Abstract

Recently there was a nationwide scandal about an incident at Frankfurt Central Station in which a boy was pushed onto the railroad tracks in front of an arriving train and lost his life due to its injuries. However, this incident is not the only one of its kind because such accidents occur from time to time at train stations. At the moment there are no security systems to prevent such accidents, and cameras only exist to provide evidence or clarification after an incident has already happened. In this project, we developed a first prototype of a security system which uses the surveillance cameras at subway stations in combination with the latest deep learning computer vision methods and integrates them into an end-to-end system, which can recognize dangerous situations and initiate measures to prevent further consequences. Furthermore, we present a 3D subway station simulation, developed in Unity3D, which generates entire train station environments and scenes using an advanced algorithm trough a real data-based distribution of persons. This simulation is then used to generate training data for the deep learning model. Finally, the model is tested and evaluated on real camera recordings from a Cologne underground station.

Contents

1 User View

The User View consists of three essential parts, namely the context, task and performance. The context gives a first description of the overall problem we want to solve with our system. The task describes explicitly which domains are to be solved and defines the goal of the project. The performance describes the constraints which are to be complied by the implementation of the system.

1.1 Context

Train stations in general are very dynamic and lively places. Each day, many thousands of passengers can enter and leave a train station to get to their desired destination. There are over 5000 unique train stations in Germany. While most of these are on the surface, a non negligible number of underground stations exists. This is especially the case for big cities like Berlin or Frankfurt am Main. And while most passengers there can follow their daily routines perfectly fine and not be interrupted whatsoever, there is a number of incidental events which do happen on train stations from time to time. Every person who regularly used or uses the train in Germany will have at least witnessed some of these events. The events we mean are other people's behaviors which have negative effects on the passengers that simply want to travel from point A to B. Every once in a while passengers will meet a noticeable drunk person, screaming, shouting and sometimes even throwing things around. On other occasions they will have to deal with mentally ill people that can act unpredictable. Or, depending on the location, spotting a person on drugs might or might not be that uncommon. But aside from these, regular people can also misbehave and risk both their own well-being and the well-being of others. Recordings exist that show seemingly perfectly fine people - sometimes teenagers, people that want to reach the opposite platform or people that simply dropped an object on the tracks - mindlessly enter the train tracks - something that is forbidden by German law. By doing so, they create a higher risk of causing an accident.
Recently, a tragic incident at the main central station of Frankfurt am Main made it to the breaking news. In July 2019, a mentally ill man pushed a mother and her son onto the track while a train was arriving, as seen in a newspaper extract in Figure 1. Unfortunately, the 8 year-old child did not survive the incident. Ever since that tragic event, many people started demanding more security procedures to prevent such incidents. Statistically speaking, Germany is placed 2nd in the ranking of countries with highest amount of deaths by train accidents. Each year, about 150 people lose their lifes due to accidents involving tracks and trains. The number of train accidents without loss of a life and the estimated number of unknown cases is even higher. It is expected, that many accidents can be prevented by reacting early. Covering over 35.000 kilometers of track length, the current train infrastructure does not offer the possibility to physically block people's attempts of entering the train tracks everywhere. So far only very few modern key stations in Germany are equipped with walls and doors made out of transparent plastic (as shown in Figure 2) on their platform edge which can hinder people from entering the tracks, be it by accident or by intention. But while they do provide added value, the Deutsche Bahn and smaller (regional) transport companies (e.g. the VGF in Frankfurt am Main) decline this specific solution approach due to its high costs. During the last few years, both underground and surface stations have been augmented with more and better cameras for surveillance. However, these cameras are usually not used to detect dangers or potential dangers. Instead, they record everything that is happening on the station in case footage will be needed as clarification or as evidence.

1.2 Task

Given the already wide existence of cameras in underground train stations, the task can be defined as the following: The already present cameras are to be used as part of a universal applicable camera hazard detection system which detects potential dangers and present dangers. A potential danger

exists when a person walks too close to the edge of the platform. However, if there is a person on the tracks itself then there definitely is danger present already. Once any danger is detected, appropriate steps shall be taken or initiated to prevent accidents. Furthermore, the system shall be able to tell the rough location of where the danger comes from.

Frankfurter Hauptbahnhof

Mutter und Kind vor ICE gestoßen - Achtjähriger stirbt

Aktualisiert am 30.07.19 um 11:18 Uhr

Polizeiabsperrung am Frankfurter Hauptbahnhof *Bild © Michael Seeboth (hr)*

Ein achtjähriger Junge ist im Frankfurter Hauptbahnhof vor einen einfahrenden ICE gestoßen und getötet worden. Seine Mutter konnte sich noch rechtzeitig aus dem Gleisbett retten. Passanten überwältigten den Tatverdächtigen.

Figure 1: Newspaper (Hessenschau) article regarding the events on the central station of Frankfurt am Main. Link: `https://www.hessenschau.de/panorama/mutter-und-kind-am-frankfurter-hauptbahnhof-vor-ice-gestossen---achtjaehriger-stirbt,hauptbahnhof-notfall-100.html`

Figure 2: Glass doors and walls on the edge of a train platform in Barcelona (image source: derstandard.at)

1.3 Performance

As for the performance requirements, the system has to feature real-time end-to-end detection of the dangers mentioned above. Therefore, the recorded data has to be transmitted to the servers immediately so that the data can be processed as fast as possible and countermeasures can be initiated instantly, if needed. Once a dangerous situation exists, it has to be detected in real-time. Another important property of the system shall be that it detects as many dangers as achievable, while also keeping the amount of false alarms very low.

2 Modeler View

2.1 Overview

Our hazard detection system operates in a complex world where many different variables need to be considered in order to warn an incoming train of potential harm. The goal of the subway station hazard detection system is to increase the safety of all humans involved in travelling by train by spotting potentially dangerous situations in real-time and sending status updates to the next incoming train so that appropriate measures, such as slowing down the train, can be taken in time. In this project we assume that the train will be driving underground most of the time but there might also be situations in which the trails can lead overground for some distance but all train stations are assumed to be underground. We use the cameras at the train stations to take pictures at regular intervals which will then be processed so that we can determine a danger level. The danger level describes the potential for human harm or harm to the incoming train. In future iterations of our project we want to be able to consider several relevant variables e.g. current position and speed of the incoming train (determines braking distance), objects or animals spotted on the rails, etc. In our current implementation the positioning of the people inside the train station relative to the safety-line is the only factor that influences the danger level which then is transformed into one of the following three actions no action, slow down, do emergency braking. This action signal then gets transmitted to the train in real-time so that a suitable action can follow from the train conductor in charge. In the scope of this project we assume that there is a human conductor present in the train that is able to overrule any recommendation sent by our detection system.

2.2 Sensors, Train Stations and Deep Learning approach

The underground train stations that the train passes through are equipped with multiple cameras which send recommendations based on analyzed pictures to the train. The images can be seen as samples of the world's current state and will be used to determine a danger level which is a number on a defined scale which determines the train's next action. This enables the train to adjust its speed accordingly when approaching a train station: If the train station is empty (no humans / unusual objects spotted by the cameras at the platform) and there is no scheduled stop, then the current danger level is low and the train can pass through at a high speed. When people are spotted at the platform, then the train will get this information from the cameras at the train station beforehand. The train then can slow down to an adequate speed before reaching the train station in order to limit the risk of accidents involving humans.

The images taken from the cameras at the train stations will be analyzed by a Deep Learning model which is able to detect people in images so that we can use the location of these people as a factor in the calculation of a danger level. Our model has been trained on simulations but models like these also already exist for real-world image. An example for such a model would be "People Detection and Articulated Pose Estimation" from Andriluka et. al which is a new approach for detecting people and their poses in complex scenes (Sources [1] and[2]). An advantage of using such a model would be that it is able to handle partial occlusions of people.

If we can automatically analyze taken images from the cameras inside the train stations this will give us the information if there are people inside the train station, where they are, which pose they currently have and an in the future we might be able to estimate where they might be moving. This information then can be used as input for our danger level estimation procedure which takes into account various factors: People that are running, falling or lying on the ground will be rated as potentially dangerous. Our approach also considers the amount of people that are currently at the train station. The more people there are the higher the potential threat of an accident occurring is.

Another important factor is how close people are standing to the rails. The closer someone stands to the incoming train, the higher is the danger of the person falling on the rails accidentally or by being pushed there by someone else. You could extend the existing Deep Learning model to not only determine the position of people but to also differentiate between children and older people. This would be useful for our application because children pose a higher threat since they are more likely to behave in an unexpected or unsafe way, their rapid movements are usually harder to predict than those of older people. Animals also pose a potential threat since it's hard to predict their behavior and when a pet falls on the rails it is likely that a human will follow and put himself/herself in danger to save the pet. Another useful extension for the Deep Learning model would be to also recognize objects on the rails e.g. smartphones. A smartphone laying on the rails can lead to dangerous situations where a human enters the rails to pick it up even though there might be a train incoming soon. The idea is to take all of these factors into account to calculate the current danger level of the situation in the analyzed picture. The shorter the interval in which we analyze these pictures the earlier the train conductor can receive the signal of the current danger level which will lead either no action, to a slow down or in extreme cases to an immediate emergency braking. The concept of a danger level is picked by us in order to build a model that minimizes potential human harm which is a central ethical principle for AI proposed by the European Commission (Source [3]).

The scope of this project is limited to underground train station because overground train stations bring many changed world conditions which influence the performance of our system. The image quality of the overground footage provided by the camera systems is influenced not only by the weather conditions but also by the lighting conditions. Dark lighting leads to higher noise levels in the images taken which lowers the certainty with which objects and people can be identified in the images. The weather conditions can also affect the image quality in a negative way e.g. strong rain or snowstorm. The vision systems also need to be able to spot large objects on the rails so that there can be a signal to the train if it needs to lower its speed or perform an emergency braking. The cameras used in the train stations are high quality 1080p cameras (1920x1080 pixel resolution), as described in an official Deutsche Bahn document (Source [4]). You can see three example images that show these cameras in a real-world application in Germany below (Figure 3).

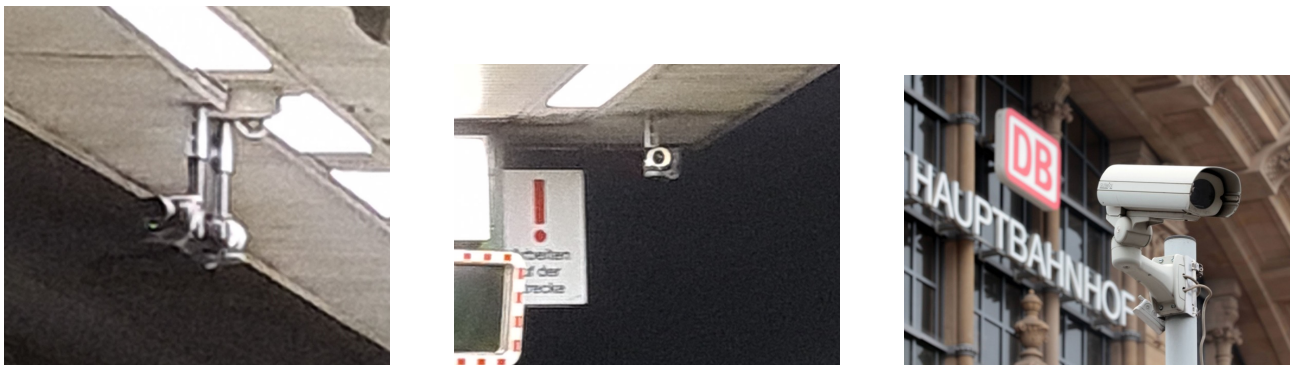

Figure 3: Example images of German train station cameras.
Source: [5]

Another aspect of the overground world that would need to be considered is the fact that weather conditions not only influence the quality of the images taken but also change the driving conditions of the train: For instance, if the rails are covered in ice then the train's braking distance will be longer than usual and we would need to be able to adjust the calculations of the braking distance that our train would have in real-time. In the underground world there won't be changing weather conditions but there might be fluctuations of the overall temperature which might influences the driving conditions only slightly. The underground world is illuminated by artificial light which constantly generates roughly the same degree of brightness in the underground stations where the cameras are placed. This means that the underground world overall has less uncertainty of conditions than the overground world in our model which means that the predictions we make for the underground world are potentially more reliable than the predictions that would be made in overground scenarios.

2.3 Train Station Structure

There are many different design choices that are implemented in real-life train stations which can affect not only the position of the fixed objects at the train station but also can influence the performance of our danger detection model. For instance, there are

- i) train stations that have rails on one side and there is a wall on the other side
- ii) train stations that have rails on both the left and the right side

In case i) there will be only one safety-line on the side of the station where the rails are located. This means that people located on the other side of the station where a wall is pose no threat and do not increase the danger level. From real world images we have learned a pattern that helps determining where fixed objects are located on these kind of stations. If there are rails on the left side and a wall on the right side of the station then most of the objects inside the stations are either close to the wall or attached to the wall.

In case ii) on the other hand there will be two safety-lines. This means that both people located too close to the left side rails and also people located too close to the right side rails will have impact on the current danger level. The object location pattern for these kind of rails are that most objects are located in the middle of the station. There usually are no fixed objects close to either side rails.

Example images can be seen below (Figure 4).

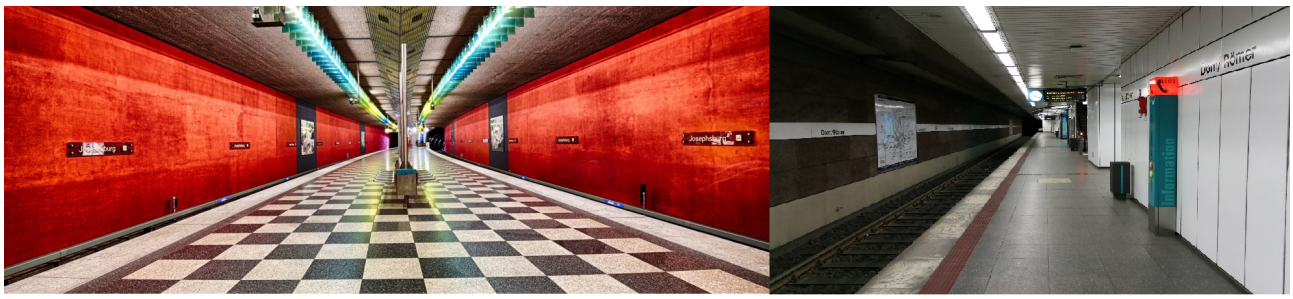

Figure 4: Example images of different train station layouts.
Source: [6] [7]

2.4 Fixed objects inside train stations

Our system needs to be able to distinguish between fixed objects that are always located at the same position inside the train station and anomalies which are objects that temporarily are spotted. After manually analyzing various pictures of German train stations we have compiled a list of the most common objects found inside a German train station:

- Ticket machine
- Garbage can
- Bench
- Railway schedule / Maps
- Destination board / other screens
- Vending machines
- First aid box
- Elevator
- Staircases
- Cameras

It should be mentioned that not all of these objects will be found inside every train station but there is a common pattern that train stations in Germany display at least a subset of these objects. The exact location of these objects varies from train station to train station even though there is a high likelihood of these objects being located as far as possible from nearby rails.

