Bachelor Thesis

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of cybernetics

Semantic Segmentation for Autonomous Student Formula Race Track Localization

Josef Capůrka

Supervisor: Ing. Jan Čech, Ph.D.
Study program: Open Informatics
Specialisation: Artificial Intelligence and Computer Science
May 2023

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Capůrka Josef**	Personal ID number:	**499321**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Semantic Segmentation for Autonomous Student Formula Race Track Localization

Bachelor’s thesis title in Czech:

Semantická segmentace pro nalezení trati závodu autonomní studentské formule

Guidelines:

The race course of the autonomous student formula is delineated by traffic cones placed along the borders. The previous approach consisted in two steps: (1) detecting cones in the image, and subsequently (2) heuristically finding the inner and outer parts. In the case of a more complex track and potential detector failure, the segmentation into inner and outer parts may be ambiguous. Design a method that formulates the problem as a model-based predictor that provides segmentation into inner and outer parts of the race track in the image.
Evaluate the predictor quantitatively using the ground-truth annotated dataset. Compare the proposed predictor with the baseline approach (cone detections + track localization).

Bibliography / sources:

[1] J. Redmon, S. Divvala, R. Girshick, A. Farhadi. You only look once: Unified, real-time object detection. In CVPR, 2016.
[2] Olaf Ronneberger, Philipp Fischer, Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI, 2015.
[3] Swapnil Waykole, Nirajan Shiwakoti, Nirajan Shiwakoti, Peter Stasinopoulos. Review on Lane Detection and Tracking Algorithms of Advanced Driver Assistance System. Sustainability 13(20):11417, 2021.
[4] Roman Sip. Visual Detection of Traffic Cones for Autonomous Student Formula. Bachelor's Thesis, Czech Technical University, FEE, 2022.

Name and workplace of bachelor’s thesis supervisor:

Ing. Jan Čech, Ph.D. Visual Recognition Group FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **06.02.2023** Deadline for bachelor thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

Ing. Jan Čech, Ph.D.	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.

Date of assignment receipt Student’s signature

Acknowledgements

I would like to thank my supervisor, Ing. Jan Čech, Ph.D., for the valuable guidance and advice he has given me throughout the year. I would also like to express my gratitude to all members of the eForce Driverless team. Last but not least, I want to thank my family for their support. The access to the computational infrastructure of the OP VVV funded project CZ.02.1.01/0.0/0.0/16_019/0000765 “Research Center for Informatics” is also gratefully acknowledged.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, May 26, 2023

Abstract

The thesis presents a race track visual localization method for Autonomous Student Formula. The track is delineated by traffic cones. We propose three different approaches for track localization: Segmenting the race track with a segmentation mask, predicting the boundaries of the race track with direct regression, and predicting the boundaries with heatmap regression. All of these approaches utilize convolutional neural networks. The annotated dataset used for training was collected specially for this problem. We quantitatively evaluated the accuracy of the models and compared them to the baseline approach. The baseline first detects the traffic cones by YOLO detector and then uses a heuristic algorithm to find the track. We show that localizing the race track with a segmentation mask produced by the UNet model achieves accuracy 0.93 IoU. The segmentation with our UNet model often outperforms the baseline in complicated tracks.

Keywords: Race track localisation, Semantic segmentation, Formula Student Driverless, UNet, ResNet

Supervisor: Ing. Jan Čech, Ph.D.

Abstrakt

Tato práce se zabývá metodou pro lokalizaci vnitřku trati pro autonomní studentskou formuli. Trať je ohraničena dopravními kužely. Navrženy byly tři různé přístupy pro lokalizaci trati: Segmentace trati se segmentační maskou, predikce hranic trati s přímou regresí a predikce hranic trati pomocí heatmap. Všechny tyto metody využivají konvolučních neuronových sítí. Pro trénování sítí byl použit námi sesbíraný anotovaný dataset. Kvantitativně jsme vyhodnotili přesnost modelů a porovnali je se stávajícím přístupem. Stávající přístup nejprve detekuje dopravní kužely YOLO detektorem a následně používá heuristický algoritmus pro nalezení trati. V práci ukazujeme, že lokalizace vnitřku trati za pomocí segmentačních masek vytvořených UNet modelem často překonává baseline metodu na komplikovaných závodních tratích.

