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

Subway Station Hazard Detection

Project Presentation
Module: Systems Engineering Meets Life Sciences II SoSe 2020
Supervisors: Prof. V. Ramesh, Dr. M. Rammensee

Overview

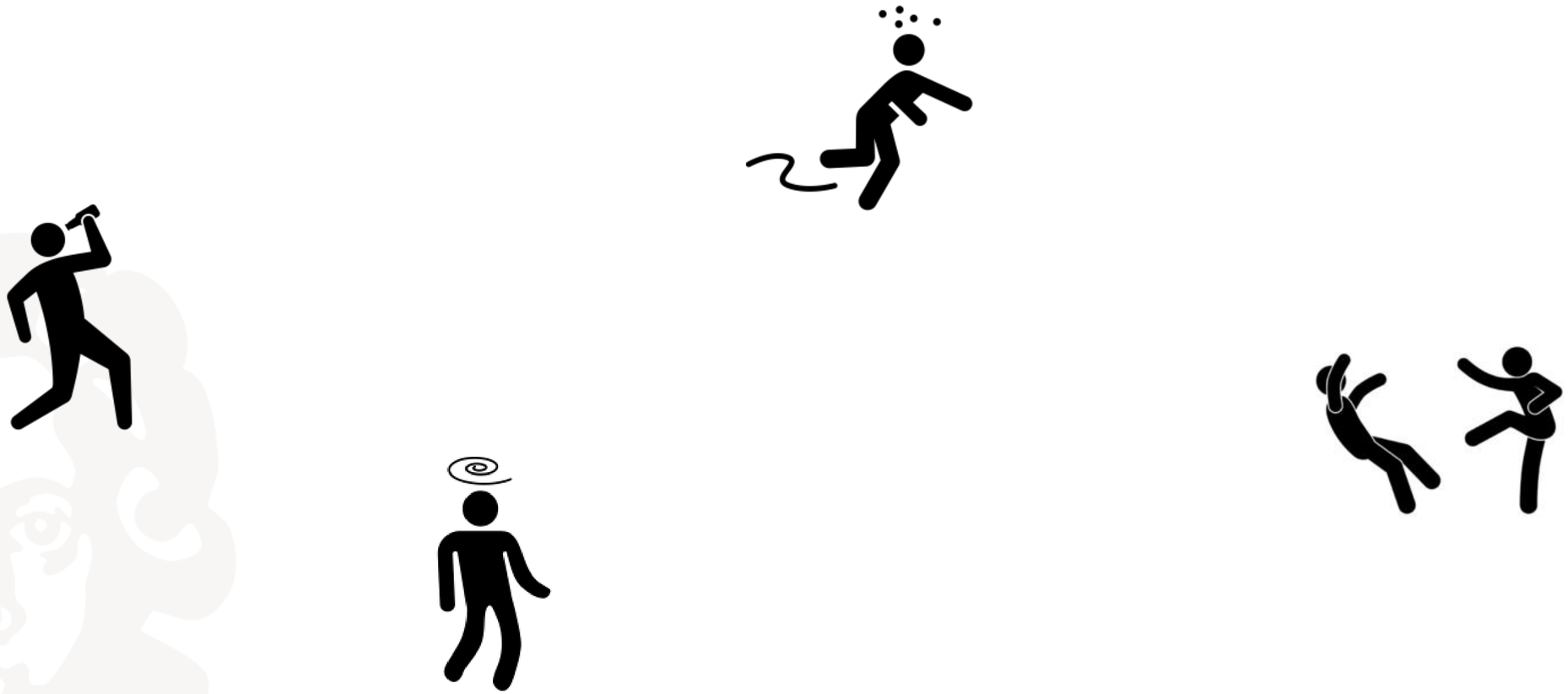
- User View (Edis)
- Modeler View (Felix)
- Implementation View (Alen & Pascal)
- Validation View (Edis & Martin)
- Summary & Outlook

User View

Edis Kurtanovic

Context

Context



Context

HORROR-TAT AM FRANKFURTER HAUPTBAHNHOF

Junge (8) vor einfahrenden ICE gestoßen! Tot!

Mann (40) afrikanischer Herkunft festgenommen ++ Er stieß Mutter (40) und Sohn vor Zug ++ Frau konnte sich noch retten, brach dann zusammen ++ Täter wollte dritte Person ins Gleis werfen

Quelle: BILD 0:55 Min.

Mann stößt Achtjährigen und Mutter vor Zug – Kind stirbt

Veröffentlicht am 29.07.2019 | Lesedauer: 3 Minuten

Am Frankfurter Hauptbahnhof sind ein Kind und seine Mutter vor einen einfahrenden ICE gestoßen worden.

- Der Achtjährige starb noch an der Unfallstelle. Die Mutter konnte sich auf einen Fußweg zwischen den Gleisen retten.
- Ein 40-Jähriger wurde festgenommen. Er wird einem Haftrichter vorgeführt, zum Motiv war zunächst nichts bekannt.

Ein achtjähriger Junge und seine Mutter sind am Montag im Frankfurter Hauptbahnhof vor einen einfahrenden Zug gestoßen worden. Das teilte die Polizei in Frankfurt mit. Die Mutter konnte sich demnach gerade noch auf einen Fußweg zwischen den Gleisen retten, das Kind wurde allerdings vom Zug erfasst und starb noch an der Unfallstelle. Ein Tatverdächtiger wurde festgenommen.

Am Hauptbahnhof kam es laut Einsatzkräften zu einem „massiven Polizeieinsatz“, 16 Rettungswagen und ein Hubschrauber waren vor Ort. Gegen den Mann ermittelt nun eine Mordkommission. Die Staatsanwaltschaft Frankfurt wertet zudem die Videoaufnahmen aus.

Die Mutter des Achtjährigen wurde in ein Krankenhaus gebracht und notfallmedizinisch versorgt. Sie solle so bald wie möglich befragt werden, sagte ein Polizeisprecher.

Context

Some numbers and statistics for Germany

- ~151 deaths per year (accidents)
- Over 2600 deaths per year on track (suicide)
- 8% of all trains do not arrive on time
  - → ~1% of all trains are caught up in accidents with humans
- Traffic accidents caused by humans and objects cost over 30 billion €
- Currently >35.000 km track length across Germany

Context

Task

- **Subway Hazard Detection System**
- proactive use of existing camera structure
- potential danger & absolute danger

  too close ↕ on tracks ↕

- proper reaction
- rough location

Performance Requirements

- Real-time end-to-end
- instant transmission & analysis
- ↑ detect dangerous situations if they exist
- ↓ low amount of false alarm

Modeler View

Felix Hoffmann

Overview (Modeler View)

1. Use available cameras inside train station to take images in regular intervals

2. Feed images to Neural Network that is trained to identify locations of humans

3. Use knowledge about position of humans to infer a signal that is sent to conductor of next incoming train

4. Signal $\in$ {no action required, slow down, emergency braking}

5. Train conductor can overrule this decision (our system is only a recommendation system!)

Sensors

Sources: https://www.deutschlandfunk.de/deutsche-bahn-mehr-kameras-an-bahnhoefen.2852.de.html?dram:article_id=306908
https://www.welt.de/politik/deutschland/article112115693/Kameras-in-allen-Bahnhoefen-kosten-Milliarden.html

Train station structure – One Track

Source: https://commons.wikimedia.org/wiki/File:U-Bahn-Station_Dom-Römer_(Frankfurt).jpg

Train station structure – Two Tracks

Source: https://cdn.muenchen-p.de/.imaging/stk/responsive/image980/dms/shutterstock-2019/verkehr/u-bahn-josephsburg-hp/document/u-bahn-josephsburg-hp.jpg

Objects inside train station

We differentiate between **fixed** and **temporary** objects.

Fixed objects:

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

Temporary objects:

- Humans
- Animals
- Objects carried by humans
  - Wheelchair / Walking Aid
  - Backpack
  - Suitcase
  - Smartphone
  - Bike
  - Clothes
  - …

Safety-distance line

Source: https://medium.com/@garethdennis/finding-ways-to-reduce-risk-at-the-platform-train-interface-1d509196ed

Deep Learning Model

- Trained on graphical simulations
- Model can differentiate between fixed and temporary objects
- Automatically detects people in images and their positioning relative to the safety-distances lines
- Inference of a danger level that enables a signal $\in$ {no action required, slow down, emergency braking}
- More features in future implementations

Required Features

With a more advanced NN we can take more variables into account when determining a value for our abstract concept of a danger level:

- Temporary objects
  - Wheelchairs
  - Smartphones
  - Suitcases
- Animals
  - Especially pets like dogs
- Tracking movement over time

Required Features

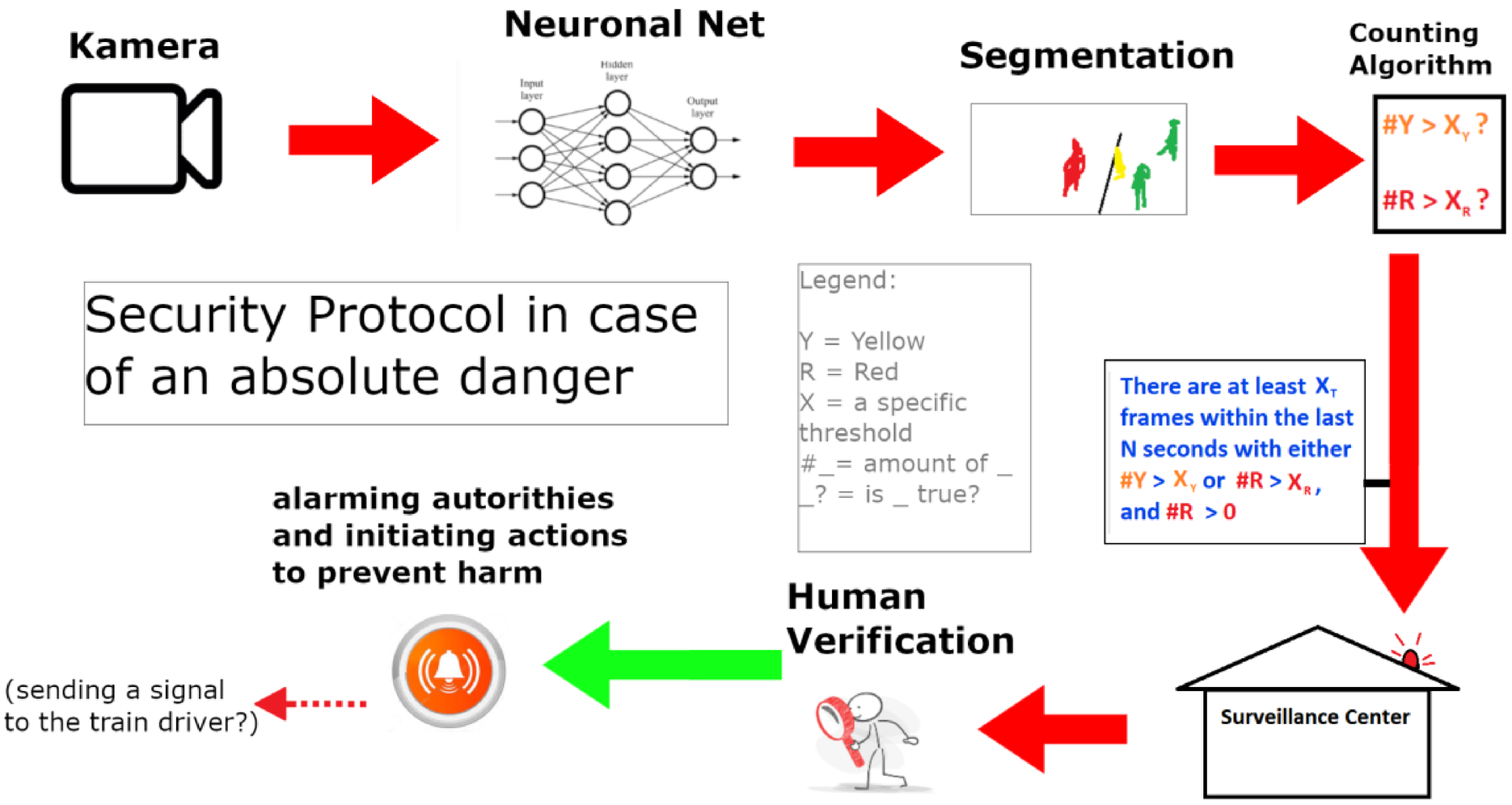
- Training model on more real-world data / more accurate simulations

- Increasing robustness to variance
  - Lighting conditions (influence noise levels in camera images)
  - Reasoning (e.g. human shown on display attached to the wall is not a real person)
  - Reflections (e.g. object duplicated in mirror / glass wall)
  - Occlusion (e.g. human (partially) hidden by another object)