There are other aspects that differentiate train stations, for example the materials being used to build the station, the color of the walls, the material and color of the ground, ... These variables need to be considered when designing a vision system that needs to be able to be robust enough to not break when for instance a glass wall is able to reflect other objects which might lead to misdetection of certain objects or people. Another aspect here is an LCD screen that is often found at the wall next to the rails which shows travelling information or breaking news to the people waiting inside the station. Future implementations of our system would need to be able to differentiate between people shown on screen and people that are actually physically present inside the train station.

There are two kinds of cameras commonly used in train stations. The first camera model is a standard Full-HD camera, the second commonly used model is a 360 degree camera. These cameras usually are located at the ceiling of the train station tilted to an angle of about 30 degree to the ground. The lighting conditions in underground stations are constant (LED or tube light at the ceiling of the station) but the degree of brightness varies from station to station. Since the degree of brightness influences the noise of the pictures taken by the cameras there needs to be consideration of a camera noise model such as described in the paper of the European Machine Vision Association (Source: [8]).

For simplicity, our proof-of-concept model assumes constant lighting from all sides, no shadows and no reflections. We also assume that there are only standard Full-HD cameras inside the train stations, no 360 degree cameras in order to keep the complexity of our project at a reasonable level. These aspects can be updated in future iterations of our model in order to achieve a more accurate approximation of the real world.

2.5 Temporary objects inside train stations

The danger level is determined not by the fixed objects inside a train station but instead the the temporary objects that change over time and which can be hard to predict. We have compiled a list of potential temporary objects which can be found regularly inside train stations which might affect the current danger level.

The first item on that is list are humans which can have different sizes, have a different age (potentially can be used to assess their predictability) and wear different clothes. All humans share a similar shape which can be used to detect them. There are cases in which the face of the humans are not visible (e.g. not turned to camera or wearing a cap) and it is important that a vision system is able to spot these humans in these cases too. Another aspect that needs to be considered is masking, for instance there might be situations in which an object completely blocks the view on a person standing behind it. Cases like these might lead to a falsely assessed danger level which is why it is important to ensure that the cameras inside the train station are located in such a way that reduces the amount of blind spots. In the future there might be an intelligent system that understands for example that a backpack can't float through the station without any person carrying it and can use this understanding of the world in order to make correct inferences.

Humans often bring objects with them. For instance wheelchairs, walking aids, backpacks, suitcases, smartphones, ... are commonly found inside train stations. These objects can potentially be detected and used in order to assess a more accurate danger level. In the future there might be a system which can spot a suitcase on the rails, understand that this suitcase belongs to a human and infer that there is a high probability that some human will enter the rails to recover the object and therefore increase the current danger level.
Another aspect that needs to be considered is how our model handles clothes with images on them. There might be a scenario where someone wears a T-shirt that shows multiple people so our system might think there are multiple people on one spot and this might affect the danger level. This is something that we will need to consider on future implementations of our system.

Animals that vary in species and size might be found inside an underground train station. Most commonly found in German underground train stations are rats and birds, both of which do not directly influence the current danger level. In future systems there might be a module that is able to automatically detect animals, classify them and determine if they should play a factor in the calculation of the current danger level.

2.6 Danger Level

Using the cameras located at the train stations, we use a deep learning model to be able to determine how many people are located at the station and where exactly these people are currently located. Furthermore, using the model we are able to automatically get an age approximation so that we can determine if a person is a child, grown person or old person. We use this information to determine the current danger level at the train station, which can be interpreted as a level of uncertainty. A high danger level means that is a high probability of an accident happening soon, in this case the next incoming train will get notified so that it can slow down before approaching the train station. This allows for emergency braking in case there will be a dangerous situation e.g. a person spotted on the rails. Most train stations in Germany are equipped with some form of a safety-distance line on the floor which is located about 1.5m away from the edge of the station platform. When implementing our system in a real world application it would be important to make sure that there is some form of safety-distance line located 1.5m away from the edge of the platform that can be recognized by the cameras inside the train station. The reason for this is that we measure the distance of people to

the safety-distance line in order to evaluate if their positioning affects the current danger level. The safety-distance lines allows us to classify dangers into three different levels:

- No danger
- Potential danger
- Detected danger

It is also important for the people at the train station to be informed about where they are supposed to be waiting for the train without endangering themselves. Optionally, if a train station is equipped with a public address system there can be an automated warning announcement in case a sufficient danger level is reached. This gives instant feedback to the people involved inside the train station and is intended to help lower the average danger level.

In order to increase the explainability of our model we use a hybrid approach: the deep learning model analyzes the cameras pictures in regular intervals and then a rule-based system is applied to the output of the neural network in order to calculate the current danger level. In the table at the bottom of this page you can see the factors that this rule-based system considers when determining a danger level. All factors that contribute to a higher danger level are listed with a predefined weight, the higher the weight the higher the potential danger level. This approach allows us to put all these factors into a formula which then calculate a floating point number which stands for the current danger level. The advantage of this approach that at any point in time you can take an image with the cameras at the train station, feed this image to the neural network in order to get the information if there are people, what kind of people there are and where these people are currently located inside the train station. Then you can apply this rule-based system to this information to get a reasoned danger level. This is no black box approach, we have chosen this method in order to have an explainable and interpretable way to determine danger levels.

We use the current position of the incoming train as a factor that influences the current danger level. The reasoning for this is that a person being too close to the rails should not lead an emergency brake of the train if e.g. the train still is 20 min away from the train station. This means at all times we always need to not only analyze the current situation inside the train station but also consider the current position of the train in order to determine a useful danger level.

Our deep learning model is able to differentiate between fixed station objects and temporary station objects and uses the knowledge in combination with other detected information to decide if the danger level is no danger, potential danger or detected danger. This danger level then will be transmitted to the train conductor in real-time so that this person can make an informed decision.

2.7 Safety Protocol

In this subsection we will discuss the process that happens after our system created its output.
First of all, it shall be noted that there can be used up to two cameras for the same platform, one on each end of it. The way we vision this system is that it will be able to run for each for each individual camera independently. For each camera input individually the system then performs a segmentation. The people will be categorized in green (no danger), yellow (person too close to tracks) and red (person on track). Of course the two cameras might not have the same assessment of the same scene, because they are applied from two different perspectives and a platform can be longer than 100 meters, so one of the two cameras might not be able to accurately recognize the situation on the other far end.
As a result of this, after each cameras feeds their image input into the system, the system's results might look for each of the two camera images, i.e. there are no red people according to camera 1

while according to camera 2 there are definitely (which means the amount of pixel surpasses the threshold discussed in chapter 5) red people on the scene. If such things happen, then the worst case has to be taken. In this particular example where camera 1 reports no dangers while camera 2 does, according to this security protocol, we would choose to listen to camera 2's judgment and prepare the necessary steps for the actions we are going to take.

As for the actions, they are based on the results the system delivers:

- in case there definitely are yellow people:

  An announcement appears, telling people to stay away from the platform's edge. The announcement has a timer (e.g. 15 seconds), so it doesn't trigger all the time excessively.[1]

- in case there definitely are red people:

  The camera recording pops up highlighted on the surveillance center's monitors. From there a human is required to either confirm or disregard the alarm. (For feature work, we could use the data that gets created this way to further train the system.) A false alarm would have no meaningful consequences for the train's arrival. However, once a specific alarm gets confirmed by a human being, it is up to the surveillance center's staff to delay a train's arrival (if necessary) and inform other authorities (e.g. police, ambulance, etc., if necessary).

Another interesting idea we had was that the arriving train's driver could get notified in real-time once the alarm has been confirmed by the surveillance center's staff. A form of how this notification could look like is a single LED (red) that shines quite brightly once it is confirmed that there are indeed people on the tracks. This would have the benefit that the arriving train's driver could react even before approaching the next traffic lights. Another idea was to broadcast the surveillance camera's video to the train driver in such a confirmed event, however, that would require expanding the networking significantly between a surveillance camera and a train. These are just two of many possible approaches and they can be evolved over time. For us right now it is important that false alarms - and they will happen - do a) not have a high negative impact on the time schedule of trains and b) further actions can be taken, where further actions are being defined by what a person (the surveillance center's staff) deems to be the correct course of action. The latter adds resistance to our system.
To avoid many short (single point of time) alarms and announcements (e.g. when a person walks too close to the platform for only a very short period of time), the system should only trigger the actions above if there is at least one person too close to the tracks or on the tracks for at least X frames taken in the last N seconds. The value of N does not change once it's configured for the system.

2.8 Probabilistic Graphical Model (PGM) for humans on the track

In order to generate realistic scenes of subway stations, we decided to use a graphical model, which entail various parameters that have an impact on the probability of having a dangerous situation.
Figure 5 shows the conceptual graphical model. The core variable of the model is the positions of the passengers, as a passenger on the track will immediately mean, there is a dangerous situation. However, if all passengers keep their distance from the track, or there are no passengers, the situation is not dangerous at all.
For dangerous events we consider two types:

- Type I: A human lands on the track on accident

[1] Fun facts: Trains or train stations in Frankfurt are already able to produce a general announcement, telling the people to step back from the doors in the event that the train's doors can't be closed. Some older wagon types also use high-pitched beep noises to force people to step back from the door area.

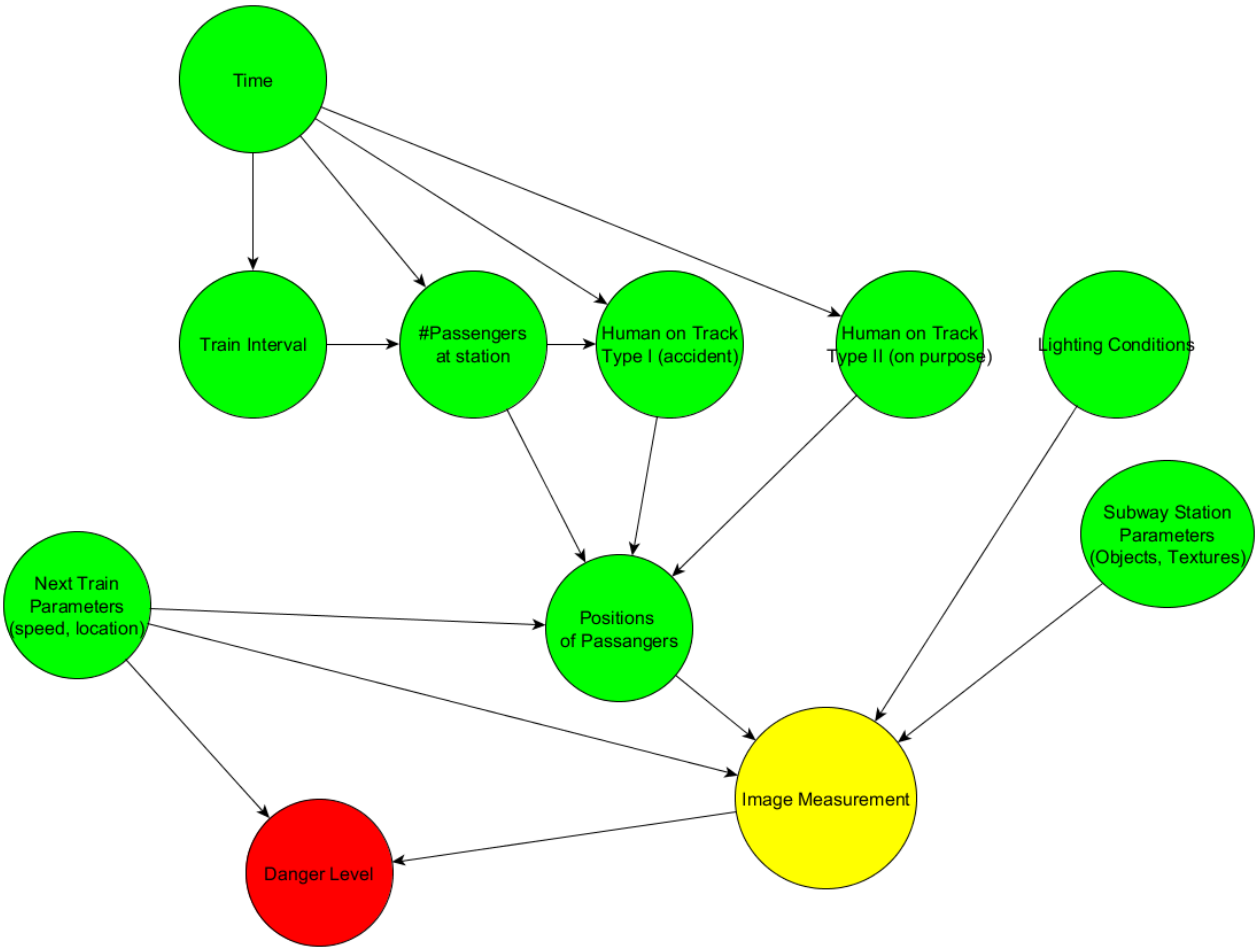

Figure 5: Graphical Model for humans on the track

- Type II: A human lands on the track on purpose

While the probability of an accident should correlate with the number of passengers at the station, we think the type II events happen independently of the number of passengers, that are at the station. The probabilities for both events however, should depend on the time of day, weekday and special events (like a soccer match, a big fair, etc.).
With train interval we mean the time between two trains halting at the station, which can depend on the time of day, day of week and special events, and has an impact on the number of passengers on the station.
The danger level of a situation is determined not only by the fact, that a passenger is on the track, but also by the position and speed of the train relative to that person.

With access to detailed passenger data on train stations, it would have been possible to approximate the relevant probabilities for this graphical model from data. Since we had no access, we tried to estimate the distributions to the best of our knowledge and with reasonable simplifications.

In the following, we will derive realistic assumptions for the generative model of passengers on a train station. For that, we use passenger data of the Frankfurt underground trains. There might be other underground train systems that deviate from this, but we think it suffices as a reference, as we only use it to roughly estimate parameters for a generative model for passengers on a train station.

According to the annual report 2019 from Frankfurt's main transport company, the so-called Verkehrsgesellschaft Frankfurt am Main (VGF), there are on average over 300.000 passengers each day that use the the city's underground train system [9]. Out of the total 84 unique stations, 27 of them are completely underground. Some stations use the same platform for different lines.

There are 9 different underground lines (U1, U2, ..., U9) in Frankfurt and each line interval (frequency) varies between 5 minutes and 30 minutes, depending on the specific line, day and time.

We assume that the majority of passengers does not have to use more than two U-Bahn lines to reach their destination. We believe that this is a fair assumption, given the structure of Frankfurt's underground system. Therefore, we can safely use 400.000 travels per day as the total amount.
A typical passenger's procedure might look like this: A passenger

- enters the premises of a station (buys a ticket)
- goes to the track, where his train leaves
- waits until the train arrives
- enters the train
- leaves the train at the destination
- leaves the premises of the station

If a passenger arrives at a (uniform) random time , the average waiting time is half the train interval. We consider two different categories of occurrences, where a human lands on the track. The first category includes accidents and deliberate pushes, which should in general be positively correlated with the amount of people on the track. The second category includes suicide attempts and graffity sprayers, which we assume usually tend to do their actions during a period of low frequency of trains and/or passengers.
For our model, only the entering, waiting and leaving of the premises are relevant. Since a passenger can be seen twice on surveillance cameras (once when entering, once when leaving), each trip of a passenger results in two possible occurrences on video surveillance. Therefore, on average, around $\frac{400.000}{86} \cdot 2 \approx 9000$ passengers should pass through a station per day, which is about 375 passengers per hour.