Klíčová slova: Lokalizace závodní trati, Sémantická segmentace, Formula Student Driverless, UNet, ResNet

Překlad názvu: Semantická segmentace pro nalezení trati závodu autonomní studentské formule

Contents

Figures

Tables

Chapter 1

Introduction

This bachelor thesis presents a visual predictor for the inner part of the race track of the Formula Student Driverless competition. The author of this thesis is eForce Driverless team member. In the competition, traffic cones delineate the race track as can be seen in Figure 1.1. Unlike many teams that use LiDAR or YOLO-based detectors to detect cones and then apply heuristic algorithms for race track localization, our proposed methods segment the inner part of the race track in a single processing step on the image. The segmentation of the race track could be used to navigate the vehicle during autonomous driving.

The structure of the thesis is as follows. In Chapter 1, we provide a brief introduction of the Formula Student competition and the eForce Formula Student team. Next, we discuss how the current autonomous pipeline works, and since the current pipeline has its limitations, we discuss them and suggest how we can improve them with the proposed localizer of the race track. In Chapter 2, we provide a theoretical background that is needed to understand the details of the implementation of the proposed methods used for the localization of the track. The methods used to localize the race track are described in Chapter 3. In Chapter 4 we describe the dataset used for training the localizer. Next, the training process of the proposed methods is discussed in Chapter 5. The results of the proposed methods are shown in Chapter 6. Finally, in Chapter 7, we discuss the advantages, limitations, and future improvements of our proposed approach.

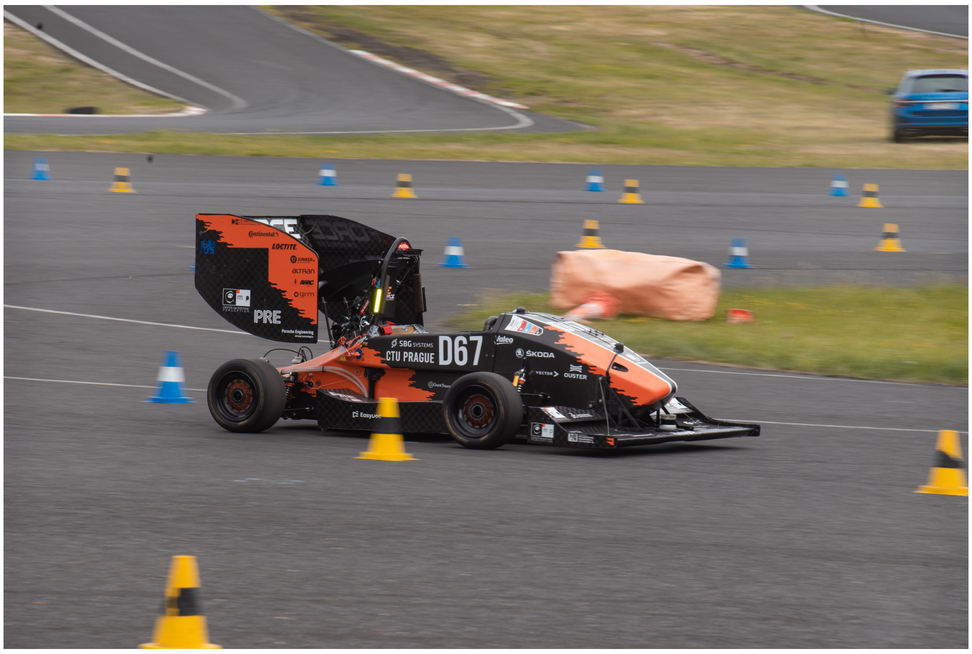


Figure 1.1: eForce Driverless formula during racing [1]

1.1 Formula Student

First, we will briefly describe the Formula Student competition, since the race track localizer is designed for this competition. Formula Student is an international engineering competition that focuses on the design and construction of student formulas. Teams from all over the world develop their own formulas to participate in this challenging competition. In this section, we will describe Formula Student Driverless competition disciplines and their rules.

1.1.1 Static disciplines

The competition is not focused only on racing. It consists of static disciplines and dynamic events. Static disciplines consist of 3 parts: an engineering design event, a cost report, and a business plan. The purpose of the engineering design event is to present the proposed design of the car, show knowledge about the proposal and justify steps that led to the proposal. The cost report provides a detailed description of the expenses involved in designing and building the formula. The aim of the business plan presentation is to create and present a business model into which potential investors could invest.