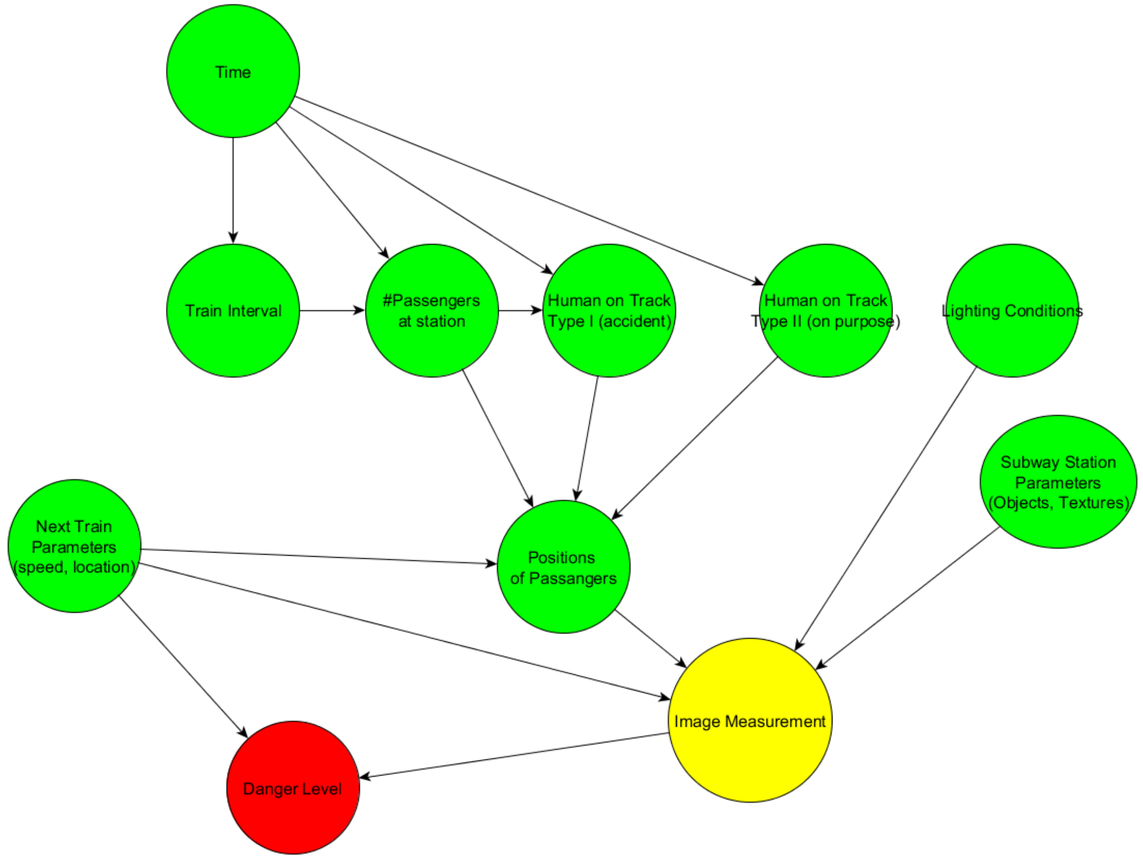
- Explainability

- Certification

Security Protocol



Probabilistic Graphical Model



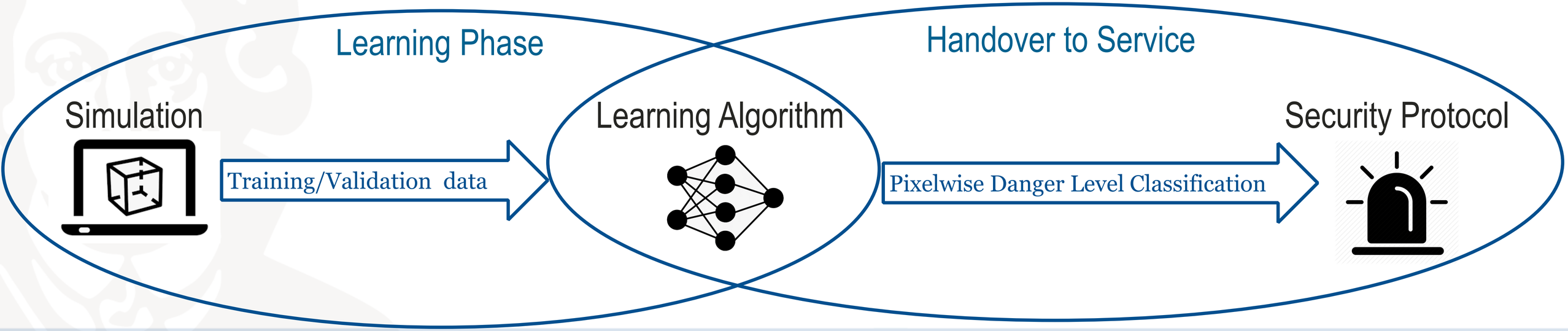
- The distributions can be estimated from real data
- Core variables: Humans on Track (two types)
- For the first dataset created from simulation a simplified model with hand-estimated distributions was used.

System Implementation View

Pascal Fischer & Alen Smajic

System Modules

- The simulation, which can generate various realistic scenarios of a subway station
- The learning algorithm, 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



Implementation Tasks

Implementation of the first prototype

1. Development and procurement of 3D station assets

External Resources:

cgtrader

Free3D

2. Implementation of the simulation

3. Integration of the „distribution of characters“ inside the simulation

unity

4. Development of the learning algorithm for pixelwise classification

PyTorch

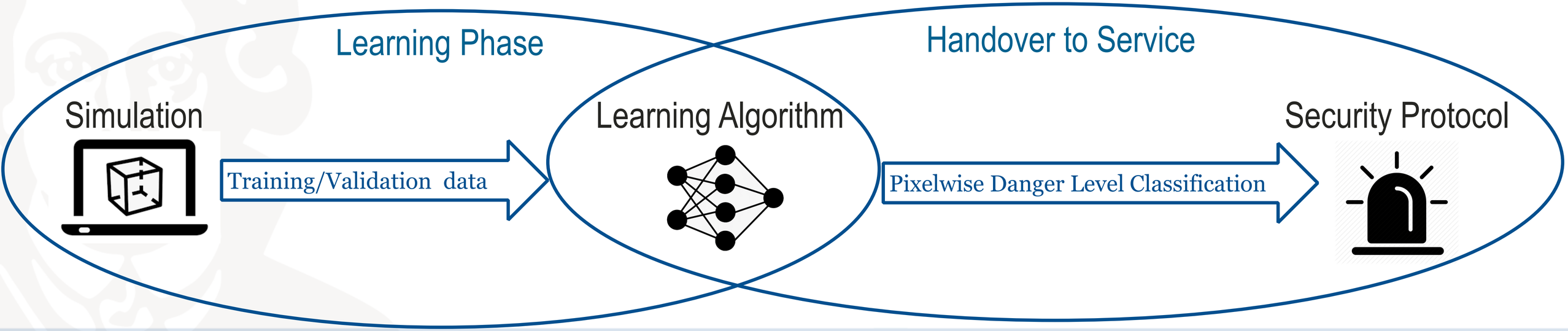
python

External Resources:

5. Implementation of the security protocol

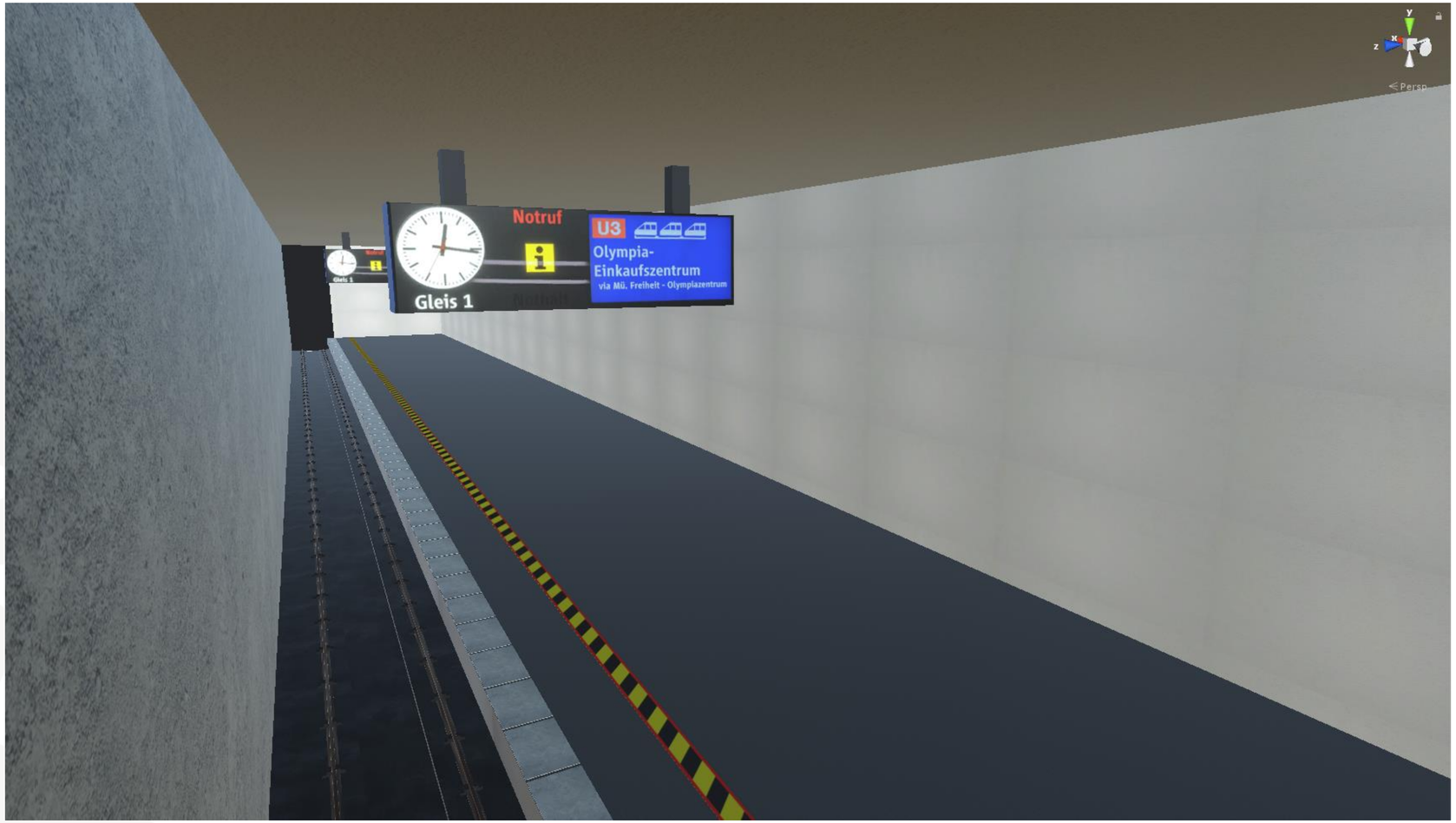
The Dependency of the System Modules

- Deep learning approach to classify the danger level (semantic segmentation)
- We use semantic segmentation to recognize and classify people into one of the 3 categories: green, yellow or red
- Depending on the classification, a corresponding security protocol decides how to deal with the respective situation and triggers a corresponding security process



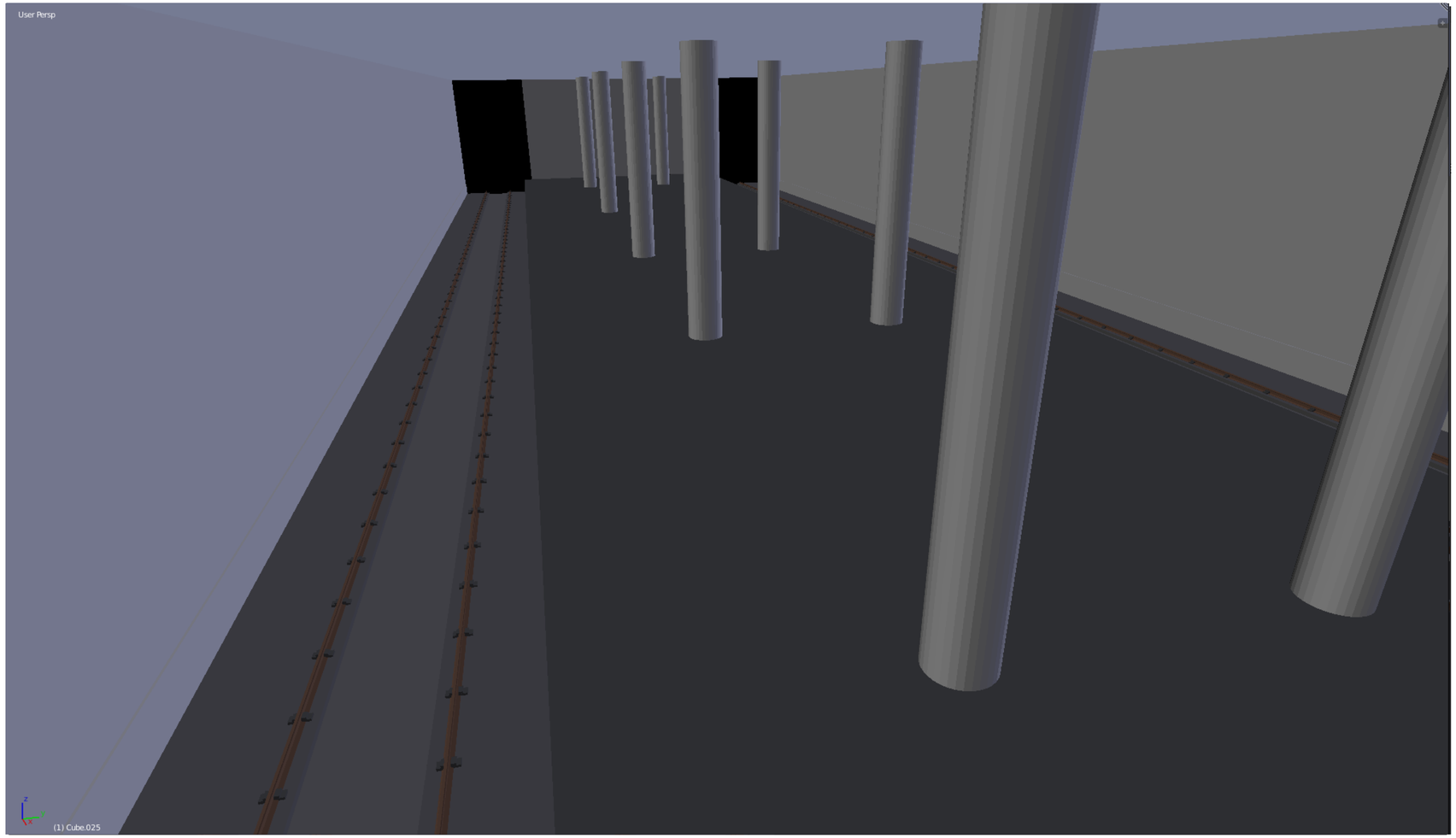
Danger Level Overview

- Floor marking on each subway platform (1.5m from the edge of the platform)