Regardless of the intervall time, the expected duration of watiting time for a passenger, that arrives at the station randomly, is half the intervall time. We assume an intervall time of 15 min. Therefore, on average about $\frac{passengers/hour \cdot expected_waiting_time}{trains/hour \cdot intervall_time} = \frac{375 \cdot 5}{4 \cdot 10} = 46.875 \approx 45$ passengers should be on a station at a random point in time.
This is not evenly distributed for the 24 hours of a day. In the morning and afternoon/evening usage is very high, while at night and midday it is very low.

For the simualtion, we simplified this by drawing uniformly from {0, 1, 2, ... 80, 100, 200, 300}, which results in a similar expected number of passengers (45.71) and also has significant probabilites for relevant cases (full station with 300 passengers; no passengers; many passengers, few passengers).

For Type II humans(on purpose) on the track, we had to make educated guesses regarding the probability of occurence. First, according to ... 3% of the delays of underground trains are caused by humans on the track. Therefore this should not be too rare of an occurence. Secondly, in Germany, in the last 10 years, every year between 800 and 1000 suicidal attempts took place on train tracks([10]).

We assume, that only a fraction of these happen in the underground. We couldn't find any statistics on that so for simplicity we assume all of these attempts happen underground.
For the position of the passengers, we defined several constraints:

- There should be a minimal distance between two passengers.
- Passengers will not block entrances and exits.
- Passengers will be assigned a random activity:

The possible activities are:

- wait on a seat
- wait standing
- move (in order to enter, leave or find a waiting place, or enter/depart a train)

Most of the passengers will thus be sampled to stand in a reasonable distance from the safety line. Moving passengers will have a non negligible probability to walk across the safety line.

For the simulation a cut off gaussian distribution seemed enough to generate the position of a passenger.

3 System Implementation View

This section covers the first implementation cycle of the project "Subway Station Hazard Detection" in which we developed a first prototype of the desired system. However, to fulfill all requirements, the system needs to go through several development cycles in which we cover all the various aspects that were stated by the Modeler View. We cover those remaining requirements in the "Future Work" section.

The implemented system consists of several modules:

- The simulation (our sample generator), which can generate various realistic scenarios of a subway station.
- The learning algorithm (a neural network), which can classify the danger level based on the current situation.
- The security protocol, which processes the outputs from the learning algorithm and initiates appropriate process steps to ensure the security of the station.

For the development of these modules we had to divide our tasks as follows:

1. Development and procurement of 3D assets for a realistic representation of subway stations.
2. Implementation of a generator for the automatic and randomized generation of various subway station scenarios. This generator represents the simulated environment for our system.
3. Integration of the distribution of characters from the Modeler View inside the simulation.
4. Implementation and validation of the learning algorithm based on the training data which was generated by the simulation.
5. Development of a security protocol which actively monitors subway stations and, depending on the classification of the situation, initiates the corresponding process step from the security protocol.

Railway stations are very complex and lively environments where a lot of things can happen in a very short period of time. Since our system should be used in general on all camera equipped subway stations, we decided to use a neural network for the classification of the danger level. The basic idea is to use semantic segmentation to train the neural network in such a way that people are recognized and classified into one of the 3 categories: green, yellow or red. Depending on the classification, a corresponding security protocol would decide how to deal with the respective situation and trigger a corresponding security process. In particular, current dangerous situations (people classified as red) and potentially dangerous situations (people classified as yellow) should be recognized. Furthermore, we decided to install a floor marking on each subway platform, which shows the distance of 1.5m to be maintained from the edge of the platform. As soon as a person crosses this marking (without the train entering the station), the system classifies them as yellow (potentially dangerous). If a person is standing on the tracks itself, they are classified as red (dangerous).

Once it was clear how the learning algorithm would be structured, we only had to obtain the relevant training data. We have decided to set up our own simulation, as this gives us more freedom in the representation of the various scenarios and we can actively generate training, validation and test data. Another advantage is that a simulation enables us to train our system step by step and expand it further in future. A key role here was the development and procurement of realistic 3D station assets.

3.1 Development and Procurement of 3D Station Assets

The following graphic components were required to implement a realistic simulation:

- Station models
- Station objects
- Human models
- Textures

3.1.1 Station models

A total of 10 different types of subway stations have been developed, covering the most common station architectures. Figure 6 shows one of the corresponding station types. The number of platforms varies between 1 and 2, while the number of tracks lies between 1 and 4, for each station. It was particularly important to cover all possible combinations of track, wall and platform positions for the camera viewing angles defined in the Modeler View. All station models were developed manually using the Blender modeling software.

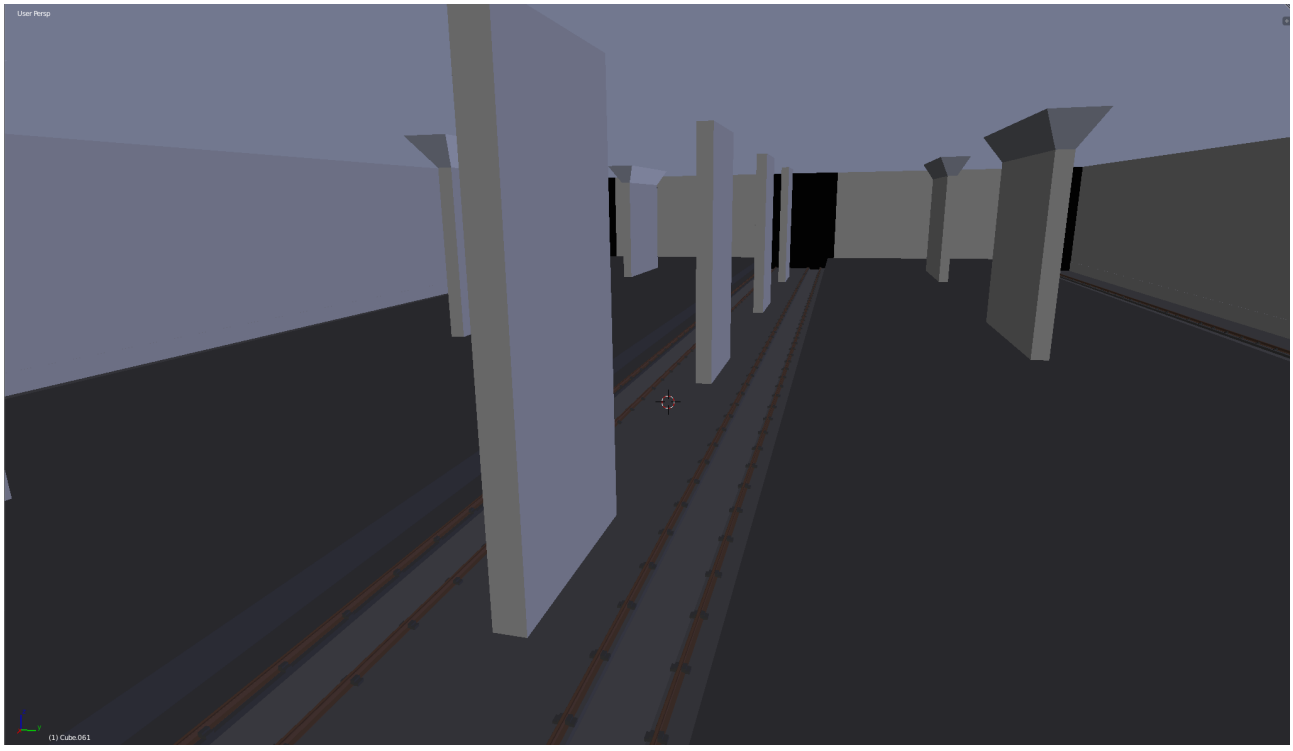

Figure 6: Visualisation of a certain station type, developed in Blender

3.1.2 Station objects

For the first implementation cycle we have limited ourselves to the following station objects:

- Rubbish bins
- Chairs and benches
- Elevators

- Escalators
- Snack machines
- Stairs
- Entrances
- Departure boards

Almost all listed objects were developed manually, using the modeling software Blender. Only the escalator was imported from an external source [11]. Furthermore, were numerous station objects developed in many different design options to cover a wide variety of subway environments. The rubbish bins, chairs and benches feature self-standing and wall-fixed design options. Figure 7 shows all station objects gathered in one place.

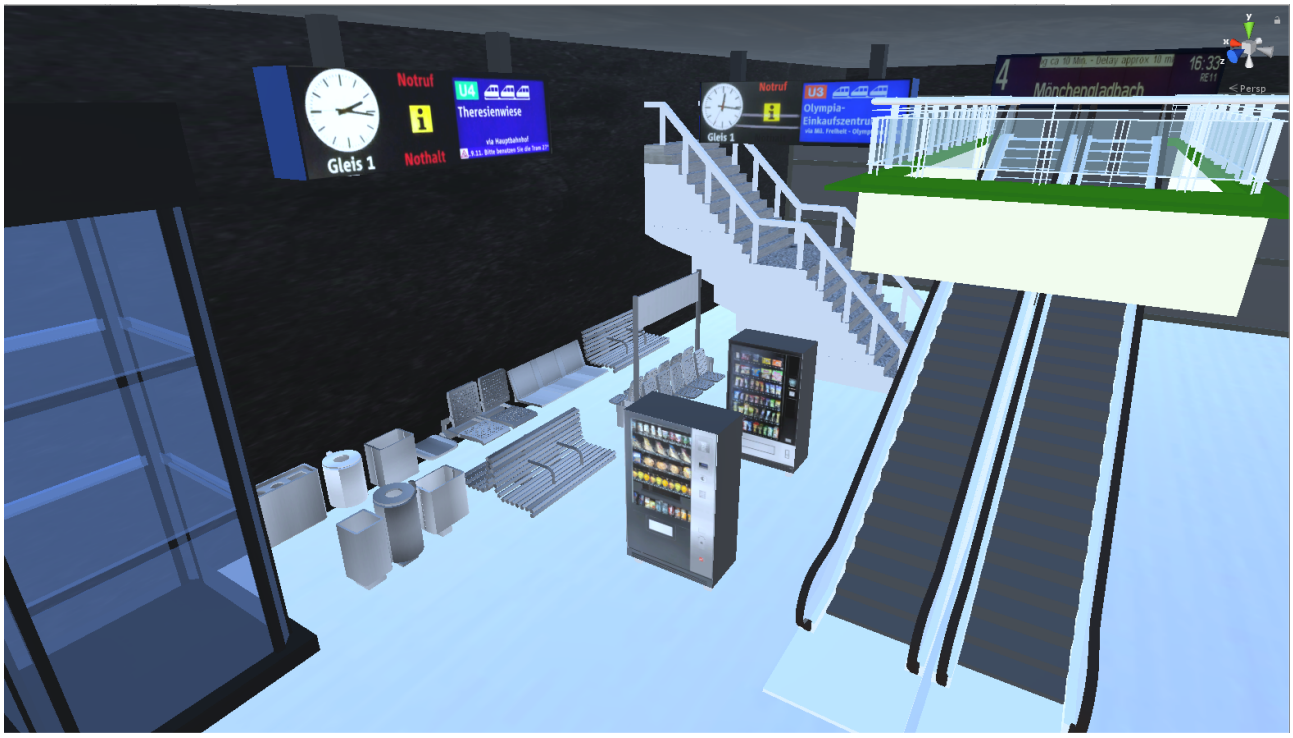

Figure 7: Visualisation of all station objects, which were used for the simulation in Unity

3.1.3 Human models

Due to the complexity of developing a realistic three-dimensional human model, we had to use external data sources [12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31] to obtain our human components for the simulation. We have a total of 25 different characters of which 7 are shown in a sitting position. Figure 8 shows some of the characters that we imported.

Figure 8: Screenshot in Unity showing a group of characters that were used for the simulation

3.1.4 Textures

In order to make our generated stations look as authentic as possible, we needed a large number of high quality textures. A total of 119 different texture packs were imported from the copyleft website Pixabay [32] and integrated into our project.

3.2 Subway Station Simulator

3.2.1 Overview of the Simulation

After all station assets had been provided, our next task was to develop the simulation. The basic aim of this simulation was to generate various randomized station scenarios (with the help of the provided assets) and record these in the form of pictures from the corresponding camera angles, as described in the Modeler View. At the same time, a corresponding ground truth output had to be generated for each recorded photo, in which each object would be correctly classified in form of semantic segmentation. Thus, the simulation was able to generate a set of samples which can be used to train and evaluate the learning algorithm.
Figure 9 visualizes two examples of randomly generated station scenarios and the corresponding ground truth images. We decided to classify everything except the characters and the floor markings with the color white, as part of the station environment. The floor marking should be classified with the color black and the characters should be classified, depending on their current position, either with the color green, yellow or red. In fact, the algorithm places all the station objects randomly at first and then distributes the characters across the station uniformly. To generate the ground truth images, we had to duplicate every single station object as well as the stations itself and texture everything with white colors. Furthermore, we had to make copies of every single character and texture them in green, yellow and red. As soon as a station scenario is recorded, the algorithm duplicates the scenario with the ground truth objects. It replaces the station and all station objects with the white versions and paints the floor marking in black. It determines whether a person is standing behind the floor marking (green character), in front of the marking (yellow character) or on the tracks itself (red character). After the scene is recorded, the algorithm proceeds with the next station scenario and repeats the loop.