Figure 1.2: Photography of teams participating in the Formula Student Czech Republic [2]

1.1.2 Dynamic events

The dynamic events take place on the race track. The boundaries of the track are circumscribed by 4 types of traffic cones: Big orange cones, small orange cones, blue cones, and yellow cones. Each type of cone has its specific purpose shown in Figure 1.3, where we can see the race track of the acceleration event, and in Figure 1.4, where we can see the race track of the skidpad event. The dynamic events consist of

1. Acceleration event,
2. Skidpad event,
3. Trackdrive event,
4. Autocross event.

The shape of the race track in the Acceleration and the Skidpad discipline is defined in the rules and so is known in advance, whereas in Autocross and Trackdrive disciplines, the shape of the race track is previously unknown. The tracks for these disciplines are more complex. The aim of the acceleration event is to pass a 75-meter straight track and successfully stop the car in the stop area as can be seen in Figure 1.3 The distance calculated between the left boundary and the right boundary of the race track is at least 3 meters. In the Skidpad event, the car first passes twice through the right part of the

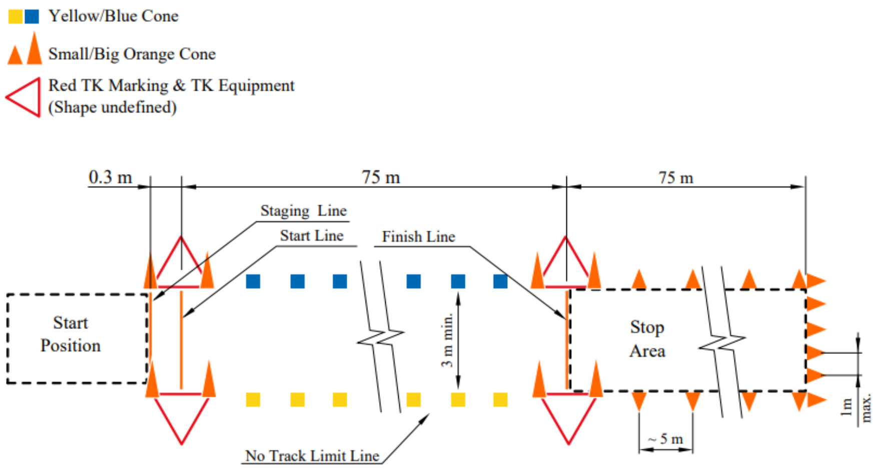


Figure 1.3: Acceleration event [3]

track, then twice through the left part of the track. Finally the car stops in the stop area as can be seen in Figure 1.4

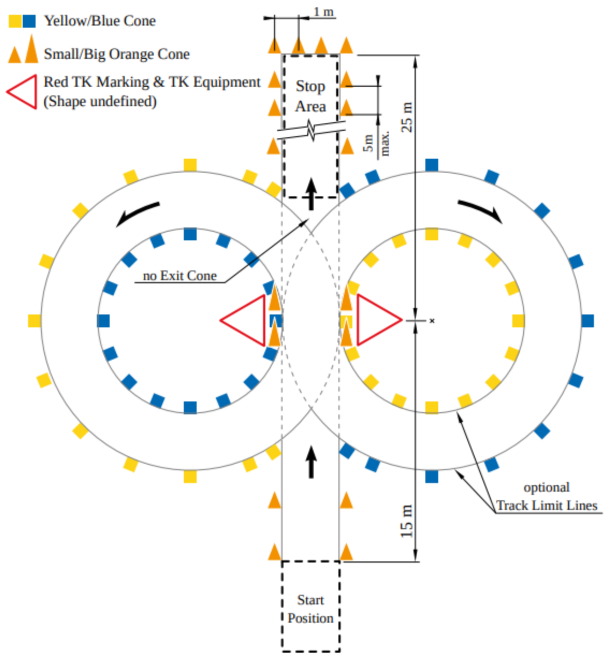


Figure 1.4: Skidpad event [4]

As we said, even though the distances between the cones are specified, the shape of the track in the Autocross and Trackdrive event is unknown prior to the race. The Trackdrive event is an one lap race. Its purpose is to test the reactive behavior of the driverless system, whereas Autocross consist of 10 laps which allows the use of the SLAM algorithm. The overall score of dynamic disciplines is affected by the number of cones hit during the race [11].