1. Development of 3D Station Assets

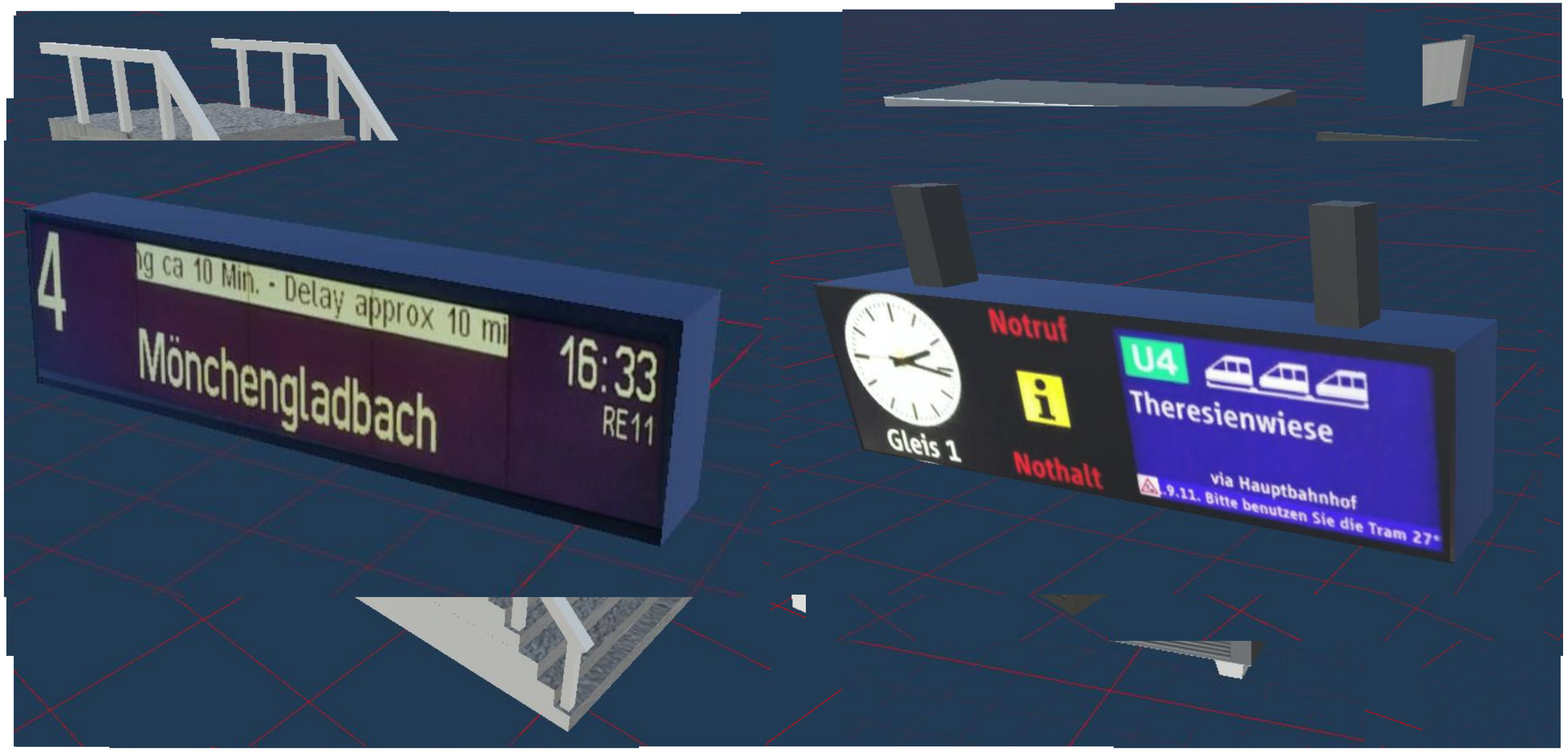
- **<u>Station models</u>**
- Station objects
- Human models
- Textures



10 different station types

1. Development of 3D Station Assets

- Station models
- **<u>Station objects</u>**
- Human models
- Textures



Rubbish bins, chairs and benches, Elevators, Escalators,
Snack machines, stairs, Entrances, Departure boards

1. Development of 3D Station Assets

- Station models
- Station objects
- **Human models**
- Textures

25 different characters

1. Development of 3D Station Assets

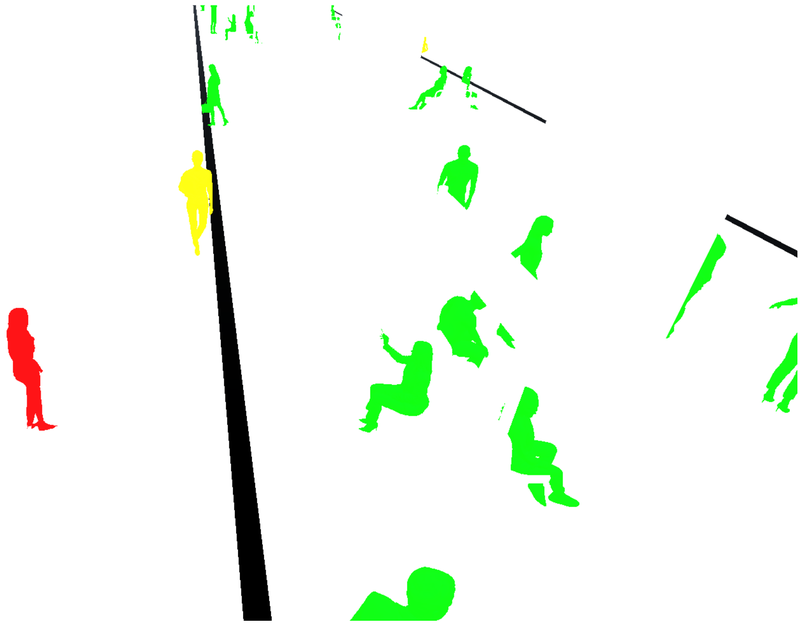
- Station models
- Station objects
- Human models
- **<u>Textures</u>**

119 different texture packs

2. Subway Station Simulator

- Generates realistic station scenarios and the corresponding ground truth images

- These images are used as training, test and validation data for the learning algorithm



- Semantic segmentation to classify every pixel into one of 5 classes: white, black, green, yellow or red

2. Subway Station Simulator

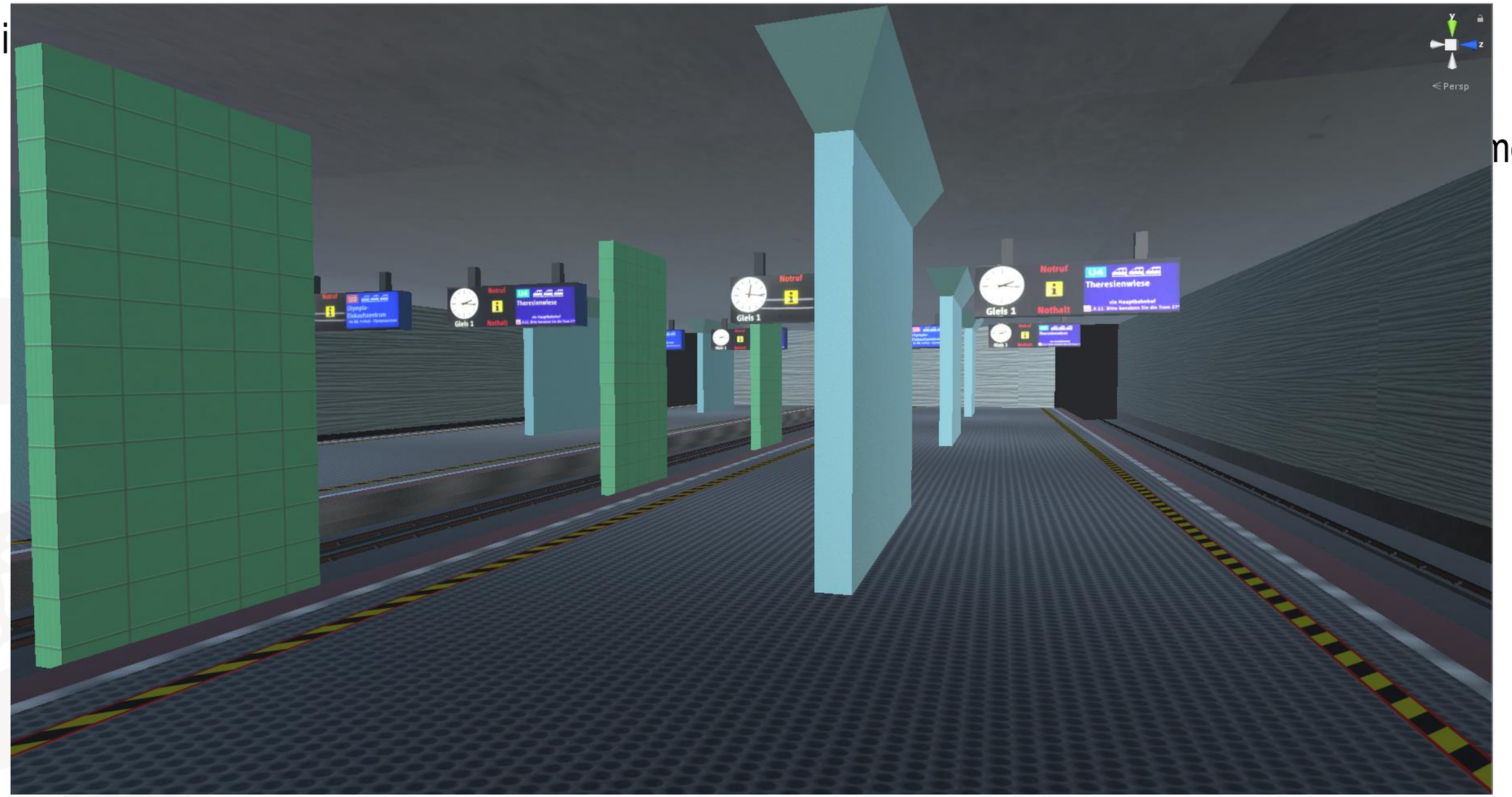
Simulator Specifications:

- Constant lighting without any shadows or reflections
- Camera recording without any occurence of noise
- Recordings from both endpoints of a station
- Cameras are positioned on the ceiling at an angle of 30° and cover the space between the tracks and the station platform
- Images are recorded in 1080p resolution

2. Subway Station Simulator

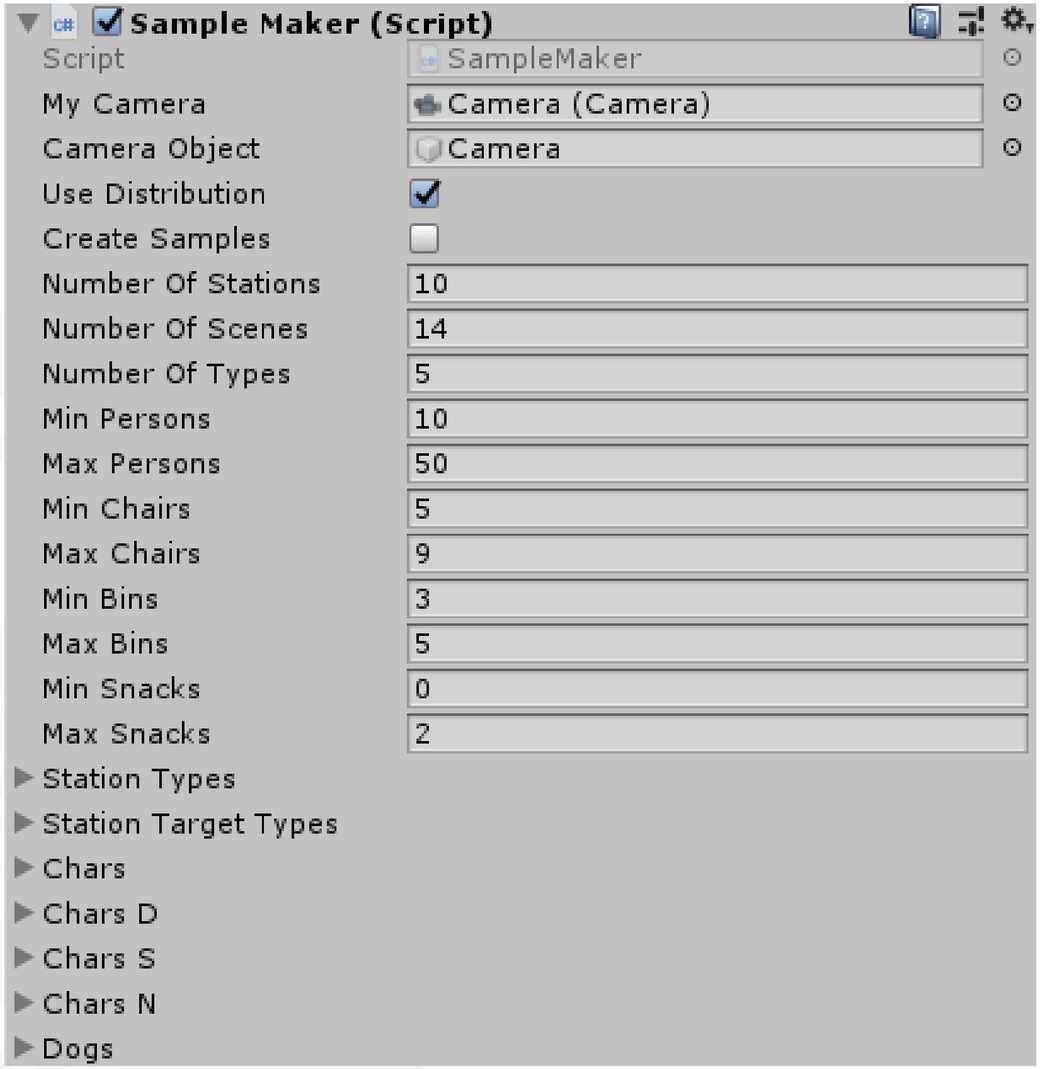
Si

- ments



2. Subway Station Simulator

Simulator Configurations:



Generating random normal distributed numbers

- Unity, C# supports only uniform distributed numbers
- Box Muller method to slow
- Using slightly different method: Marsaglia polar method

Generating random normal distributed numbers

Step1: Generating $Z \sim \mathcal{N}(0,1)$

Algorithm 1 MarsagliaPolarMethod()

Ensure: $Z \sim \mathcal{N}(0,1)$

repeat

$U_1 = \text{Random.Range}(0,1)$ -1

$U_2 = \text{Random.Range}(0,1)$ -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

Generating random normal distributed numbers

Step2: Transform $Z \sim \mathcal{N}(0,1) \longrightarrow Z \sim \mathcal{N}(\mu, \sigma^2)$

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 $\mu + z \cdot \sigma$

Generating random normal distributed numbers

Step2: Prevent random numbers outside a given range [min, max]

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
    $Z = \text{MarsagliaPolarMethod}()$
    $Z = \text{Gaussian}(Z, \mu, \sigma)$
  until min$\leq Z \leq$max
  return Z

Character Distribution - Pascal

- Number of characters in a scene = Uniform({0, 1 , 2, … , 79, 80, 100, 200, 300})
- Place every character on the platform in general
- Add additional character on the tracks with probability 1/200
- Required distribution: Two different types of characters on the tracks
- Probability type 1 (accident) = 1/309600, and type 2 (on purpose) = 1/3600
- To obtain correlation with people on the tracks: Probability character on the tracks: 1/10000

Character Distribution

Three states for characters on the platform

- Sitting
- Standing
- Walking

We don‘t make a difference between walking and standing

- Probability standing = 2/3
- Probability sitting = 1/3

If is no seat available then every character will be standing

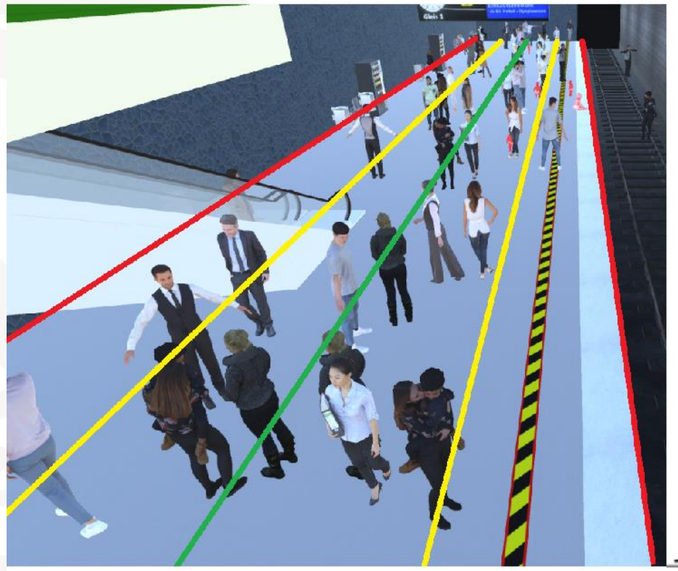
Character Distribution

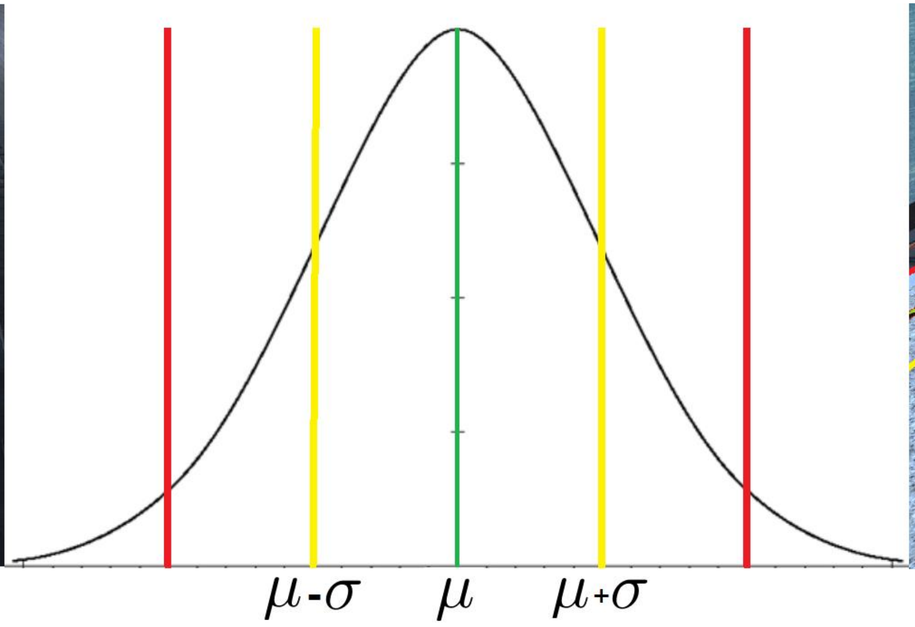
- y-rotation angle = Uniform({1, 2, … , 360})
- x-rotation angle = 0 (90 degrees to the platform)

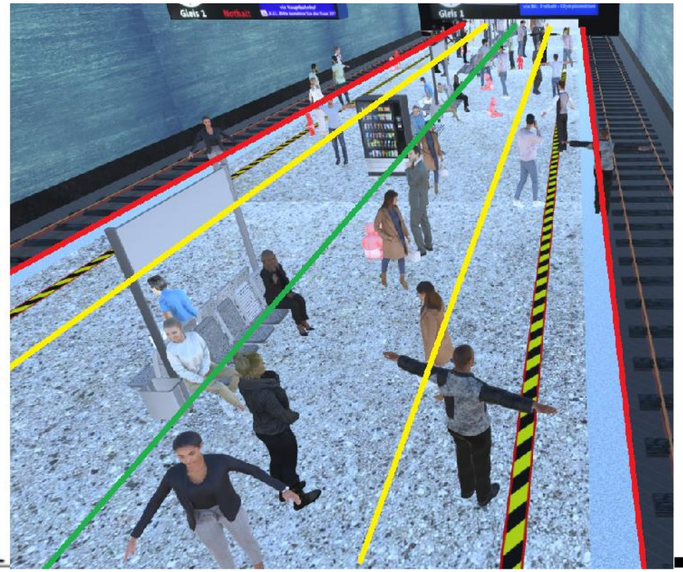
Character Distribution

x-position on platform = RandomNormal(center, width/4, end1, end2)

- center = x coordinate of platforms center
- width = platform length on x-axis
- end1 = center – width
- end2 = center + width







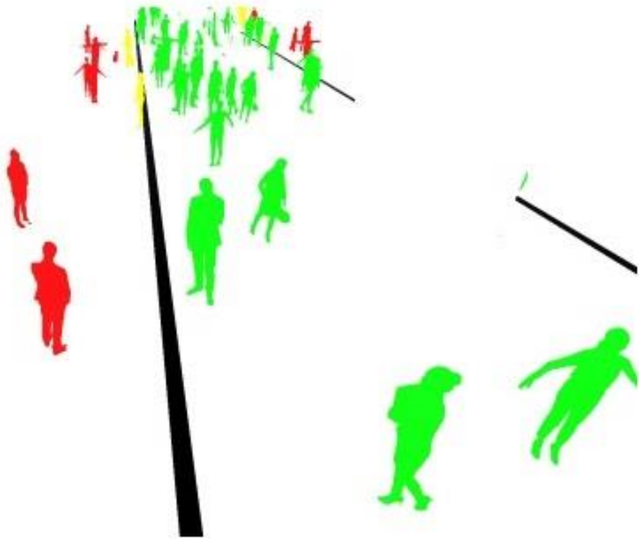
Character Distribution

- y-position on platform/tracks = Uniform({0, … , platform_length})
- X-position on tracks = Uniform({0, …, platform_width})
- y-position and rotation for character on a seat is uniform distributed over all seats

Subway Station Segmentation

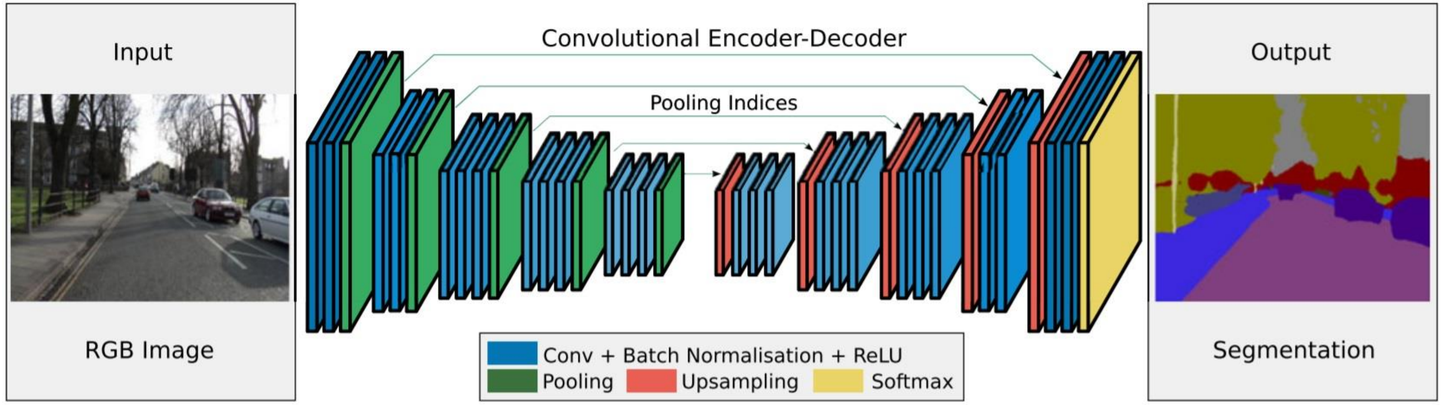
Classify each pixel of an image to one of the 5 classes

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



Subway Station Segmentation

Network Architecture



Layer		Filters	Size	Input	Output	Layer		Filters	Size	Input	Output
1	enc	64	3×3	$320 \times 240 \times 3/1$	$320 \times 240 \times 64$	19	up		2×2	$10 \times 8 \times 512$	$20 \times 15 \times 512$
2	enc	64	3×3	$320 \times 240 \times 64$	$320 \times 240 \times 64$	20	dec	512	3×3	$20 \times 15 \times 512$	$20 \times 15 \times 512$
3	max		2×2	$320 \times 240 \times 64$	$160 \times 120 \times 64$	21	dec	512	3×3	$20 \times 15 \times 512$	$20 \times 15 \times 512$
4	enc	128	3×3	$160 \times 120 \times 64$	$160 \times 120 \times 128$	22	dec	512	3×3	$20 \times 15 \times 512$	$20 \times 15 \times 512$
5	enc	128	3×3	$160 \times 120 \times 128$	$160 \times 120 \times 128$	23	up		2×2	$20 \times 15 \times 512$	$40 \times 30 \times 512$
6	max		2×2	$160 \times 120 \times 128$	$80 \times 60 \times 128$	24	dec	512	3×3	$40 \times 30 \times 512$	$40 \times 30 \times 512$
7	enc	256	3×3	$80 \times 60 \times 128$	$80 \times 60 \times 256$	25	dec	512	3×3	$40 \times 30 \times 512$	$40 \times 30 \times 512$
8	enc	256	3×3	$80 \times 60 \times 256$	$80 \times 60 \times 256$	26	dec	256	3×3	$40 \times 30 \times 512$	$40 \times 30 \times 256$
9	enc	256	3×3	$80 \times 60 \times 256$	$80 \times 60 \times 256$	27	up		2×2	$40 \times 30 \times 256$	$80 \times 60 \times 256$
10	max		2×2	$80 \times 60 \times 256$	$40 \times 30 \times 256$	28	dec	256	3×3	$80 \times 60 \times 256$	$80 \times 60 \times 256$
11	enc	512	3×3	$40 \times 30 \times 256$	$40 \times 30 \times 512$	29	dec	256	3×3	$80 \times 60 \times 256$	$80 \times 60 \times 256$
12	enc	512	3×3	$40 \times 30 \times 512$	$40 \times 30 \times 512$	30	dec	128	3×3	$80 \times 60 \times 256$	$80 \times 60 \times 128$
13	enc	512	3×3	$40 \times 30 \times 512$	$40 \times 30 \times 512$	31	up		2×2	$80 \times 60 \times 512$	$160 \times 120 \times 128$
14	max		2×2	$40 \times 30 \times 512$	$20 \times 15 \times 512$	32	dec	128	3×3	$160 \times 120 \times 128$	$160 \times 120 \times 128$
15	enc	512	3×3	$20 \times 15 \times 512$	$20 \times 15 \times 512$	33	dec	64	3×3	$160 \times 120 \times 128$	$160 \times 120 \times 64$
16	enc	512	3×3	$20 \times 15 \times 512$	$20 \times 15 \times 512$	34	up		2×2	$160 \times 120 \times 64$	$320 \times 240 \times 64$
17	enc	512	3×3	$20 \times 15 \times 512$	$20 \times 15 \times 512$	35	dec	64	3×3	$320 \times 240 \times 64$	$320 \times 240 \times 64$
18	max		2×2	$20 \times 15 \times 512$	$10 \times 8 \times 512$	36	dec	2	3×3	$320 \times 240 \times 64$	$320 \times 240 \times 2$