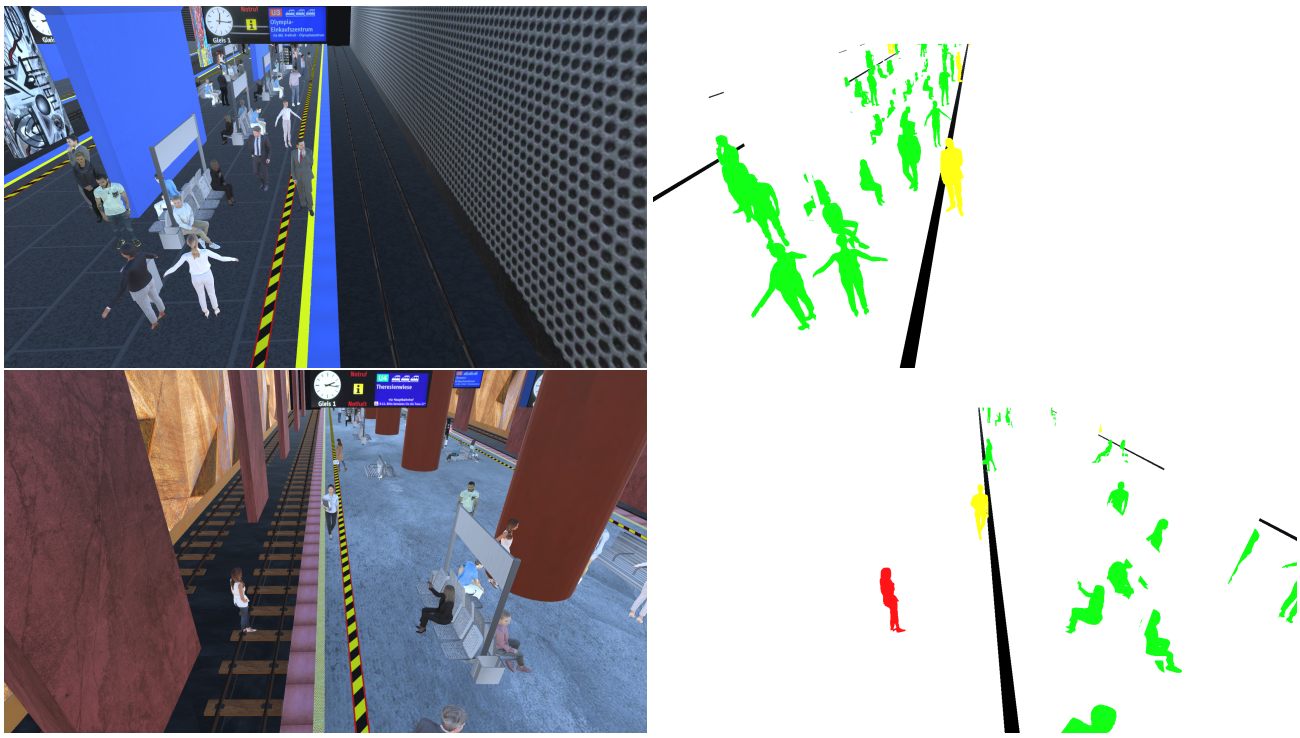

Figure 9: Two examples of randomly generated station scenarios and the corresponding ground truth images

3.2.2 Simulator Specifications

For the development of this simulation, we used the Unity game engine as it contains a lot of useful functionalities for the development of realistic 3D scenes and our team already had previous experience with this tool. Another key feature of Unity was that it provides very complex lighting options which can be customized. Since this was our first implementation cycle, we chose to have constant lighting in the scene without any reflections or shadows. However, to implement a fully functional system it is indispensable to develop certain illumination models for our simulation in future work. Furthermore, we decided to texture our stations manually because of time constraints in developing fully randomized and realistic texturing algorithms but want to include this in a future version of the software. To still provide a wide variety of station environments, we developed 5 unique station designs for each of the initial 10 station types resulting in a total of 50 different and unique subway stations. Figure 10 shows 6 different station types that were textured manually to look as realistic as possible. Another important simplification to be noted, besides the pre-textured stations and the constant illumination ratios, is that we have not implemented a camera noise model that revises the camera recordings in Unity. The current prototype takes normal pictures of the station without any occurrence of noise. These aspects are subject for a future version of the simulation and we revisit them in the "Future Work" section. Additionally, for every station scenario there are two camera recordings which are taken from both endpoints of the station to achieve a higher coverage of the station environment. The cameras itself are placed on the ceiling at an angle of 30° and cover the space between the tracks and the station platform. For the recorded images we use a resolution of 1080p.

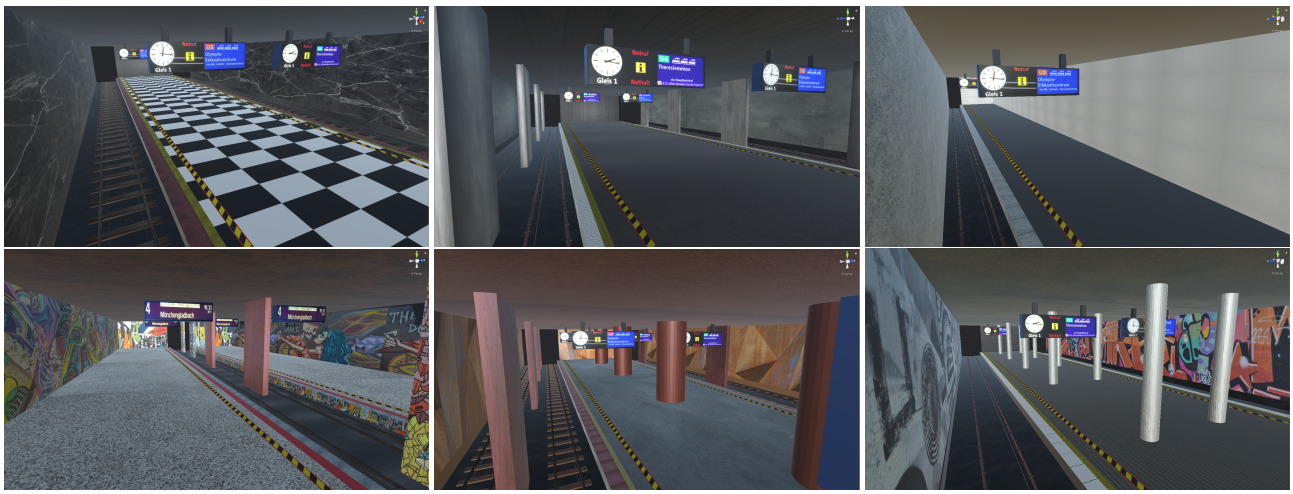

Figure 10: Various subway station types, which were textured manually

3.2.3 Simulator Configurations

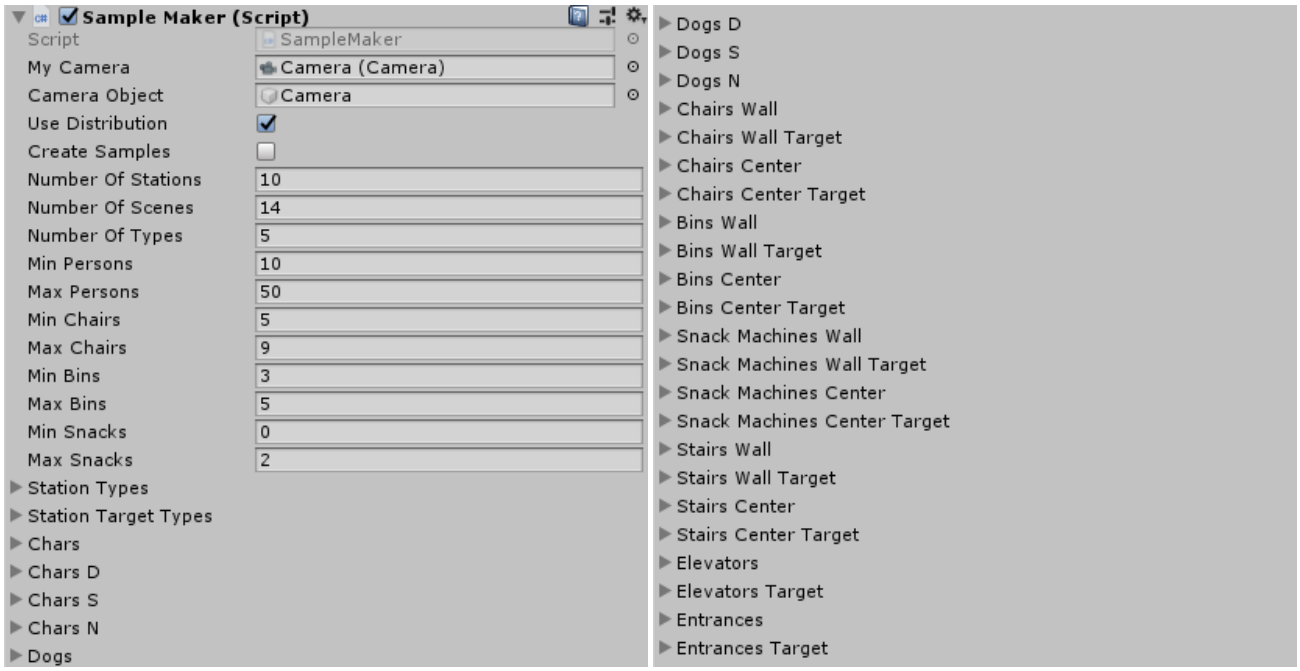

Figure 11: Screenshot of the simulator UI developed with C# in Unity

The whole simulation was developed using C#. It supports the dynamic expansion with further design options of existing station assets and can be easily extended to support completely new station assets. For this feature we developed a simple user interface in form of dynamic and extensible lists of public game objects which can be added to the simulation. Figure 11 visualizes the mentioned UI with all available options. Within this UI the user can specify the camera object, which is used for the recordings of the simulation. If the "Use Distribution" checkbox is selected, then the simulation will distribute the characters as specified in subsection 2.8 of the Modeler View. Subsection 3.3 deals with the implementation of the distribution of characters and describes this topic in more detail. Furthermore, if the user selects the "Create Samples" checkbox, all station images as well as the corresponding ground truth images will be saved in separate folders. We implemented this

option for debugging purposes. We also gave the user the ability to determine, how many station environments should be generated (textured stations with station objects) as well as how many station scenarios on each of these station environments should be produced (textured stations with station objects and characters). The user can also specify the spawn ratio for the following objects: persons, chairs/benches, bins and snack machines. The "Station Types" drop-down menu contains a list with all 50 stations, which were manually textured. Inside the Target version of the drop-down menu you can find the corresponding ground truth stations which are textured completely in white except for the floor marking which is painted in black. This same logic applies for all of the station objects. For the characters we have 4 different drop-down menus which contain: the real versions of the characters, the green versions, the red versions and the yellow versions. After all game objects have been added to the user interface and all configurations have been made, the user can initiate the program to start recording the station scenarios as well as the corresponding ground truth images. Figure 12 visualizes a randomly generated station scenario, produced by our simulation.

Figure 12: Screenshot of a randomly generated station scene

3.3 Implementation of the Character Distribution

3.3.1 Generating random numbers from a normal distribution

In Unity and C# there is no library to generate random numbers from a normal distribution. It only supports the creation of uniform distributed numbers. However, by using the Box-Muller Method[33] [34] it is possible to create random numbers from a normal distribution by using a uniform distribution. For this purpose it uses trigonometric functions, which are notoriously slow. Because of this, we decided to use a slightly different method called Marsaglia polar method[34].
In the first step we generated 2 uniform distributed random numbers $U_1, U_2 \sim \mathcal{U}(0,1)$ and transformed them with the help of the Marsaglia polar method[34] (algorithm 1) to $Z_1, Z_2 \sim \mathcal{N}(0,1)$. Let Z be our generated random number.
Our second step is to transform Z from $\mathcal{N}(0,1)$ to $\mathcal{N}(\mu, \sigma^2)$, by scaling with σ and shifting it by μ (algorithm 2).

Algorithm 1 MarsagliaPolarMethod()

Ensure: $Z \sim \mathcal{N}(0,1)$
  repeat
    U_1 = Random.Range(0,1) -1
    U_2 = Random.Range(0,1) -1
    $Z = u_1^2 + u_2^2$
  until $(Z \leq 1 \quad and \quad z \neq 0)$
  $Z = \sqrt{\frac{-2 \cdot \log(Z)}{Z}}$
  return Z

Algorithm 2 Gaussian(Z, μ, σ)

Require: $Z \sim \mathcal{N}(0,1), \mu, \sigma \in \mathbb{R}$
Ensure: $Z \sim \mathcal{N}(\mu, \sigma^2)$
  return $\mu + z \cdot \sigma$

Unfortunately there is a problem that occurs when we use this distribution for our simulation. We don't want to get object coordinates outside a given range, because we don't want to place characters outside of the subway station or in the wrong area. Because of this, we add two more parameters min and max. Now if we get a random number outside of $[min, max]$ in the last step(algorithm 3), we simply generate a new random number.

Algorithm 3 RandomNormal(μ, σ, min, max)

Require: $\mu, \sigma, min, max \in \mathbb{R}$
Ensure: $Z \sim \mathcal{N}(\mu, \sigma^2)$, min$\leq Z \leq$max
  repeat
    Z = MarsagliaPolarMethod()
    Z = Gaussian(Z, μ, σ)
  until min$\leq Z \leq$max
  return Z

3.3.2 Character Distribution

The algorithm for placing people in the scene with a distribution, chooses the number of characters uniformly in $\{0, 1, 2, \ldots, 79, 80, 100, 200, 300\}$ as required in the Modeler View.

$$numberOfCharacters = \text{Uniform}(\{0, 1, 2, \ldots, 79, 80, 100, 200, 300\})$$

In general we place every person on the platform. However, to keep the number of samples in the validation set equal, we say that for every placed character there is a $\frac{1}{200}$ probability that an additional character will be spawned in the rail area.
In the required distribution there are two types of persons on the tracks. The first one has probability $\frac{1}{309600}$, and the second one $\frac{1}{3600}$. To obtain the correlation to the people on the platform, we chose a probability of $\frac{1}{10000}$ for every person to be placed in the rail area instead on the platform.
Every person on the platform could be in one of the three states: standing, sitting and walking. In our current simulation we don't make a difference between standing and walking people, that's why we only use the states sitting and standing. A person sits with probability $\frac{1}{3}$ and stands with probability $\frac{2}{3}$. If there is no more seat available, then the rest of the people stand.

The angle for the rotation on the y-axis is uniform distributed by

$$yRotation = \text{Uniform}(\{1, 2, \ldots, 360\}).$$

The rotation angle on the x-axis is currently fixed on 90 degrees to the platform. That is a rotation on the x-axis of every person by

$$xRotation = 0.$$

In a later version we should adjust this to a distribution, which makes more sense.
For the position on the x-axis of every person on the platform we use the normal distribution $\mathcal{N}(\mu, \sigma^2)$, where μ is the center on the x-axis of the platform, and σ is $\frac{1}{4}$ from the width of the platform. If we get a number for the x-position, which is outside the platform we generate a new one. The calculation of the x-position of a character can be described by algorithm 3 as

$$xPosition = \text{RandomNormal}(center, \frac{width}{4}, end1, end2),$$

where

- $center$ is the x coordinate of the platforms center
- $width$ is the platforms length on the x-axis
- $end1 = center - \frac{width}{2}$
- $end2 = center + \frac{width}{2}$

A visualization of the distributed parameters is given in Figure 13.

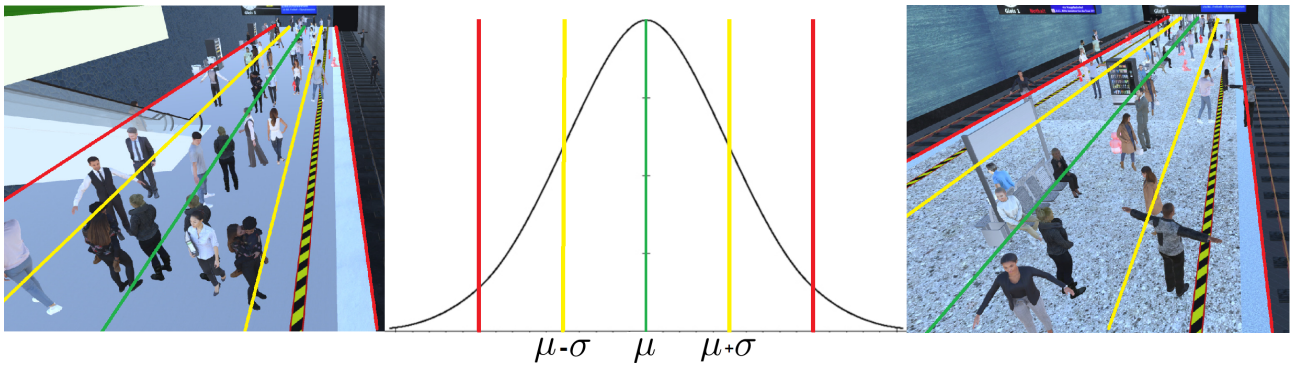

Figure 13: Distribution for character position on x-axis

The position on the y-axis is uniform distributed by the length of the platform

$$yPosition = \text{Uniform}(0, length).$$

The position and the rotation of a person who sits on a seat is uniformly distributed on all available seats. The position on the x-axis is unform distributed by width of the platform

$$xPosition = \text{Uniform}(0, width).$$

Before a character is placed in the scene, another algorithm checks if it collides with other placed objects. If this occurs, the character will be placed in a different position.