1.2 eForce Driverless

The first team in the Czech Republic to build an electrical student formula was the eForce FEE CVUT. In 2020, eForce introduced the first driverless student formula in the Czech Republic, eForce DV.01 and successfully participated in many races. As there are new demands from the automotive industry, eForce is currently building FSE.12, an electrical monopost that will have integrated autonomous functionality. Since our localizer will be deployed in the eForce driverless formula, we will now briefly explain the autonomous pipeline of the formula DV.01 2022, mainly the perception and path-planning node, because they are important in our thesis.

1.2.1 Perception

Unlike many teams that use LiDAR for cone detection, our pipeline consists of a camera-based perception system. Information about the vehicles surroundings is taken by a Stereolabs ZED 2 camera. The approach of most teams that use a camera for vision is to detect bounding boxes of cones or predict segmentation masks of cones. In our current system, a single camera is used. The RGB images are forwarded to a cone detector node that predicts the cone's bounding boxes with a real-time object detection YoloV3-based algorithm[12]. Then the estimated cone centers are computed from the predicted bounding boxes. After estimating the cone center, the center points are projected to the local vehicle coordinate system using known homography mapping. The projection matrix used for this transformation is obtained from the camera calibration process[12]. The cone detections can be seen in Figure 1.5.

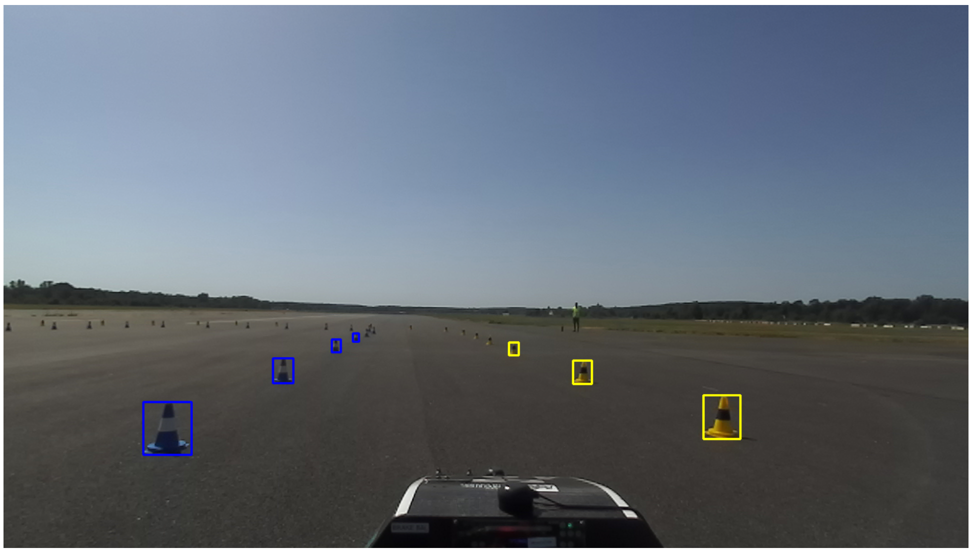


Figure 1.5: Detected bounding boxes of cones

YOLO

You Only Look Once (YOLO) is an object detection algorithm introduced in 2016. In a single forward pass, the network produces bounding boxes with the corresponding class probabilities and labels of the bounding box. One of its main advantages is its quick processing of image, so it can be used for real-time object detection [13].

1.2.2 Path planning

The input to the path planning algorithm are the projected center points of the detected cones. The heuristic algorithm iteratively connects the respective blue and yellow cone centers that are closest in the scene. The algorithm then reconstructs the path from those cones, which can be seen in Figure 1.6, where the boundaries of the race track are represented by blue and yellow lines. The centerline is represented by a red line. In Figure 1.7, we can see the boundaries of the race track obtained from planning algorithm projected into image by applying inverse homography mapping.

The planning algorithm has 2 modes. The reactive mode is activated while the car drives in the first round and observes information about its surroundings. After the completion of one lap, thus collecting data about car surroundings, the car is able to switch to an optimal path-planning algorithm with a speed profile as the map of the race track is known [14]. This year, the speed profile for the first lap was implemented by Dmytro Khursenko [15].