Subway Station Segmentation

Loss Function: Categorical Weighted Cross Entropie Loss

$$\text{loss}(x, class) = \text{weight}[class]\left(-\log\left(\frac{\exp(x[class])}{\sum\limits_{c \in Classes} \exp(x[c])}\right)\right)$$

$$= \text{weight}[class]\left(-x[class] + \log\left(\sum_{c \in \text{Classes}} \exp(x[c])\right)\right)$$

- Classes = {white, black, green, red, yellow}
- Class ∈ Classes is the correct class for x

Subway Station Segmentation

Loss Function: Weights

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

- $\text{pixel}(class) = |\{p \in img \mid \text{pred}(p) = class \mid img \in T\}|$
- Multiply weight for white class with 10 to get a higher precision

Subway Station Segmentation

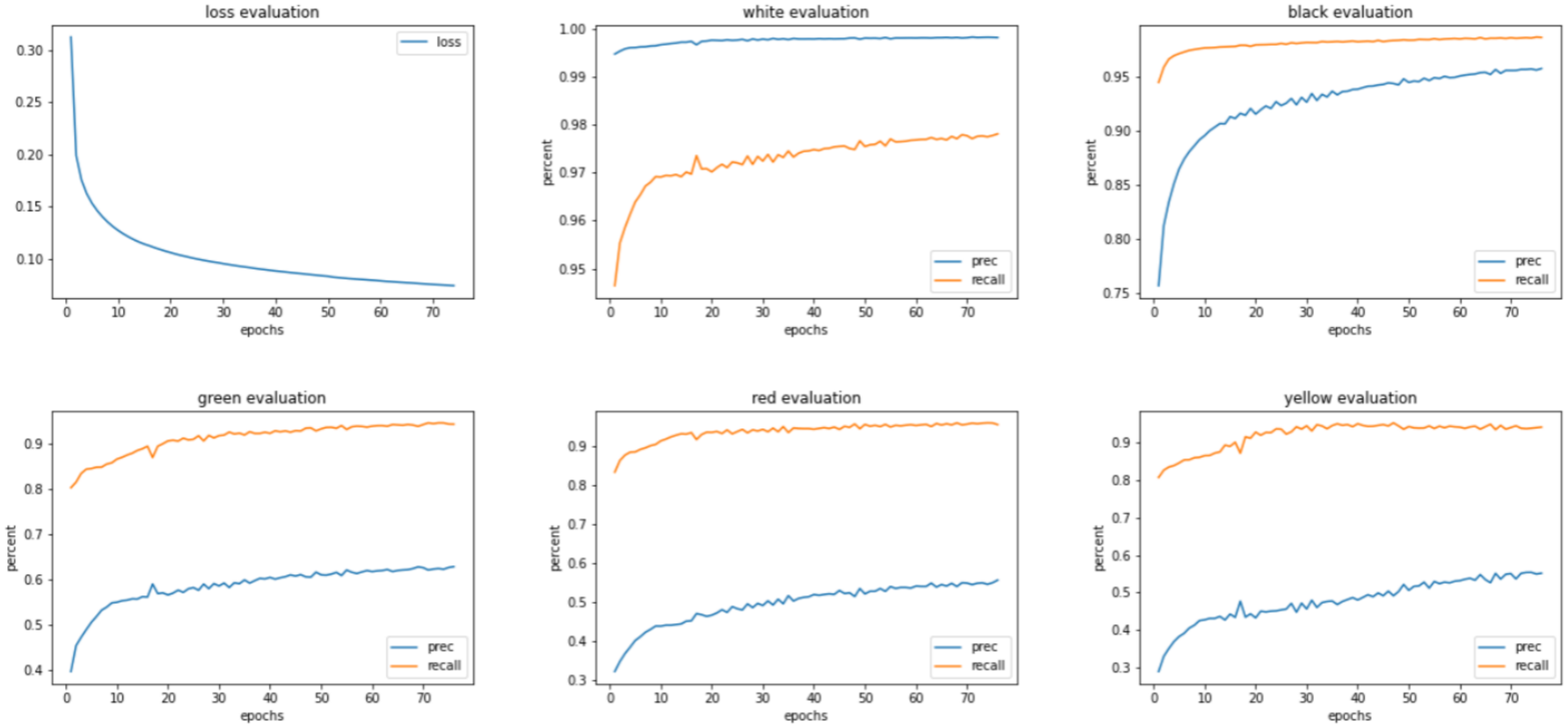
Training

- Training (50.000 imgs) and validation (10.000 imgs) dataset
- 75 Epochs of training on training dataset
- Nvidia Volta GPU from AI Lab
- Stochastic Gradient Desent

Subway Station Segmentation

Results: Evaluation

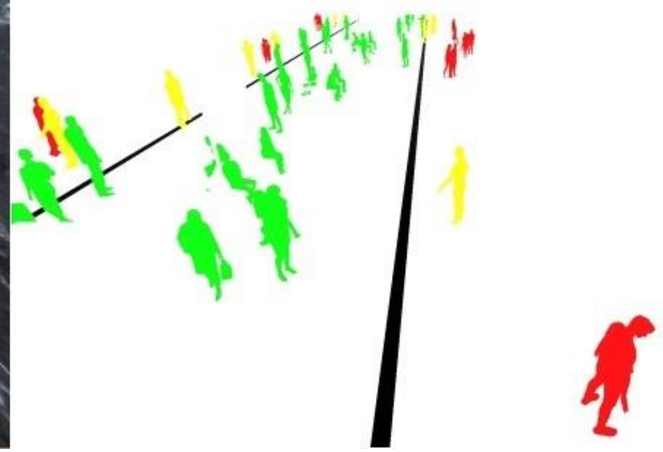
class	precision	recall
white	100%	98%
black	95%	99%
green	62%	93%
red	54%	96%
yellow	54%	94%



Subway Station Segmentation

Results: Example One

Image



Ground Truth

Prediction

Subway Station Segmentation

Results: Example Two

Image

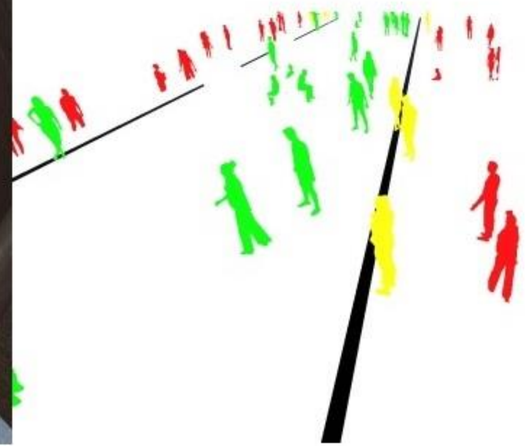
Ground Truth

Prediction

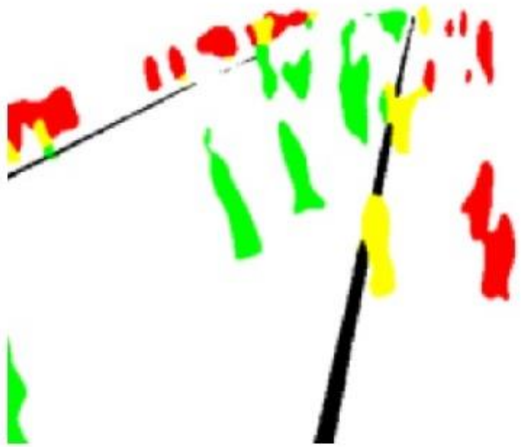
Subway Station Segmentation

Results: Example Three

Image



Ground Truth



Prediction

Thresholding Algorithm

Input: Segmentation output for both Images, Threshold

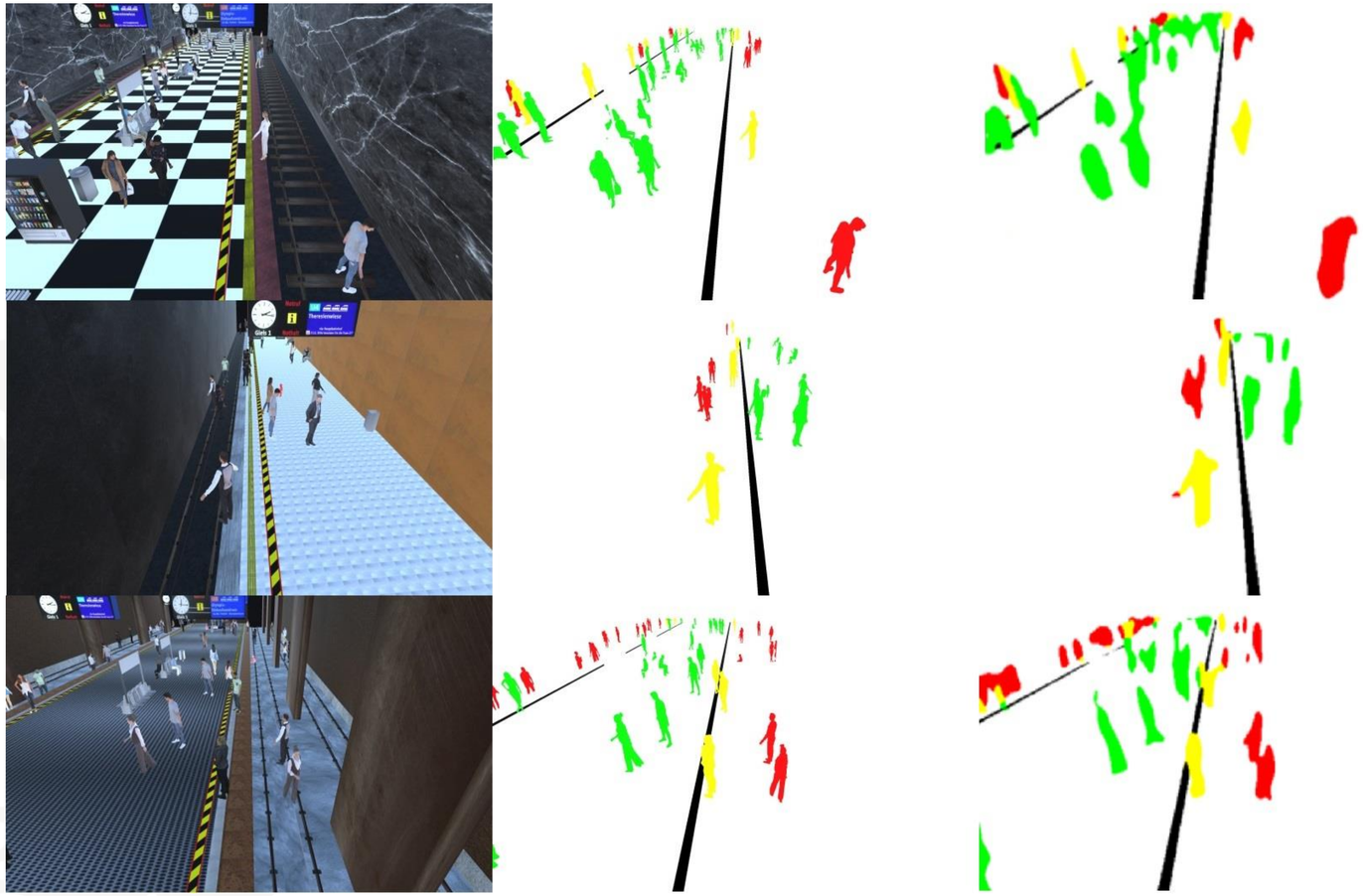
→ Count number of red (yellow) pixels in predicted output

→ If the number is greater than the specified threshold,
a situation is classified as dangerous (potentially dangerous)