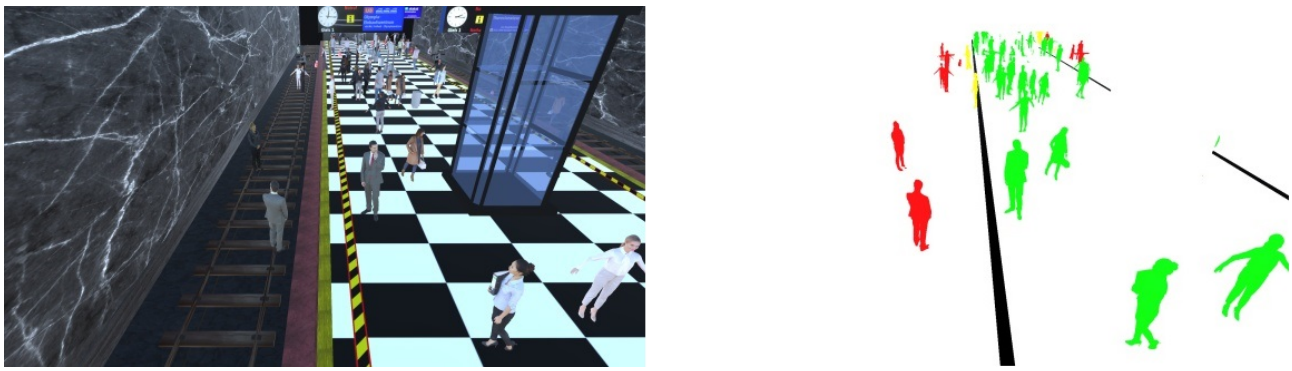

Figure 14: Subway segmentation example with input picture and ground truth output

3.4 Subway Station Segmentation

The first task of our subway station hazard detection, is the subway station segmentation based on semantic segmentation. For the implementation we are using Python and the library PyTorch. We decided to use semantic segmentation, to detect characters on the platform, instead of real-time crowd density estimation[35] or something like that, because our system should work on every subwaystation, without adaption.
Our segmentation classify each pixel of an image from a subway station to one of the following five classes.

- white - background
- black - security line
- green - character in save area
- yellow - character near the dangerous area
- red - character in the dangerous area

In figure 14 you can see an example of our semantic segmentation.
The background is everything that's part of the considered subway station like chairs, bins, walls, trails, elevators, snack machines etc.
The security line is only the area of our selfmade security line on the platform.
On a platform where tracks are on both sides like in Figure 14, the green area is the whole platform between the two security lines. On a platform where is only on one side a track and on the other side a wall, the green area is the whole platform between the wall and the security line.
Every character who is between the track and the security line, will be detected in the part of 'near the dangerous area'.
The dangerous area is the whole area next to a platform where the tracks be.

3.4.1 The Architecture

To perform the subway segmentation we use a convolutional neural network named SegNet[36]. It is a deep encoder-decoder architecture for semantic segmentation researched and developed by members of the Computer Vision and Robotics Group at the University of Cambridge. SegNet[36] is actually developed for the Cityscapes Dataset[37] for semantic urban scene understanding, but it also achieves good results on other datasets like indoor scenes for example. The architecture of SegNet[36] is shown in figure 15. It's a typically encoder-decoder architecture followed by a pixelwise classifier, where

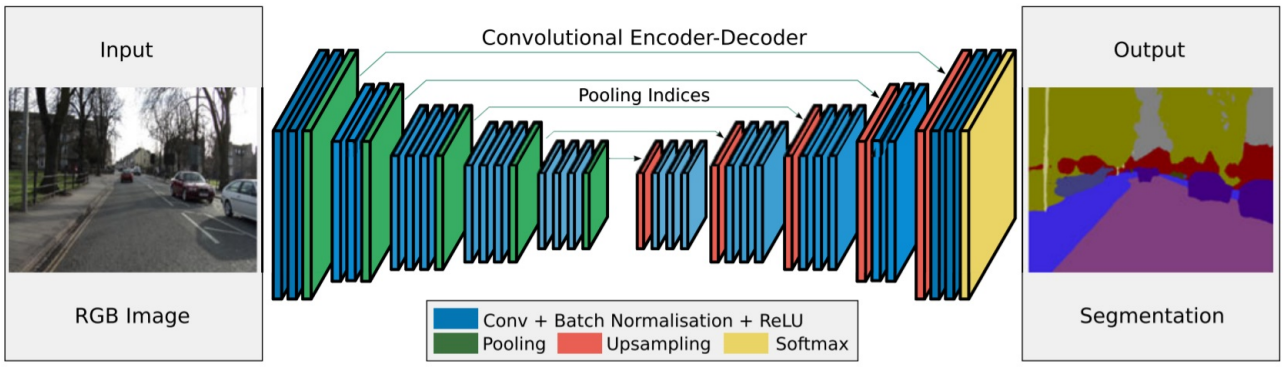

Figure 15: SegNet Architecture, Fig.2 in [36]

each convolutional layer of the encoder consists of more than one convolutional layer with batch normalization and ReLU as activation function, followed by max-pooling to enable a high number of feature-maps in the deep layers. Each layer of the decoder is the inverse to the opposite layer of the encoder, but the decoder layers using upsampling instead of maxpooling to restore the original image resolution. The use of max-pooling indices in the decoder layers to perform upsampling of low resolution feature maps is one key ingredient of SegNet[36]. This has the important advantages of retaining high frequency details in the segmented images and also reducing the total number of trainable parameters in the decoders. On top of that there's no fully connected layer, what makes the prediction speed very fast. The entire architecture can be trained end-to-end using stochastic-gradient for example. In the case of hazard detection we need a fast prediction, that's why we are using SegNet[36].

	Layer	Filters	Size	Input	Output		Layer	Filters	Size	Input	Output
1	conv	64	3×3	$320 \times 576 \times 3$	$320 \times 576 \times 64$	19	up	512	2×2	$10 \times 18 \times 512$	$20 \times 36 \times 512$
2	conv	64	3×3	$320 \times 576 \times 64$	$320 \times 576 \times 64$	20	conv	512	3×64	$20 \times 36 \times 512$	$20 \times 36 \times 512$
3	max		2×2	$320 \times 576 \times 64$	$160 \times 288 \times 64$	21	conv	512	3×3	$20 \times 36 \times 512$	$20 \times 36 \times 512$
4	conv	128	3×3	$160 \times 288 \times 64$	$160 \times 288 \times 128$	22	conv	512	3×3	$20 \times 36 \times 512$	$20 \times 36 \times 512$
5	conv	128	3×3	$160 \times 288 \times 128$	$160 \times 288 \times 128$	23	up		2×2	$20 \times 36 \times 512$	$40 \times 72 \times 512$
6	max		2×2	$160 \times 288 \times 128$	$80 \times 144 \times 128$	24	conv	512	3×3	$40 \times 72 \times 512$	$40 \times 72 \times 512$
7	conv	256	3×3	$80 \times 144 \times 128$	$80 \times 144 \times 256$	25	conv	512	3×3	$40 \times 72 \times 512$	$40 \times 72 \times 512$
8	conv	256	3×3	$80 \times 144 \times 256$	$80 \times 144 \times 256$	26	conv	256	3×3	$40 \times 72 \times 512$	$40 \times 72 \times 256$
9	conv	256	3×3	$80 \times 144 \times 256$	$80 \times 144 \times 256$	27	up		2×2	$40 \times 72 \times 256$	$80 \times 144 \times 256$
10	max		2×2	$80 \times 144 \times 256$	$40 \times 72 \times 256$	28	conv	256	3×3	$80 \times 144 \times 256$	$80 \times 144 \times 256$
11	conv	512	3×3	$40 \times 72 \times 256$	$40 \times 72 \times 512$	29	conv	256	3×3	$80 \times 144 \times 256$	$80 \times 144 \times 256$
12	conv	512	3×3	$40 \times 72 \times 512$	$40 \times 72 \times 512$	30	conv	128	3×3	$80 \times 144 \times 256$	$80 \times 144 \times 128$
13	conv	512	3×3	$40 \times 72 \times 512$	$40 \times 72 \times 512$	31	up		2×2	$80 \times 144 \times 128$	$160 \times 288 \times 128$
14	max		2×2	$40 \times 72 \times 512$	$20 \times 36 \times 512$	32	conv	128	3×3	$160 \times 288 \times 128$	$160 \times 288 \times 128$
15	conv	512	3×3	$20 \times 36 \times 512$	$20 \times 36 \times 512$	33	conv	64	3×3	$160 \times 288 \times 128$	$160 \times 288 \times 64$
16	conv	512	3×3	$20 \times 36 \times 512$	$20 \times 36 \times 512$	34	up		2×2	$160 \times 288 \times 64$	$160 \times 288 \times 64$
17	conv	512	3×3	$20 \times 36 \times 512$	$20 \times 36 \times 512$	35	conv	64	3×3	$160 \times 288 \times 64$	$160 \times 288 \times 64$
18	max		2×2	$20 \times 36 \times 512$	$10 \times 18 \times 512$	36	conv	5	3×3	$160 \times 288 \times 64$	$320 \times 576 \times 5$

Table 1

Our implementation details of SegNet[36] are shown in table 1. We use an input size of $320 \times 576 \times 3$, 26 convolutional layers with kernel size 3×3, 5 max-pooling layers with kernel size 2×2 and 5 upsampling layers with kernel size 2×2. On the output feature map of size $320 \times 576 \times 5$ we applied a softmax-layer. The real output featuremap consists of 5 images, one image for every class (white, black, green, red, yellow), where every single value in one image is the probability of this class for the pixel in the predicted image.

3.4.2 Loss Function

As loss function we are using PyTorchs weighted cross entropy loss[38] from the torch.nn module. This criterion combines log softmax and negativ log likelihood loss. Cross entropy loss is usefull by classification problems with more than two classes, in the way of semantic segmentation we have a classification problem for each pixel of an image.
The weighted categorical cross entropy loss for a single pixel of an output image of our application can be described as:

$$\begin{aligned} \text{loss}(x, class) &= \text{weight}[class]\left(-\log\left(\frac{\exp(x[class])}{\sum_{c \in Classes} \exp(x[c])}\right)\right) \\ &= \text{weight}[class]\left(-x[class] + \log\left(\sum_{c \in \text{Classes}} \exp(x[c])\right)\right) \end{aligned} \tag{1}$$

Figure 16: Categorical cross entropy loss[38]

,where

- Classes = {white, black, green, red, yellow}
- $class \in$ Classes is the correct class for x

First we calculated the weight for each class only proportionally by sum all pixels p of an image in the training dataset T, which has to be predicted as this class:

$$weight[class] = \frac{\min\limits_{c \in class} \text{pixel(c)}}{\text{pixel}(class)} \tag{2}$$

with

$$\text{pixel}(class) = |\{p \in img \mid \text{pred}(p) = class \mid img \in T\}|, \tag{3}$$

where pred(p) is the target of pixel p.
Later in part of the training, we noticed that the weight of the white class is to small, and we get a better result, if we multiply it with 10.

3.4.3 Training

For training and validation we created a training (50.000 pictures) and validation (10.000 images) dataset. We trained the neural network 75 epochs on the training dataset with a Nvidia Volta GPU. All pictures and target images of both datasets are self generated from our subway station generator. We don't use the distribution for placing the people here, to make that we have enough people on the tracks area. The people in the images are evenly distributed on the platform, that shows more then the reality is, but that's how we could ensure, that our neural network will learn every situation, and we reduce those situations where it fails. For backpropagation we are using stochastic gradient descent.

3.4.4 Result

For evaluation we use the validation set of 10.000 images and calculate precision and recall for each class in every epoch.
We get a result of:

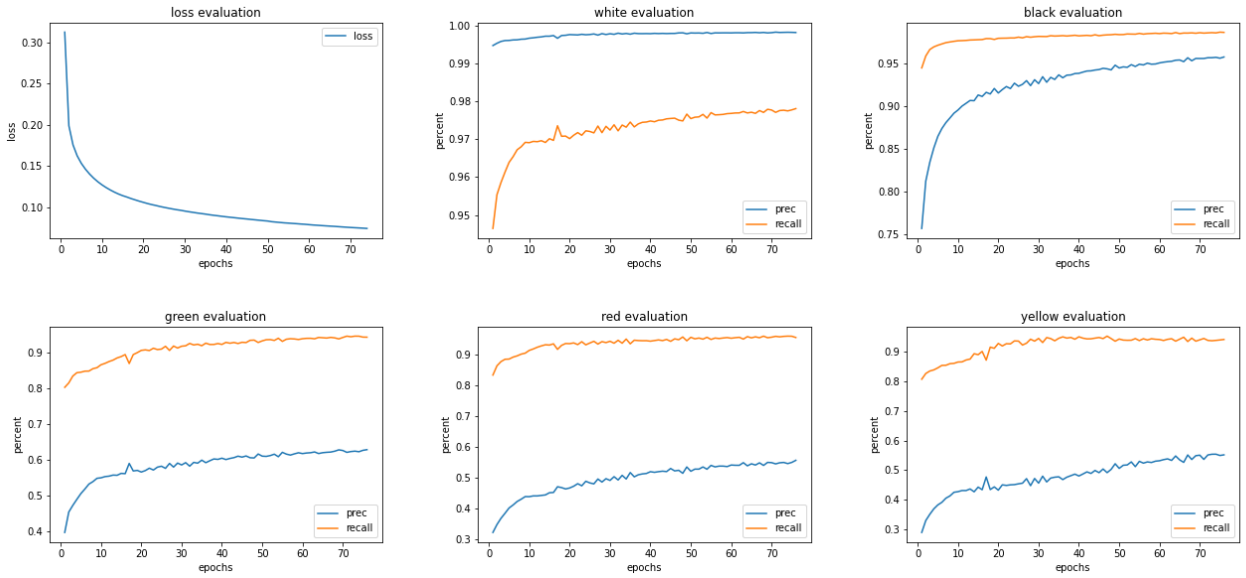

Figure 17: Training evaluation

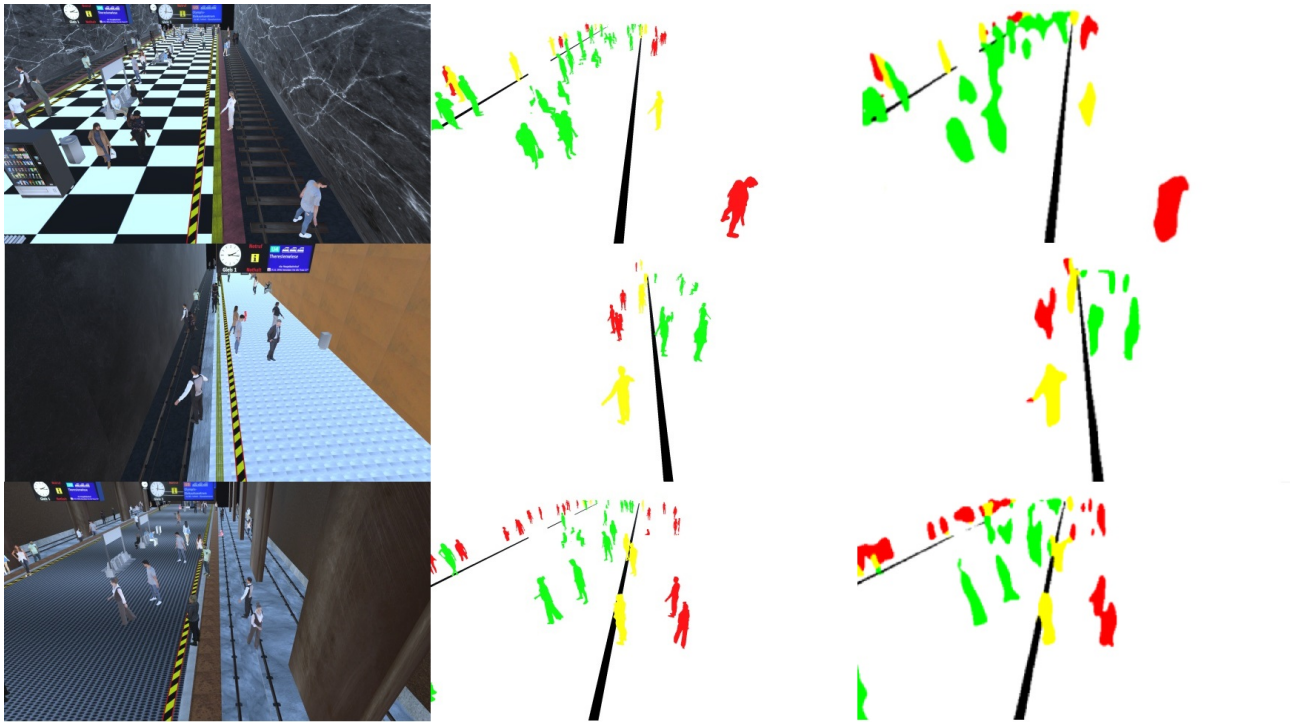

Figure 18: Evaluation Examples: left: original, center: ground truth, right: model prediction

class	**precision**	**recall**
white	100%	98%
black	95%	99%
green	62%	93%
red	54%	96%
yellow	54%	94%

The high recall in every class shows, that our network learned to recognize the security line, and

how to classify characters in the right area. The precision of 100 percent for the white class and the precision of 95 percent on the black class shows that it learned really good, what the background and the security line is. The lower precision for the three classes of characters in combination with a high recall over 90 percent means in general, that the neural network doesn't make really sharp borders by classifying the characters. Figure 18 shows several outputs of the trained CNN.

4 Evaluation & Validation

In this chapter, we evaluate the provided system of the implementation team using two kind of data. The first evaluation is based on a validation dataset. The validation dataset was not used for training and is drawn randomly from the distribution of the generative model.
The second evaluation is based on a variety of real world images from various sources. Lastly, we give validate, whether the system meets the performance requirements specified in the User View and give recommendations to improve the performance of the current approach or of the overall system by other means.

4.1 Evaluation of the trained System

4.1.1 With Validation Set (simulated Data)

Figure 18 shows several examples of simulated data, ground truth and segmentation predictions of the CNN, where humans were present on the track. The overall predictions are reasonable. However, edges tend to get blurred out.

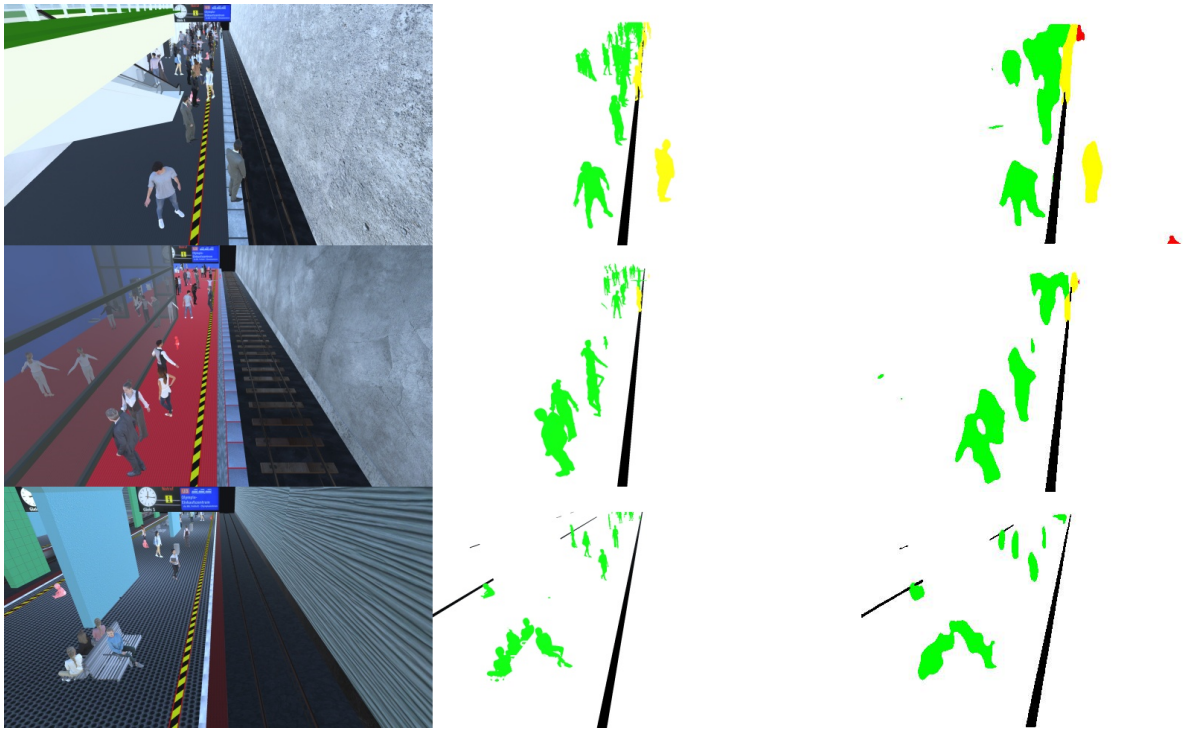

Figure 19: More Evaluation Examples: left: original, center: ground truth, right: model prediction

We also looked at some outputs for images, where no people were spawned on the track. In two of these cases, some red artifacts were generated by the model prediction. They seem to be caused by either specific textures of the wall (see uppermost example) or people, that were just on the edge of the security line.

This is why it seemed reasonable for the first implementation to use a simple thresholding algorithm to determine the danger level of a Segmentation output from the implemented Neural Network. Since the camera positions are fixed and show the same scene from approximately opposite positions, each unoccluded object will generate approximately the same amount of pixels in both images, independent from the exact position of the object. (If the object is far away

from one camera, it is close to the other camera.) Therefore we chose a threshold X, and if more than X pixels in both images are classified as red, the scene is classified as dangerous. If at least one pixel in the target images is red, then the ground truth of the situation is set to dangerous.

We used a python script to count the pixels for the NN-outputs and ground-truths.

From this, we calculated the necessary values to generate ROC and Prec-Recall-Curves (false positive, false negative, true positive, true negative, precision, recall).

Figure 20 shows a plot of the accuracy against the chosen threshold. Best results of around 70% accuracy are achieved with threshold values between 120-200.

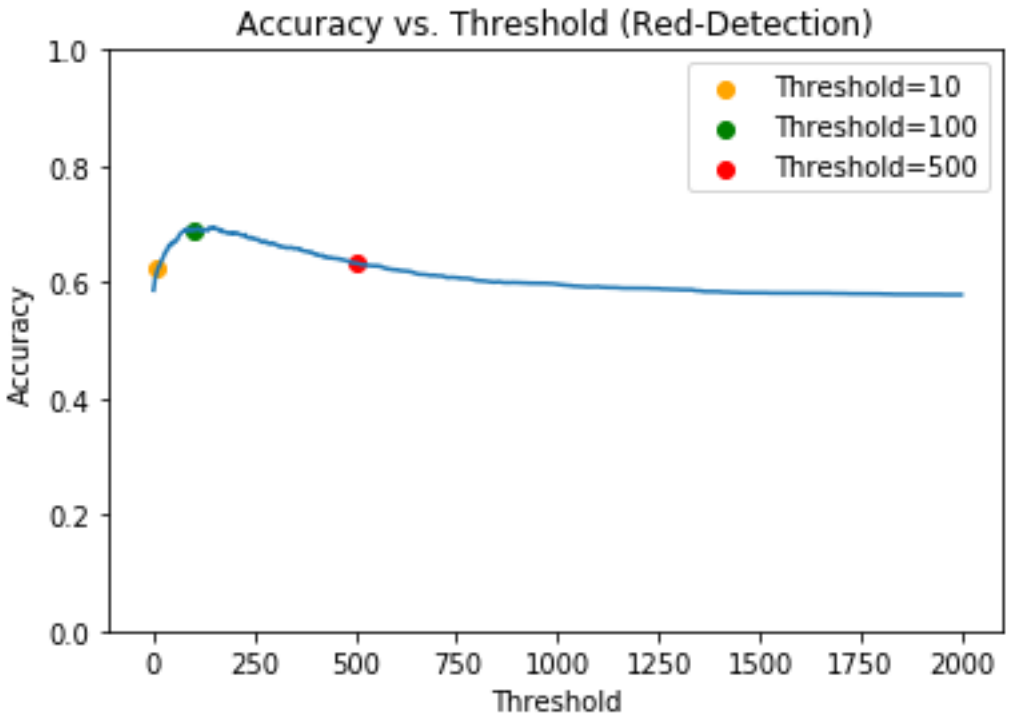

Figure 20: Accuracy for correct predictions based on the threshold

More informative is the ROC-Curve, where the true positive rate (correctly predicted dangerous situations) is plotted against the false positive rate (system deems a harmless situation dangerous). Furthermore, we added some values for the threshold to show, how the threshold affects the "moving" along the ROC-curve.
In this first implementation we weren't able to get a true-positive-rate greater than around 90%, in which case the false positive rate is $\approx 50\%$. If an employee wants to set a maximum on the false-positive rate (e.g. 20%) the true positive rate drops accordingly. In later cycles of the development phase, the ROC-curve is expected to improve. Furthermore, the curve might not be dependant on a threshold value in future implementations, but a system could directly produce a danger percentage or even a more specific description of why a situation is dangerous.
Another curve worth considering is the Precission-Recall-Curve, shown in Figure 22. The precision is defined as

$$Precision = \frac{tp}{tp + fp}$$

where tp is the number of true positives and fp the number of true negatives. A high precision means,

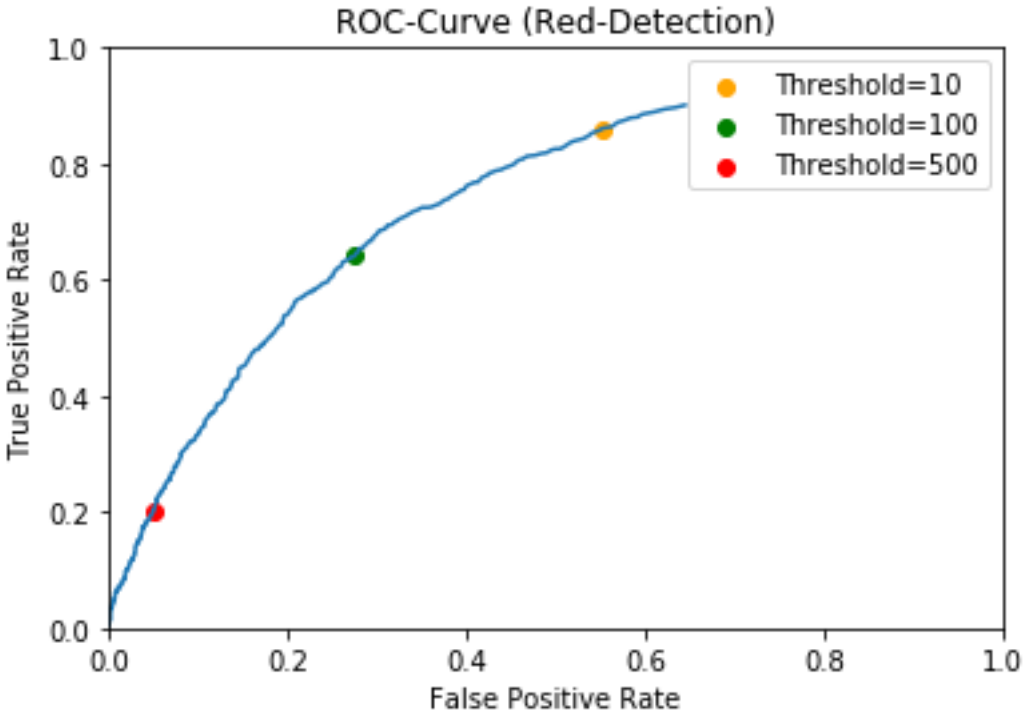

Figure 21: ROC for red segmentation

that a system makes almost no false positive predictions, which in our case corresponds to a false alarm.
The Recall is defined as

$$Recall = \frac{tp}{tp + fn} \tag{4}$$

again, with tp being the number of true positives and fn the number of false negatives.
A high recall in our case means, that the system doesn't miss many dangerous situation. In general, we try to maximize both precision and recall. But it is also sensible, to demand, that the precision should not be below a certain value, as this would lead to delays of trains.

The same analysis was done for the yellow classification output. Results are shown in Figure 23 and Figure 24.
The results for the yellow classification, i.e. detection of people, that have crossed the safety line, shows much better results, as the detection of people on the tracks. For example, with a threshold of 100 pixels, the TPR is at 78% and FPR below 10%.