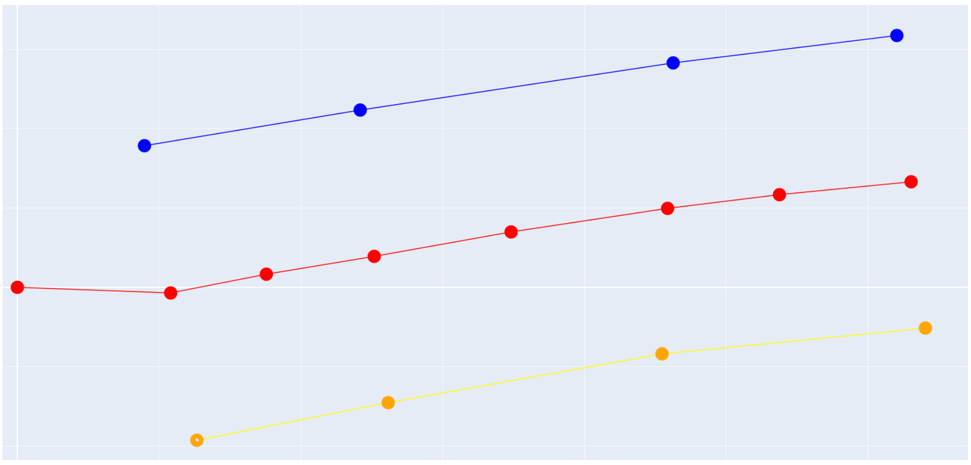


Figure 1.6: Projected center of cones and centerline points into local car coordinates

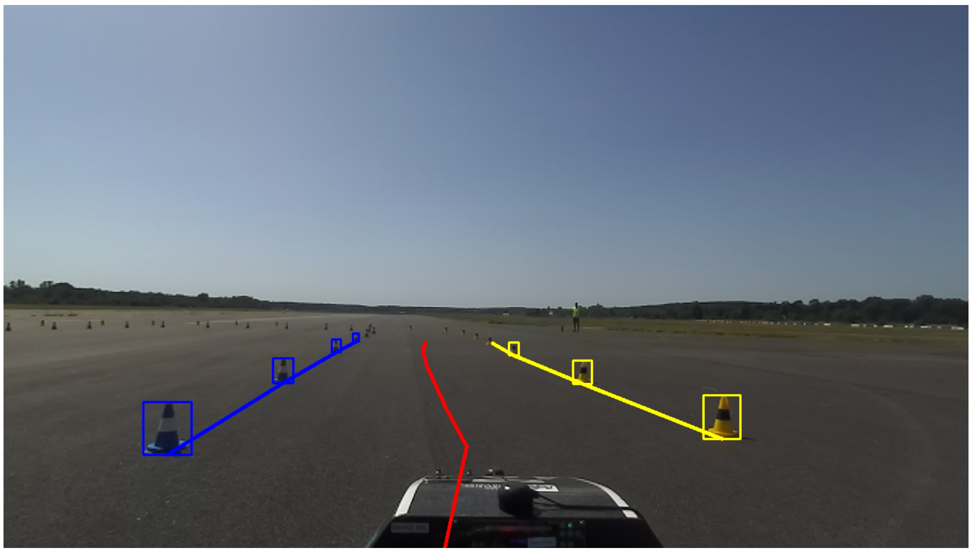
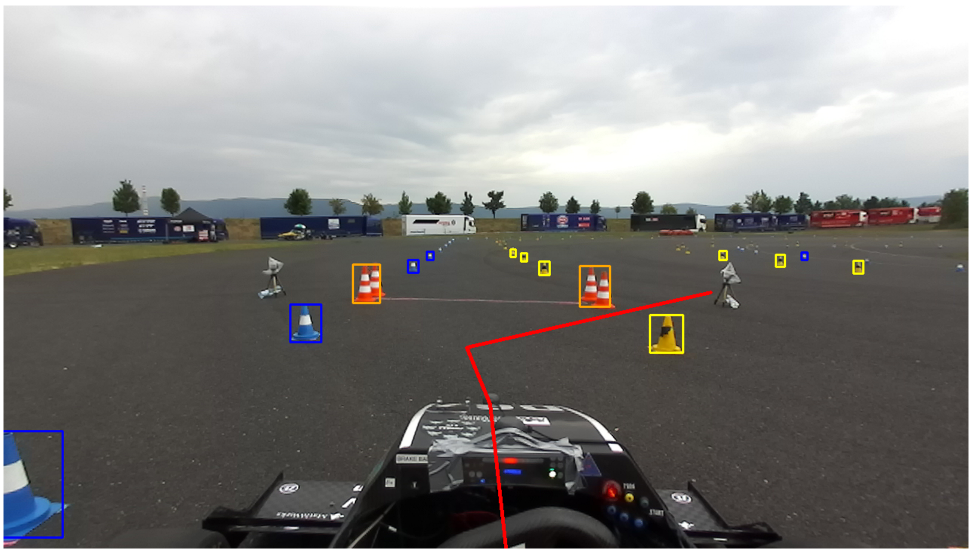