Validation View

Martin Ludwig, Edis Kurtanovic

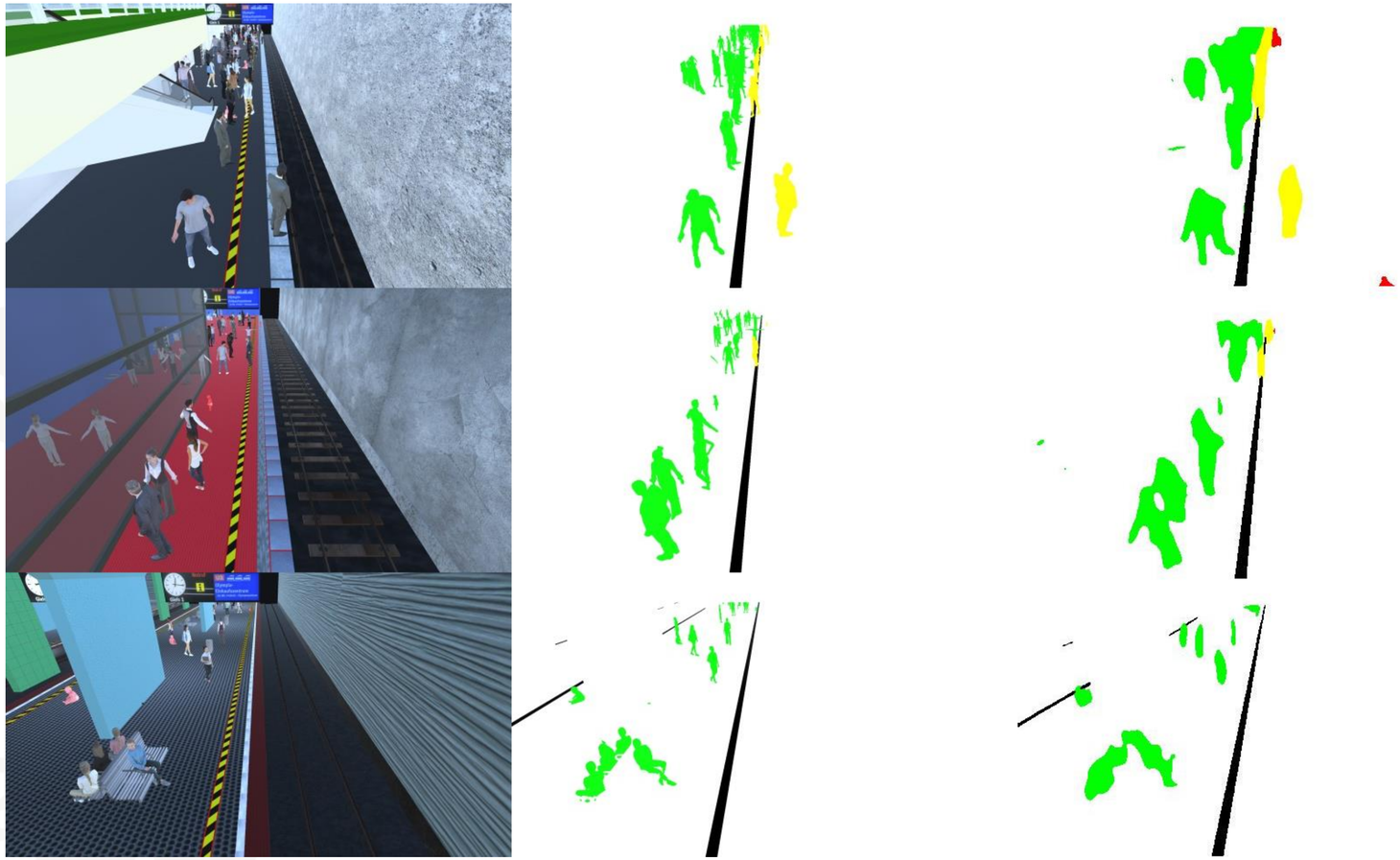
Segmentation Output Examples I



Input Ground Truth Output

- Samples for Validation set are drawn from same distribution as training set.
- ≅ 43% of samples show red in the ground truth (much higher, than in reality)
- Segmentation output is in general quite good, but blurry edges

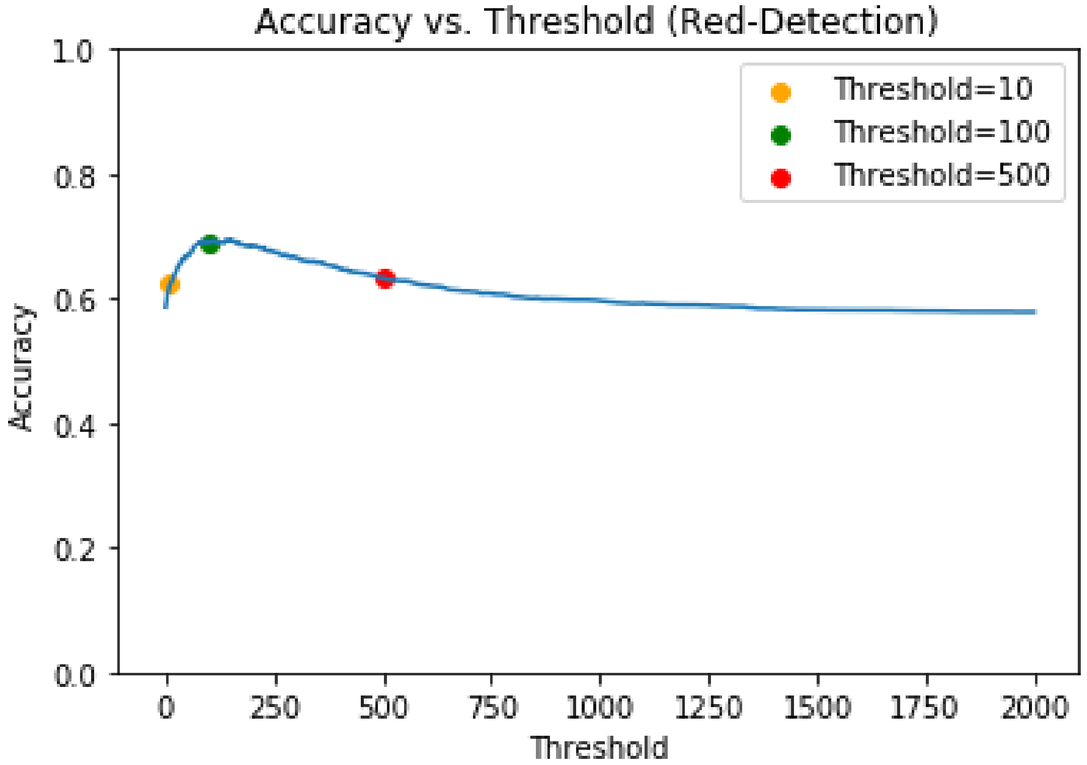
Segmentation output examples II

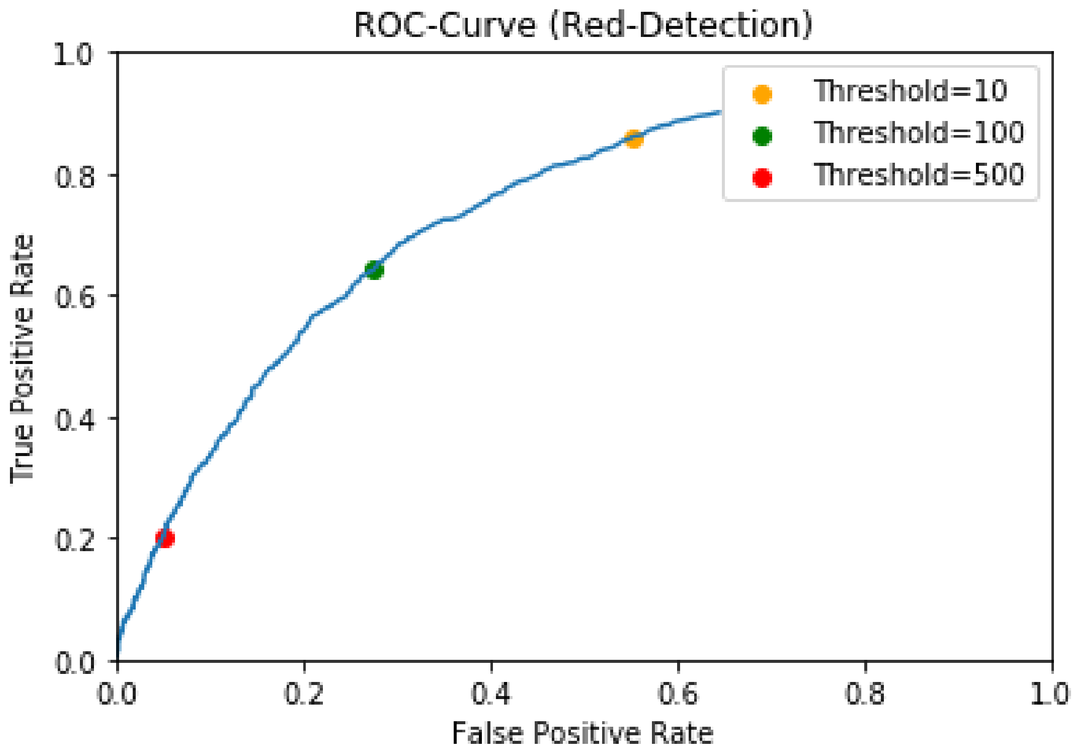


Input Ground Truth Output

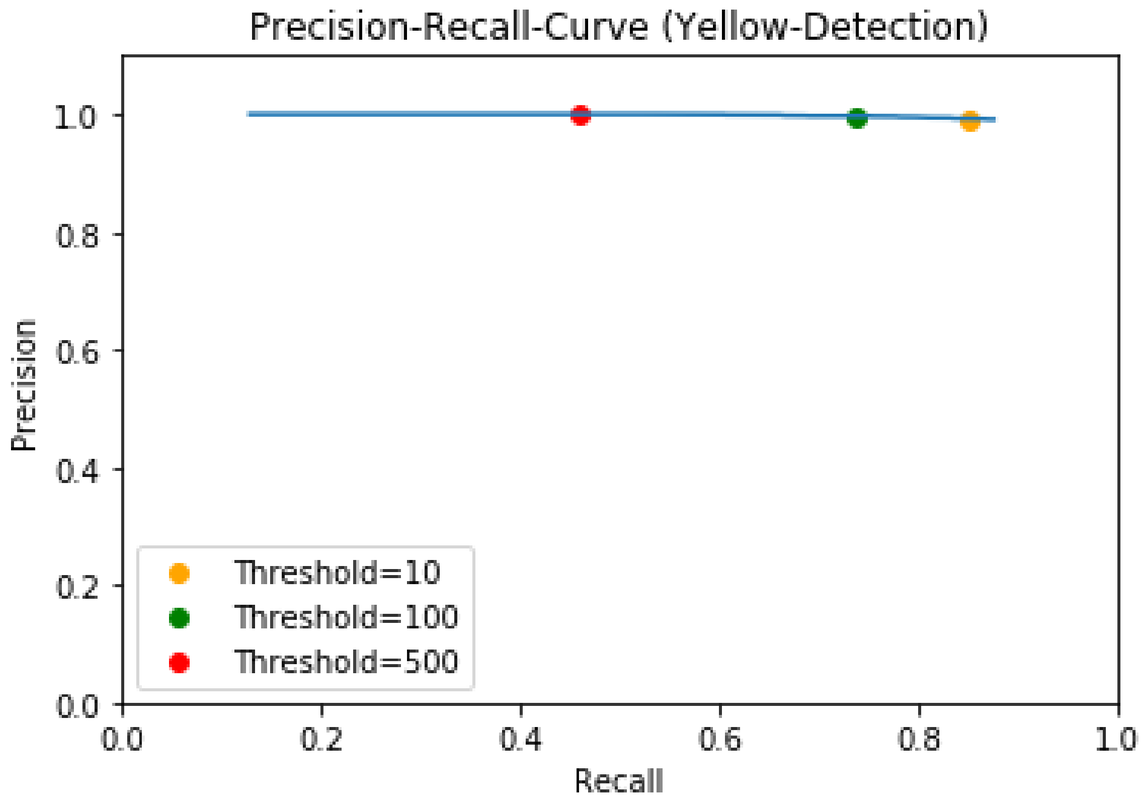
- ≅ 57% of samples contain no red in Ground Truth
- Output often contains small red artifacts
- →Threshold-Procedure to decrease False-Positives (=False Alarms)