4.1.2 With Images from Video (real Data)

An important topic to look at in general is how well models trained in simulation generalize to reality. Before a system can fully launch and enter our daily lifes, it must first prove itself. That is, it is required to fulfill its task successfully within our expectations for the real world. Such expectations can include the functionality, the overall behavior, the error rate, the robustness of the system, its flexibility, and much more depending on the application itself.
In this particular subsection, we will talk about performing an evaluation of our prototype system using real world data. To be more concrete, this means our prototype we created gets tasked to do segmentation on pictures taken from real underground station surveillance cameras. However, given

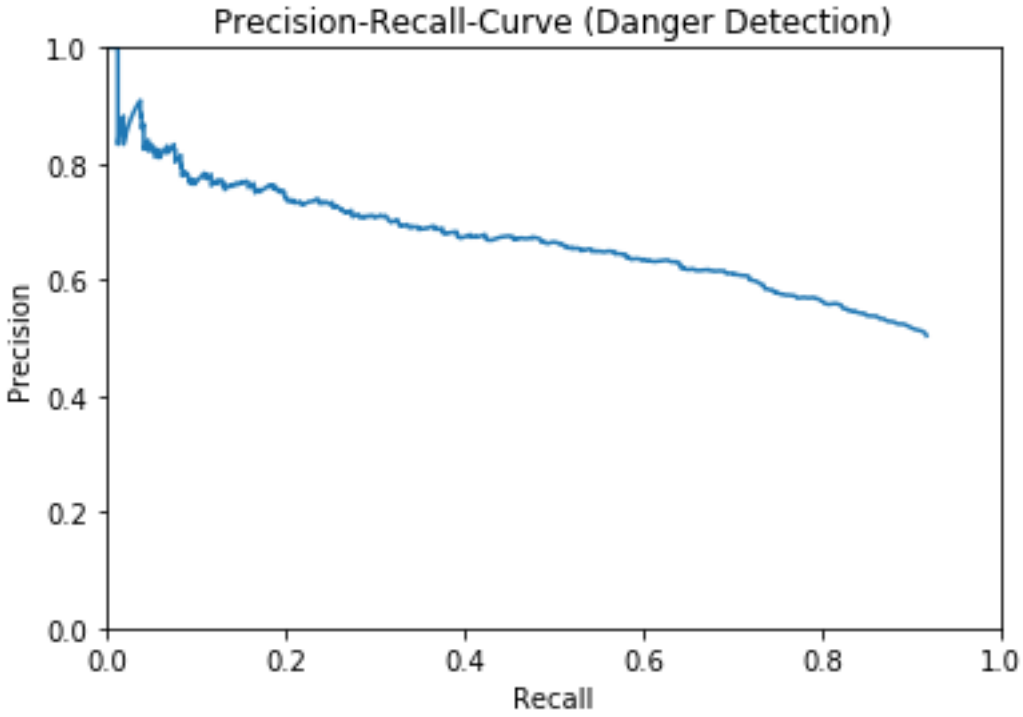

Figure 22: Precision-Recall-Curve for red segmentation

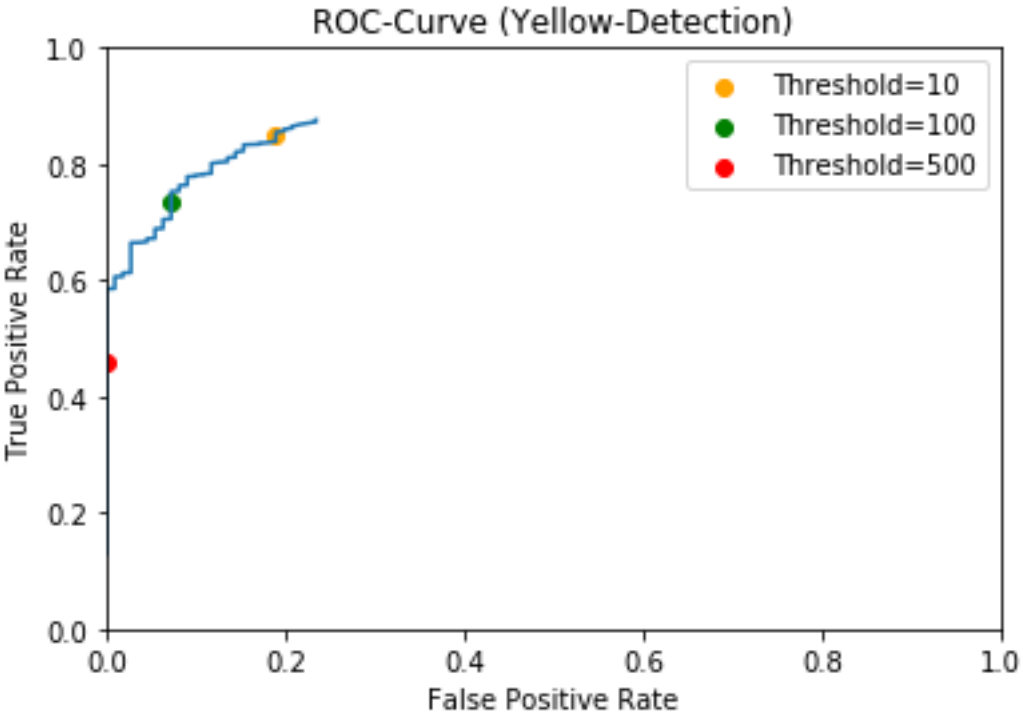

Figure 23: ROC for yellow segmentation

how hard it is to acquire such video data without breaking various German laws, the evaluation will use only videos publicly available on the internet. One major downside of this is that these videos often contain additional editing and suffer loss of quality by being uploaded and compressed. Under

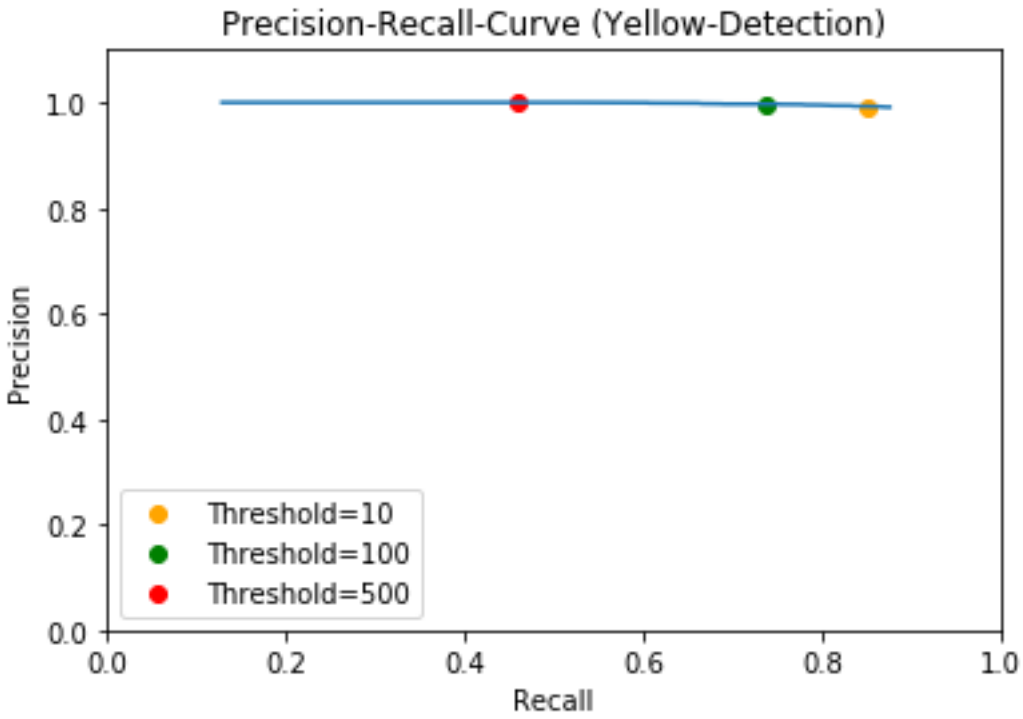

Figure 24: Precision-Recall-Curve for yellow segmentation

Figure 25: Caption

ideal circumstances, we would not have to accept these conditions. To avoid getting screwed by these hindrances, we will take a closer look at the prototype's output image after each step, analyze it and make adjustments to the input.
Regarding the expectations we have, given that this prototype of ours is the very first iteration, and the fact that it was solely trained using computer-generated graphics, before running these experiments we did not expect great results with using real world data. Prior to the first runs using real world data, we expected it to perform much worse than in the evaluation with simulated data.
For our evaluation with real world data, we decided to use a surveillance camera video created in 2018 that got picked up by the Russian television sender Russia Today (RT) and uploaded on Youtube.[2]
The original video was taken by a surveillance camera in an underground station in Cologne, Germany, in 2018. It shows different kinds of people standing, sitting or walking across the entire platform until at some point (timestamp: approx. 57 seconds into the video) one person intentionally pushes another person onto the tracks. Fortunately, there is no train. There are only few videos of this category available on the internet and we chose this one in particular because out of all the other options we had, this one featured the highest video material quality and its point of view does align with the point of view we used for our training.
As far as the video is concerned, it contains editing - primary the logos of RT and the police in Nordrhein-Westfalen. Without looking at the future results, we expect these logos to cause some misdetection by our prototype. Another important attribute of the video is that human faces (heads) are censored. This is required by German law, else it would have been allowed to be broadcast on television and internet. The low quality, the logos, the censorship and the reflections are problems the prototype has not yet learned to handle yet in the training simulation. As if this wasn't enough already,

[2]The video can be viewed here: `https://www.youtube.com/watch?v=r9TzCc8Z_N0` [15th September 2020 06:06]

Figure 26: Preview of the video ("Koelner Schubser")

we have to face the consequences of a design decision we committed to earlier: To help recognize people in potential danger ("yellow" people), we demanded that every underground platform gets a new line to indicate that the platform's edge is only 1.5 meters away. Such warning lines already exist in the real world, however, their design and distance vary a lot (some are plain white, some are yellow, etc.). The line our team wants to add onto any underground platform has a red outline and a black-and-yellow warning look in its center. Figure 27 shows the design concept. Note that the prototype has already been trained with such a line in mind.
A problem that occurs with this new requirement is that now there exist virtually zero real world video sources that show an underground platform with such a line. Therefore, if we want to do an evaluation with real world data, such as with the "Koelner Schubser" video presented earlier, there is no way around manually editing the videos - or rather the single frames - to contain the new line. Thus, the entire set of 77 images we have taken from the "Koelner Schubser" video mentioned above has been manually edited to now contain the new line. Figure 28 shows an edited frame containing the new line.

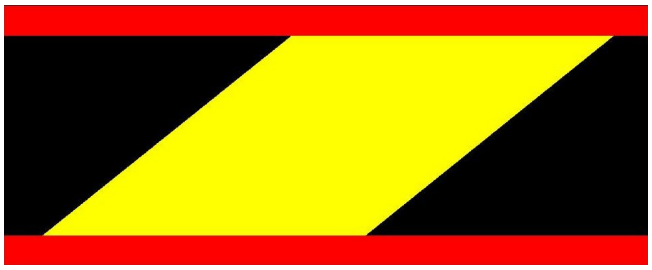

Figure 27: Design of new warning line

Figure 28: Frame 28: Edited to include the new line

The current (first iteration) prototype can only handle single frames. Therefore it is required to first break the video down into single frames before handing them over to the prototype. We accomplish this by taking a frame for each second of the video. Although the linked video is longer, we end up with 77 frames that are relevant to us. These will be fed into our prototype.
The output of the prototype will be a segmentation image (featuring white, black, green, yellow, red).

In order to evaluate the output, we need something to compare it to. This is accomplished by the ground truth which had to be created manually by hand. Due to time restraints we only managed to create 13 such ground truths for our set of 77 frames. One labeling tool used was Hitachi's Semantic Segmentation Tool.[3]

Figure 29: Ground Truth of Frame 59 (left) and Frame 44 (right)

In a first experiment we fed Figure 28 into our prototype. The resulting segmentation can be seen in Figure 30.

Figure 30: Resulting segmentation of edited Frame 59 from prototype

4.1.3 Analysis of first real data result and input corrections

Upon reviewing the prototype's output in Figure 30, the reactions are mixed. First up, the detection of people is not very satisfying. The next thing that's noticeable is that the colors are spread across the entire picture, while in reality one would expect the entire left side to be only white or red. However, the right picture in Figure 30 shows green people on the left half of the picture and red people on the very right side which normally is supposed to be a safe zone for people, so people detected there should therefore appear green instead of red. One thing it does well however is the detection of the added warning line. Another relative good attribute of the output is that while it does falsely detect people close to the warning line, it at least colors them somewhat correctly as yellow.
An important observation to make for Figure 30 is that it detects more than one warning line. To be more precise, there appear to be fragments of a detected warning line both at the bottom left of the picture and the top left. If we look back at the edited Frame 59 (to the left of Figure 30), we notice that these are the places where the logos (RT and NRW police) are displayed. One theory we came up with when reviewing these results is that the existence of multiple such incorrectly detected warning

[3]Hitachi Semantic Segmentation is free to use and available on github: `https://github.com/Hitachi-Automotive-And-Industry-Lab/semantic-segmentation-editor`)

lines might be the cause for the wrong colors on both sides of the segmentation picture. Therefore, a second experiment was initiated, with the main differences being the reworked input picture.
For the second experiment we re-edited the set of frames again to now contain no logos. We also cut off the black edges on both sides of the frames. These re-edited frames were then once again fed into the prototype and Figure 31 summarizes the evolution of a frame. To sum up, the original frame coming from the video ("Koelner Schubser") gets edited first, so that it doesn't contain any logos or black edges any longer, and the new warning line gets added. The static pole on the top left also has to be cut off. After these adjustments are made, it can then be fed into the prototype. To evaluate the prototype's output, a ground truth is created manually for the specific frame.