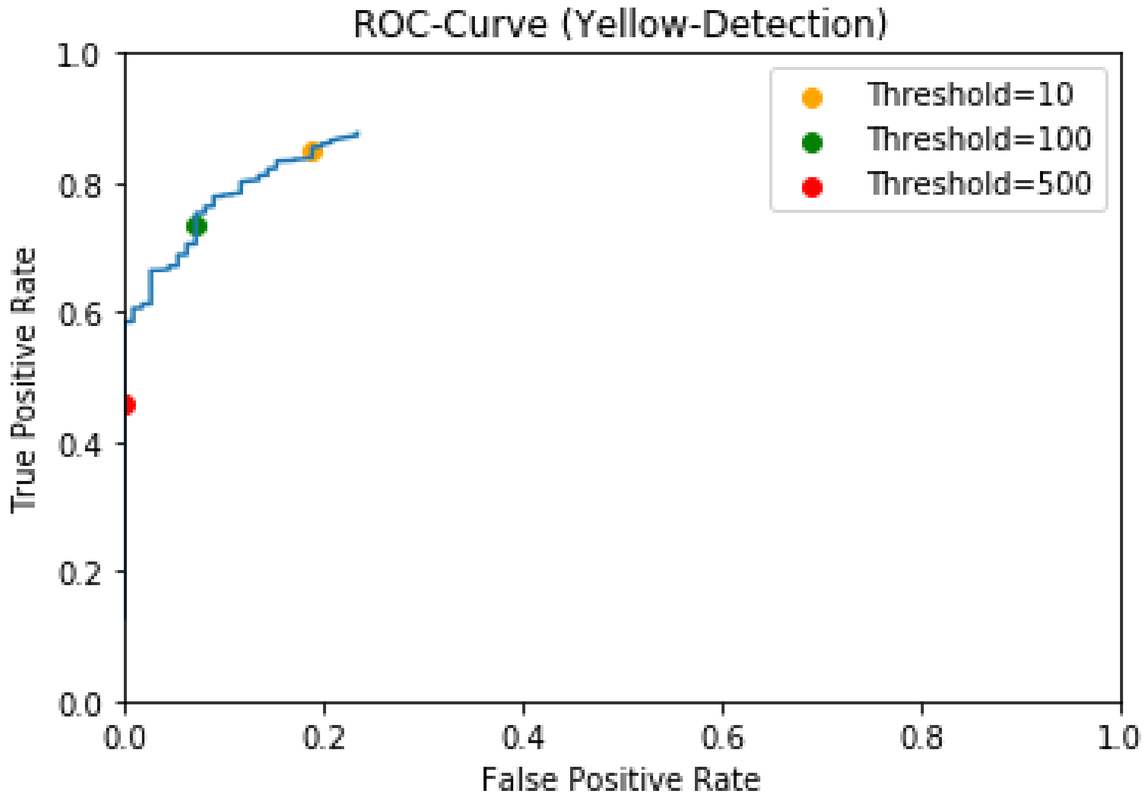
Validation - Validation Set





Validation - Validation Set





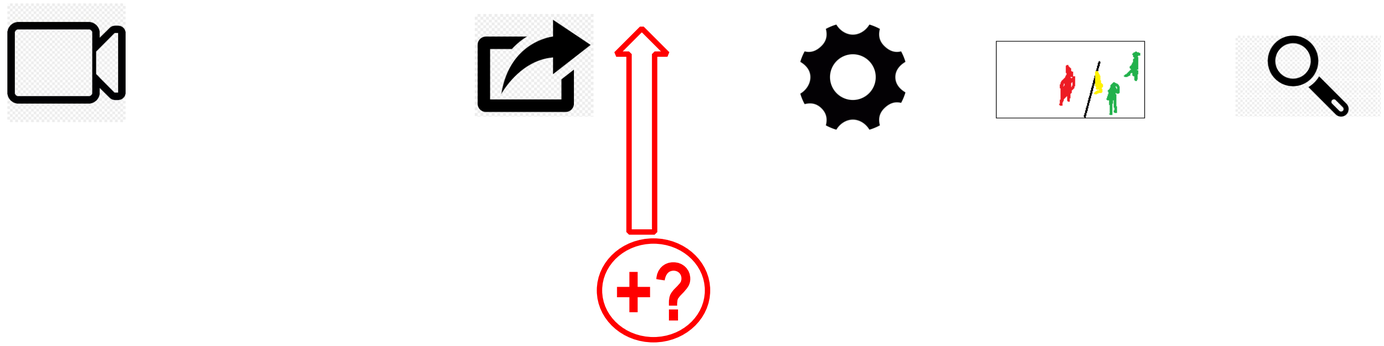
Validation using real data

- **How well do models trained in simulations generalize to reality?**

- Expectations like those in the Performance Requirements:
  - ↑ amount existing dangers detected, ↓ amount false alarms
  - ideally results as good as previous results which used simulated data

- Idea:

Camera recording → extracted frames → prototype → output → analysis



Validation using real data – Searching for good camera recordings

- Looking for:
  - underground station
  - similar point of view
  - humans across the scene
  - good enough quality to recognize things

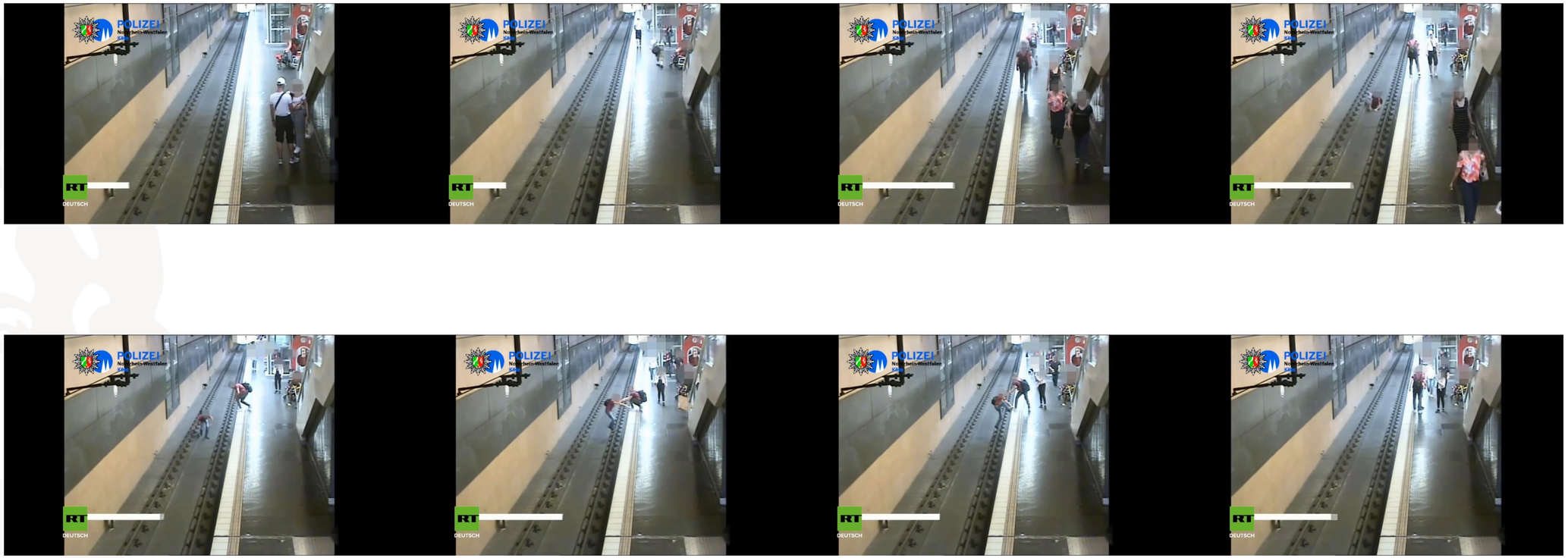
→ **Very hard to find such videos!**

- Chosen video: “Koelner Schubser” (2018)

Validation using real data – The chosen video (“Koelner Schubser”)

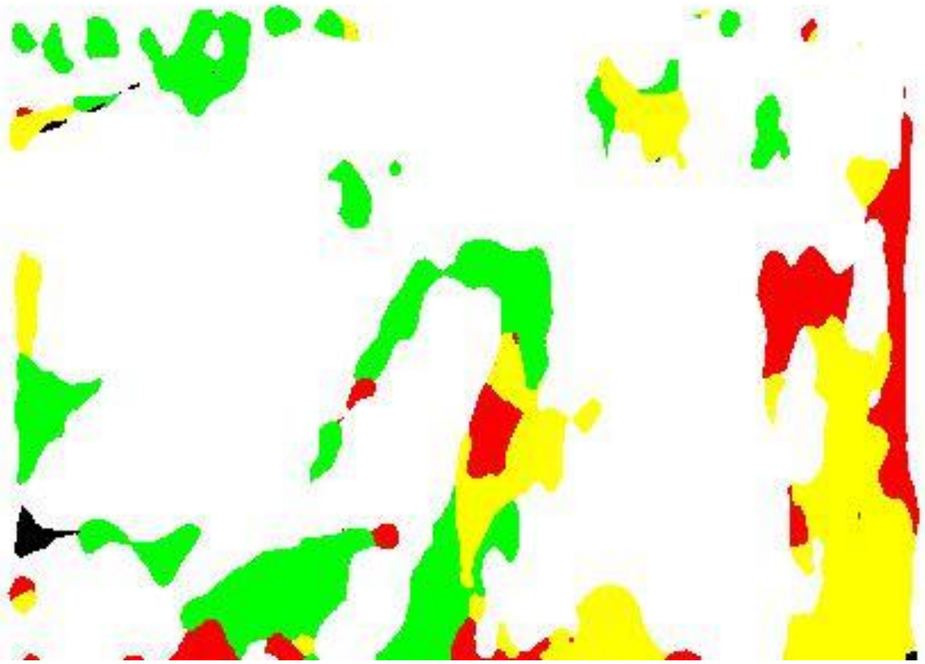
Validation using real data – Extracting Frames

- 77 interesting frames

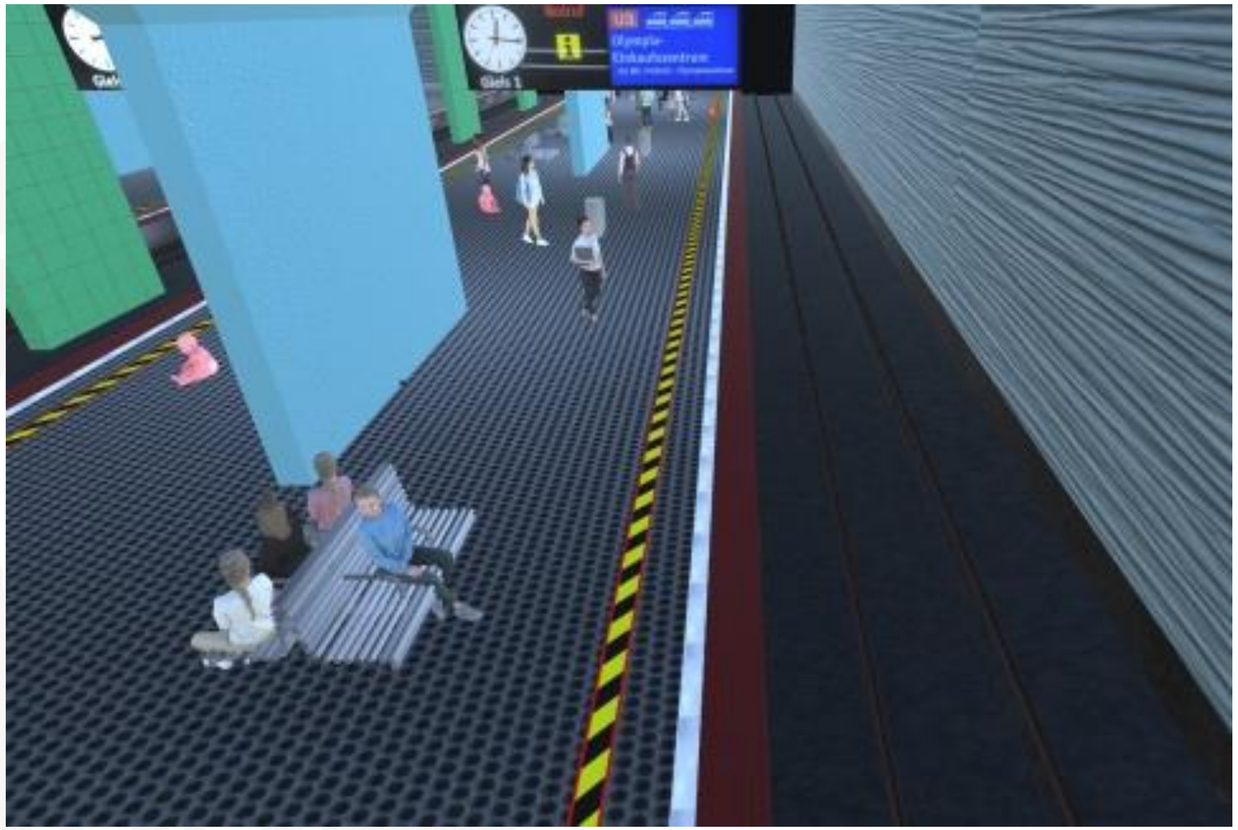


POLIZEI
Nordrhein-Westfalen
Köln
RT
DEUTSCH

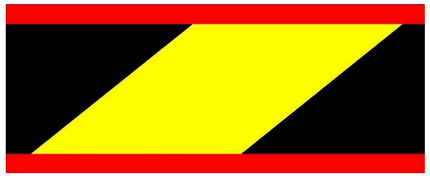
Validation using real data – First results