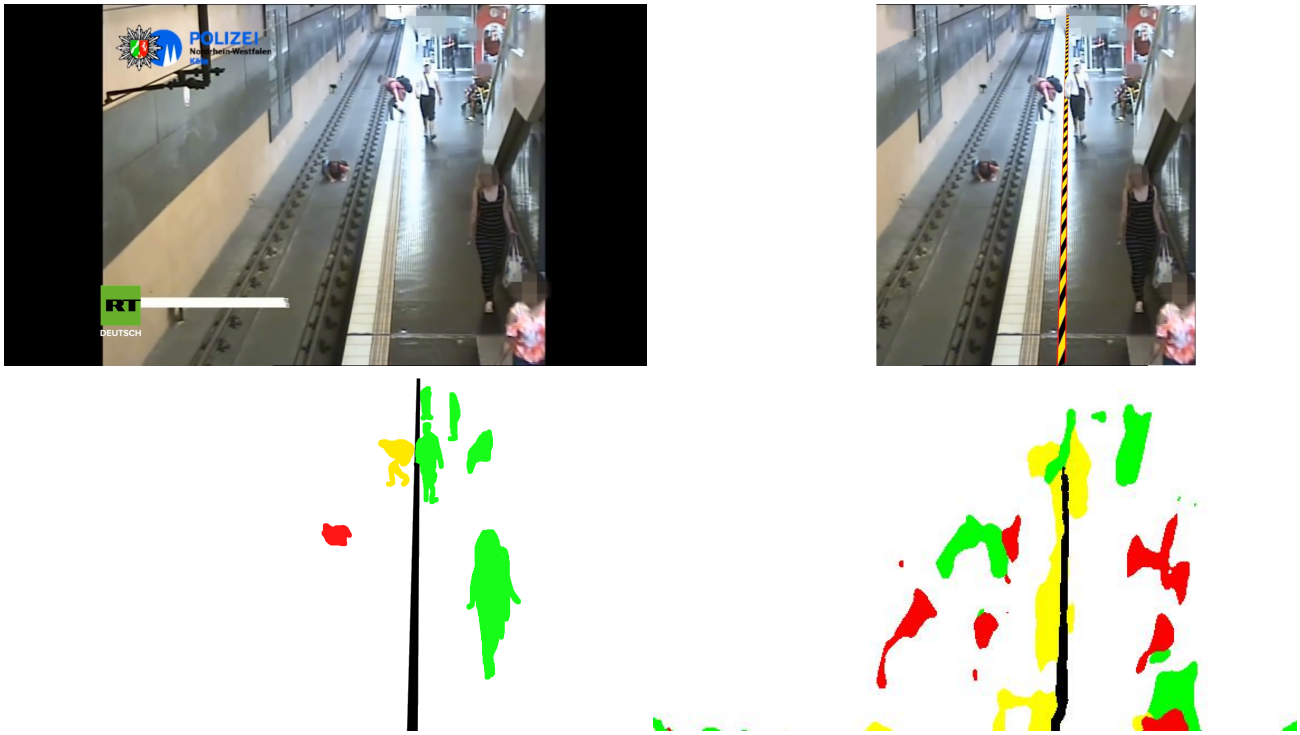

Figure 31: Evolution of Frame 59

By putting the most recent output on top of the most recently edited input picture, as shown in Figure 32, and putting it's transparency down, we can already see that the output now has been improved and it now better detects people. This improvement has been accomplished simply by cutting out things from the input we found out the algorithm can't handle. Despite the improvement made in detecting people, the categorizing (colors) is still done wrongly. The prototype was initially trained to mark people that are on tracks as red and people that are between tracks and the warning line as yellow. It most certainly failed to perform this task correctly. As for the reason, one might speculate that it did not recognize the tracks properly, as there are red fragments scattered across the tracks.
Overall, the most recent output in Figure 32 has still exceeded our expectations, given the fact that this is just the very first iteration of our prototype. As for some concrete numbers, the table below contains the accuracy, precision and recall numbers for the Frame 59 shown in Figure 32.

category	accuracy	precision	recall
white	0.88	0.97	0.91
black	1.0	0.86	0.63
green	0.93	0.042	0.038
red	0.96	0.0	0.0
yellow	0.97	0.12	0.81

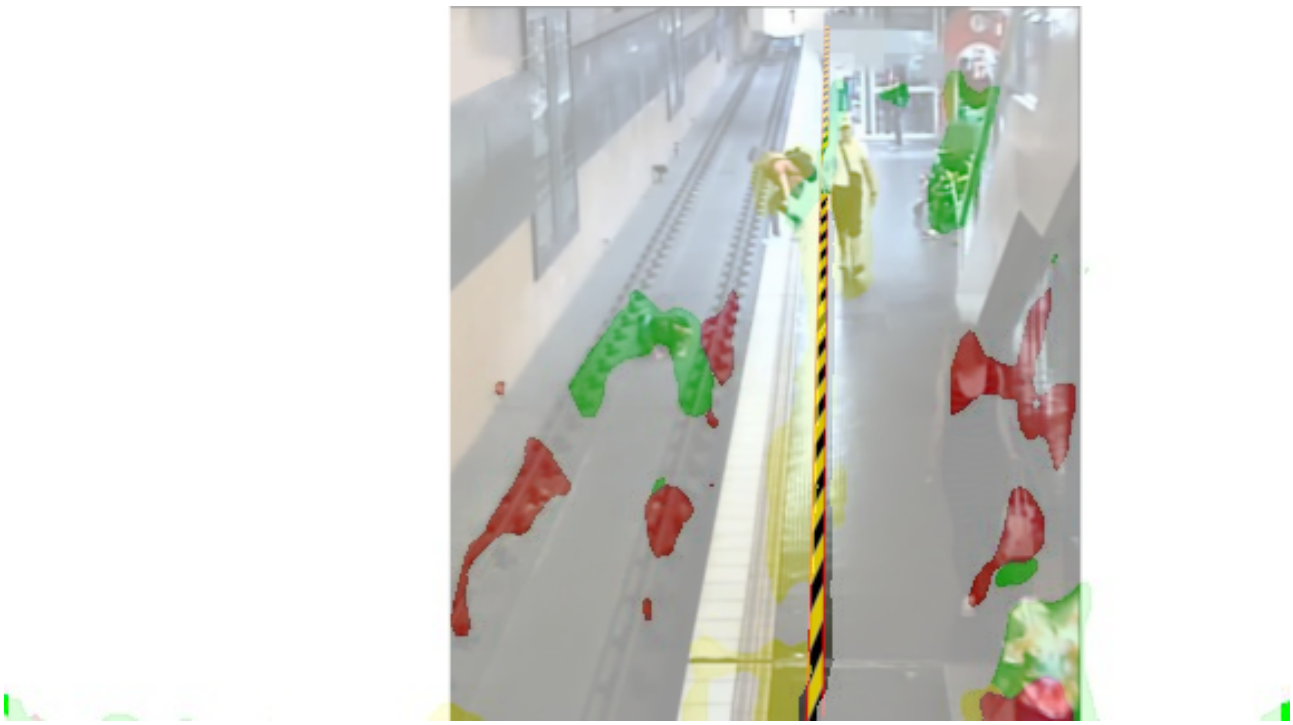

Figure 32: Comparing Frame 59 with output

4.2 Weaknesses of current implementation

We found that the current system can consistently recognize the safety line that was employed in order to increase the overall accuracy of the system. In real validation data, the tracks were not detected correctly, which results in red areas on both sides of the safety line. Furthermore, reflections off the ground and glass surfaces pose difficulties for the system. Another aspect that can disturb the segmentation are advertisements, info-screens and most likely a variety of trash.
The current prototype has only been trained with a very limited amount of unique human models. This could pose a problem for events in which humans dress up unusual (e.g. carnival) or in colors that make them harder to distinguish from something else on the platform. Certain rare poses - e.g. a person that is sleeping on the platform or a person that is falling mid-air - have not been included yet in the training and pose a threat.
Something that got noticed in the first experiments with real data static structures - such as poles or pillars - are being interpreted as (fragments of) warning lines quite often.

4.3 Recommendations for future implementations

All in all, the simulation environment is a good basis point to expand upon. Based on the critique formulated above, the simulation environment for the next development-cycle should include at least

- varying lighting conditions,
- reflective and opaque surfaces,
- more variety in station objects,
- more station types,
- more movable object variety - humans, trolleys, animals, etc.

- more poses of humans

A camera noise model should be used and implemented in the simulation. This adds more and realistic variance to the generated images.

Furthermore, single images will always pose the danger of occlusions and reflections, that can be misinterpreted on a frame-by-frame-basis. A system, that works on a series of images (=video-stream) instead of a single image, could potentially differentiate between objects and features, that are actually relevant and their nonrelevant counterparts like reflections or advertisement or even animals and other objects.

We think it would be very beneficial to expand the simulation environment to moving scenes. Even though this would require a lot of working hours, it would enable us to bring the most relevant objects into the scene: The subway trains. This could drastically change the implemented system, as now the position and speed of the train can be considered as well, when determining, whether a situation is dangerous or not. Furthermore, modelled humans on a simulated train station can move across the platform and interact with objects and each other.

Therefore, if a moving scene generator is implemented in the next cycle, this should be accompanied by a system, that tracks objects over time, instead of frame-by-frame. For example, existing systems for people-tracking could be used to predict their movements [39].

Another aspect, that could be considered, is to include real existing train stations in the simulation (replicas). This would have the benefit, that the training data is closer to reality. Another plus is, that footage from installed cameras at the corresponding train stations, can also be used for training a potential candidate. The simulation environment would still be very useful, as dangerous situations can be easily simulated, without bringing any human to harm.

5 Summary and Outlook

In this course, we followed the recipe provided to create the foundation for a subway station surveillance system. In the first cycle of development, we worked out the basic problems of the task and gave either full solutions, partial solutions or potential future solutions to address these problems.

In the User View, we first worked out the real world context for subway station security. From this, we formulated the task, that has to be executed and performance requirements that need to be fulfilled for a potential system, that can be used in practice.

In the Modeler View, we considered several existing approaches for a system, that can meet the specified requirements and narrowed down on a simulation based approach, as it would be difficult to get much data for dangerous situations in the real world, without unnecessarily putting humans in danger. From this point, we developed the concept for a prototype simulation environment, that can automatically create realistic scenes of populated subway stations. This includes both inanimate and living objects. Furthermore the types of availabe cameras were investigated, as these have to be part of the simulation as well. In the first development cycle, we decided to start with a simple classification of situations in 3 types: Not dangerous, potentially dangerous, definitely dangerous. A safety protocol, what happens if a system recognizes a situation as dangerous was worked out.

In the Implementation View the specifications from the Modeler View were implemented using Blender and Unity. For a start, we designed 10 different fixed station types and populated them

with inanimate objects. The humans were added according to the provided distributions, to create many varying realistic scenes. These scenes were used to create training data. For the detection of dangerous situations, we decided to use a Deep-Learning Segmentation-Approach.

The Validation View, during the projects development phase, we gave feedback to the simulation and implementation members, how the specified requirements can be reached. Furthermore, we analyzed the outputs of the provided system both on simulated validation data, as well as prepared data from real images. From this analysis we derived recommendations for future cycles of the development, to achieve the tight performance requirements of a security system.

All in all this project lays the ground work for the development of a robust system that in short term can increase security in subway train stations considerably. The core of the system is the simulation, which can provide cheap and abundant training data. However, in order to achieve good results in the real world, the simulation should not differ too much from reality, which it does at the moment. This is why we made the simulation environment be easily expandable to include more scenes and situations and generate more realistic images. As next major step in development, we think firstly the simulation has to be expanded in these ways. Furthermore, instead of generating only single-image scenes, animated sequences should be generated. This would enable the implementation and testing of more sophisticated people tracking systems based on video material.

References

[1] Andriluka et. al. People detection and tracking - max planck institut, 2009.

[2] Andriluka et. al. Pictorial structures revisited: People detection and articulated pose estimation, 2009.

[3] European Commission. Ethics guidelines for trustworthy ai. 2019.

[4] Deutsche Bahn. VideoÃ¼berwachung am bahnhof. 2019.

[5] Deutschlandfunk. Deutsche bahn camera. 2014.

[6] Image. Train station layout 1. 2020.

[7] Image. Train station layout 2. 2020.

[8] European Machine Vision Association. Emva standard 1288. 2016.

[9] Wikipedia contributors. Frankfurt u-bahn — Wikipedia, the free encyclopedia, 2020. [Online; accessed 15-August-2020].

[10] Wikipedia. Schienensuizid — wikipedia, die freie enzyklopÃ¤die, 2020. [Online; Stand 15. August 2020].

[11] asri. escalator, 2006. https://www.turbosquid.com/3d-models/3ds-escalator/312943 Accessed: 17.09.2020.

[12] NumikDigital. 3d model marina 1276, 2020. https://www.turbosquid.com/3d-models/3d-model-people-corona-character-1574786, Accessed: 17.09.2020.

[13] renderpeople. Dennis 004 standing business man free 3d model, 2020. https://www.cgtrader.com/free-3d-models/character/man/dennis-0263-standing-business-man, Accessed: 17.09.2020.

[14] renderpeople. Fabienne percy posed 001 3d model, 2020. https://free3d.com/3d-model/fabienne-percy-posed-001-977244.html, Accessed: 17.09.2020.

[15] renderpeople. Mei posed 001 3d model, 2020. https://free3d.com/3d-model/mei-posed-001-963144.html, Accessed: 17.09.2020.

[16] renderpeople. Nathan 003 animated walking casual free low-poly 3d model, 2006. https://www.turbosquid.com/3d-models/3ds-escalator/312943, Accessed: 17.09.2020.

[17] asri. escalator, 2019. https://www.cgtrader.com/free-3d-models/character/man/nathan-003-animated-walking-casual, Accessed: 17.09.2020.

[18] renderpeople. Sophia animated 001 idling 3d model, 2020. https://free3d.com/3d-model/sophia-animated-001-idling-130876.html, Accessed: 17.09.2020.

[19] RUSTtm88. 3d scan man 1, 2016. https://www.turbosquid.com/3d-models/water-park-slides-3d-max/1093267, Accessed: 17.09.2020.

[20] Komppis3D. Woman 3dscanpp1 free 3d model, 2018. https://www.turbosquid.com/3d-models/3dscanpp1-3d-model-1337614, Accessed: 17.09.2020.

[21] Gobotree. 3d free man jack a business man standing thinking model, 2019. https://www.turbosquid.com/3d-models/3d-gobotree-scanned-model-1372823, Accessed: 17.09.2020.

[22] hloman. black male 3dmodel free low-poly 3d model, 2015. https://www.cgtrader.com/free-3d-models/character/man/black-male-3dmodel, Accessed: 17.09.2020.

[23] renderpeople. Carla rigged 001 3d model, 2020. https://free3d.com/3d-model/carla-rigged-001-762221.html, Accessed: 17.09.2020.

[24] hloman. black male 3dmodel free low-poly 3d model, 2015. https://www.cgtrader.com/free-3d-models/character/man/black-male-3dmodel, Accessed: 17.09.2020.

[25] renderpeople. Claudia rigged 002 3d model, 2020. https://free3d.com/3d-model/claudia-rigged-002-167206.html, Accessed: 17.09.2020.

[26] renderpeople. Eric rigged 001 3d model, 2020. https://free3d.com/3d-model/eric-rigged-001-771956.html, Accessed: 17.09.2020.

[27] 3dcap. Casual woman walking free low-poly 3d model, 2016. https://www.cgtrader.com/free-3d-models/character/woman/business, Accessed: 17.09.2020.

[28] NumikDigitall. Woman jess casual walking 001 free 3d model, 2019. https://www.cgtrader.com/free-3d-models/character/woman/woman-jess-casual-walking-001, Accessed: 17.09.2020.

[29] NumikDigital. Ivan 1304 free 3d model, 2020. https://www.cgtrader.com/free-3d-models/character/man/ivan-1304, Accessed: 17.09.2020.

[30] doschdesign. Dosch 3d people business sample free 3d model, 2016. https://www.cgtrader.com/free-3d-models/character/other/dosch-3d-people-business-sample-b6617df1ad6a6c8af819f7ab83954195, Accessed: 17.09.2020.

[31] humano3d. Humano casual woman sitting and talking 0218 free 3d modelhumano casual woman sitting and talking 0218 free 3d model, 2019. https://www.cgtrader.com/free-3d-models/character/woman/humano-casual-woman-sitting-and-talking-0218, Accessed: 17.09.2020.

[32] Pixabay. https://www.pixabay.com Accessed: 17.9.2020.

[33] Takashi Shinzato. Box muller method. *Hitotsubashi University, Tokyo*, 2007.

[34] Alan Zucconi. How to generate gaussian distributed numbers, June 2016. Accessed: 10.09.2020.

[35] Nikos Paragios, Visvanathan Ramesh, Bjoern Stenger, and Frans Coetzee. Real-time crowd density estimation from video, November 21 2006. US Patent 7,139,409.

[36] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[37] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[38] Torch Contributors. Crossentropyloss, 2019. Accsessed: 2020-09-08.

[39] People tracking using deep learning, February 2019. Accessed: 10.09.2020.