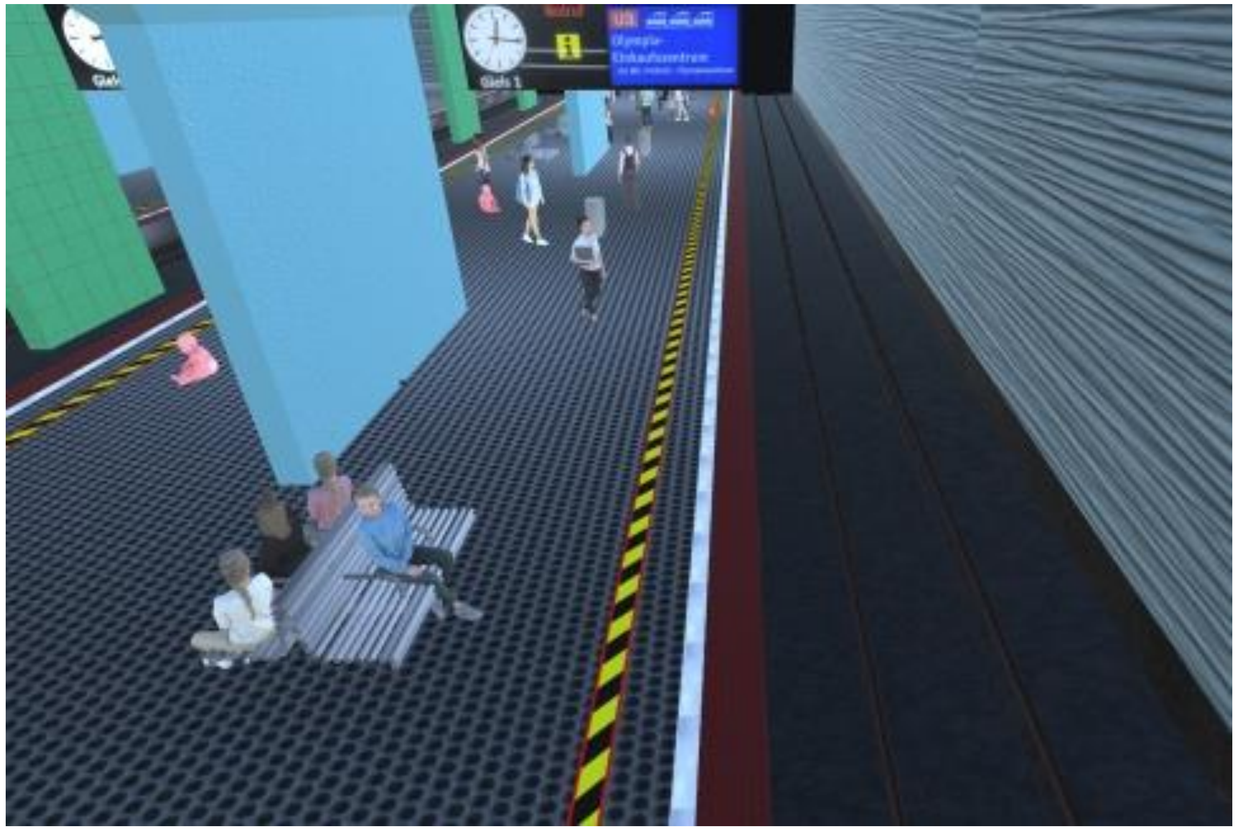


Validation using real data – First results

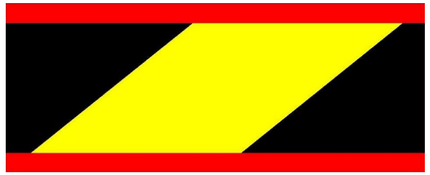


Validation using real data – There is no warning line!





Validation using real data – There is no warning line!



- **Solution**: Manually add the line to each frame.

POLIZEI
Nordrhein-Westfalen
Köln
RT
DEUTSCH

Validation using real data – First results

- ✓ Required line got added
  - ➢ now we can start!

Validation using real data – Judging first results

- Good detection of existing warning line
- Unsatisfying detection of actual danger
- Unsatisfying amount of colored fragments
- Unsatisfying categorizing (colors of detected people)

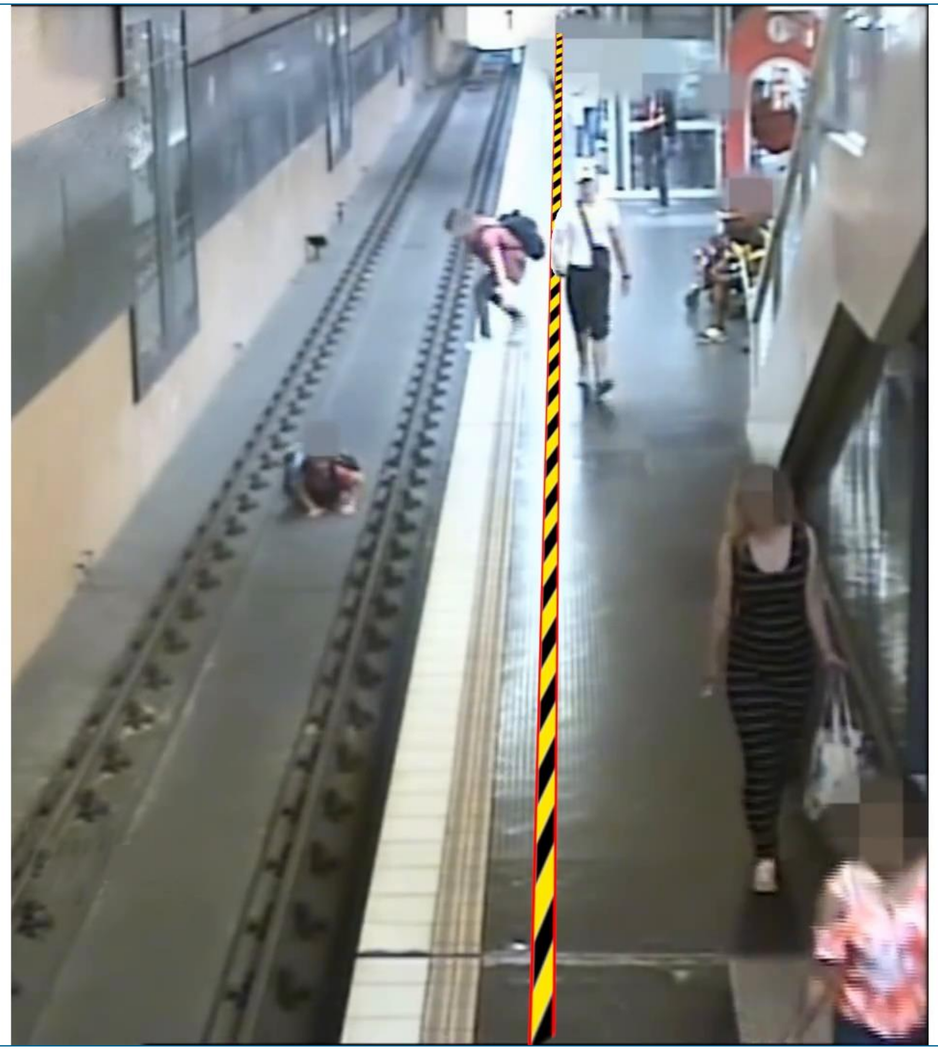
Validation using real data – Afterthoughts

- **Was the food too spicy?**

Validation using real data – Adjusting input data

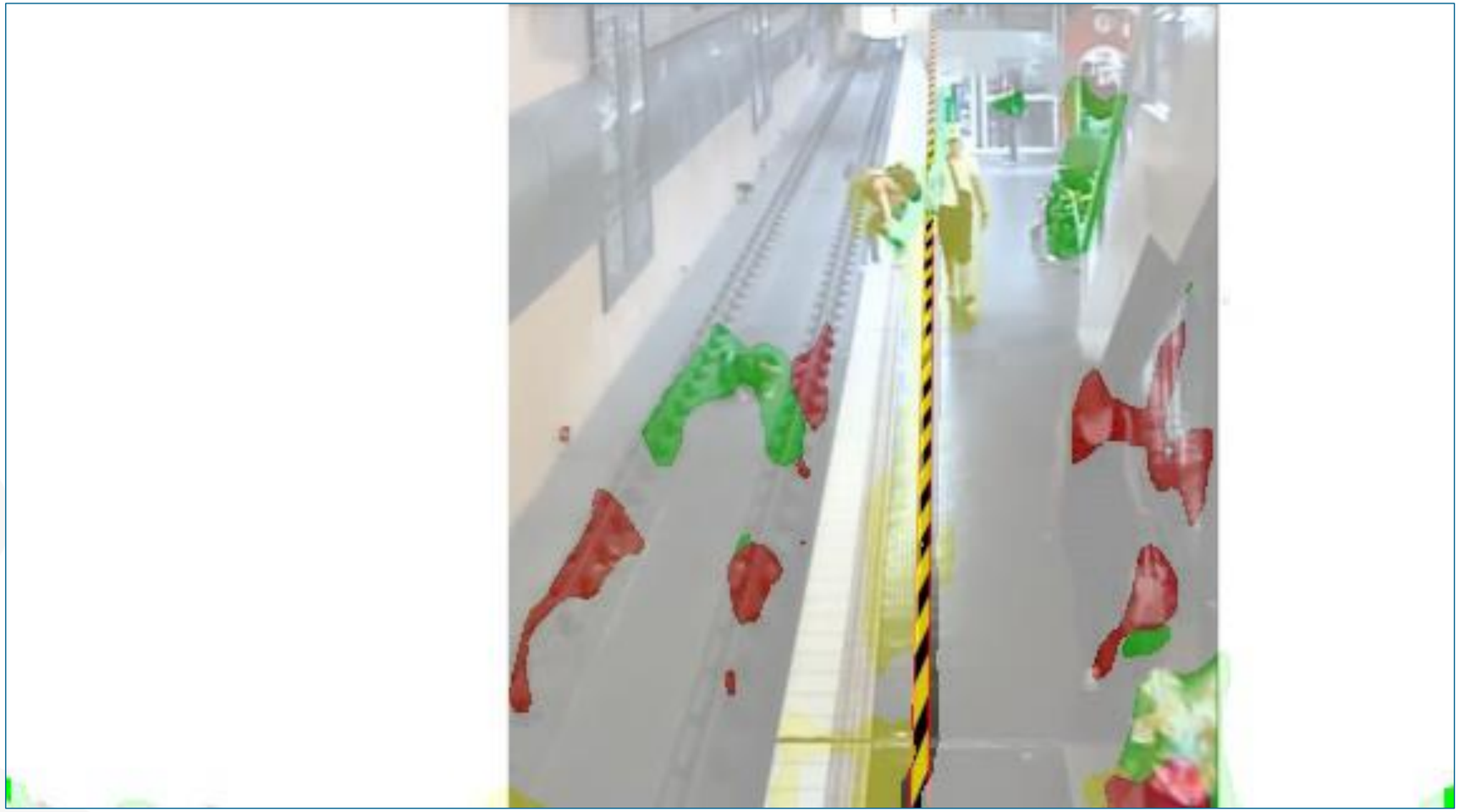
- Let's try again:
  - Cut off **edges** to the side
  - Remove static **pole** on the top right corner
  - Remove RT and NRW police **logos**

Validation using real data – Adjusting input data



Validation using real data – New output

Validation using real data – Comparing



Validation using real data – Verdict

- Much better results after input adjustments
- Still issues with detecting dangers (0% precision red)
- Still issues with categorizing people correctly
- There is still a lot to do on this project, but given that this is our very first iteration, it looks very promising!

Weaknesses of current implementation

- Detection of tracks
- Reflections
- Random things being interpreted as a warning line
- Lack of training assets
- Uncommon events

Recommendations for future Development-Cycles

The simulation environment should include

- Varying lighting conditions
- Reflective and opaque surfaces
- More variety in stationary objects and moveable objects
- Generative model for station types
- Animation capabilities
- Virtual Replicas of real train stations

→Implement a human tracking system and train on generated video data
→Fine tune system with specific scenarios (accidents and on purpose)
→Respect Real Time Requirement

Summary

User View
- Specified Context, Task and Performance requirements for subway station security system

Modeler View
- Modelled specified Context
- Considered several approaches for Implementation

Implementation View
- Implemented Subway Station Simulations
- Implemented (basic) Danger Detection System

Validation View
- Labelled Real Data for Validation
- Analyzed System Performances

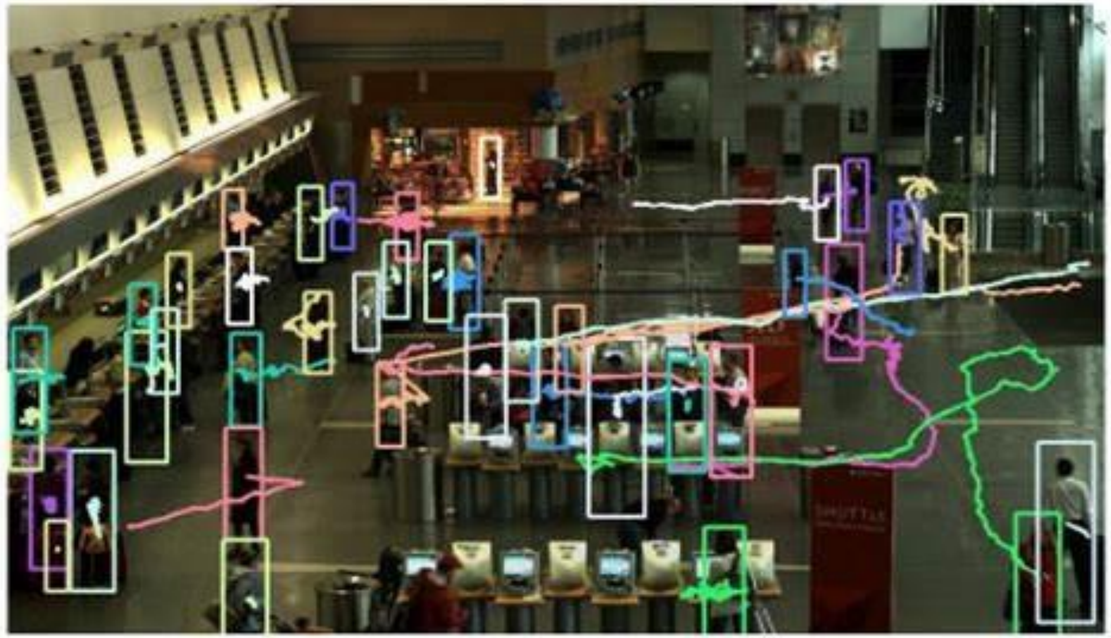
Outlook

Expansion of the simulation environment

- More realistic through lighting, reflection, opacity
- More diversity in many aspects
- Animated Simulation instead of single images

Hazard Detection System:

- Real Time people tracking and movement prediction



Thank you for your attention

All sources can be found in the accompanied document