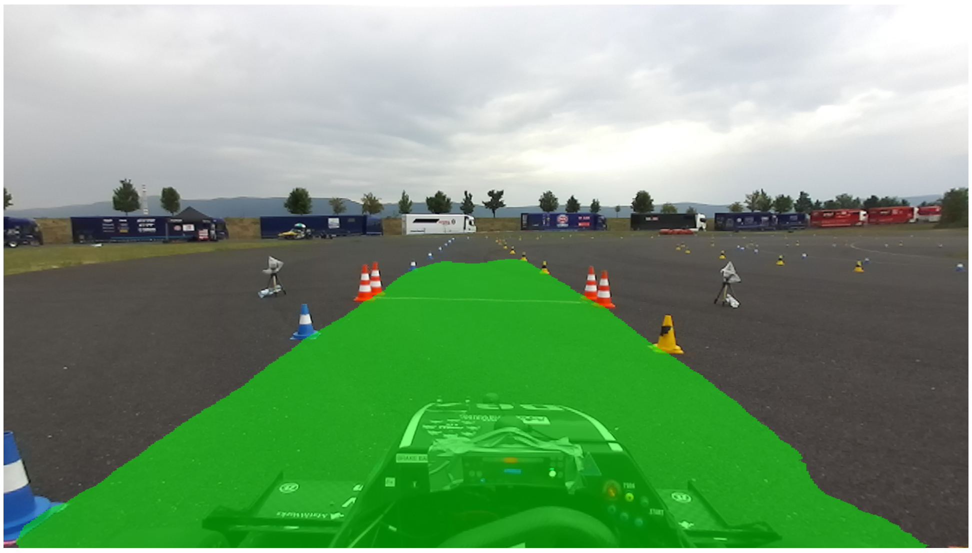
Figure 1.7: Image from test day in Milovice

1.2.3 Limitations of the current approach and how to overcome them

The reason why we decided to design a race track localizer is that the baseline approach may fail in cases where the detector misses detecting some cones or the heuristic planning algorithm predicts the path incorrectly in case of the tracks of greater complexity, as can be seen in Figure 1.8a. The objective of this thesis is to design a reliable localizer that is trained end-to-end from in-race images. Thus the localizer learns the prior distribution of the track and does not rely on a fragile connection between the bounding box detector and the heuristic estimator. Our proposed methods will utilize convolutional neural networks because they have the capability to learn higher-level semantic relations and thus are able to segment complex objects.



(a) : Baseline approach (cone detector + path-planning)



(b) : Our proposed method producing segmented inner part of the track

Figure 1.8: Comparison of baseline approach and our proposed method

1.2.4 Thesis contributions

In this thesis, we describe three end-to-end training methods for segmentation of the inner part of the race track. All of these methods use CNNs. Since the training dataset was not available for this problem, we collected and annotated images for training the models. We then trained the models, evaluated them on an independent test dataset and compared them with the baseline method.

Chapter 2

Theoretical background

This Chapter will cover the theoretical background that is needed for understanding the implementation part. First, we will describe the neural network architectures that will later be used for the localization of the race track. We will describe the architecture of UNet and ResNet neural network. Lastly, we will explain what fine-tuning is since it was also utilized in our work.

2.1 Semantic segmentation

Semantic segmentation is a computer vision problem whose task is to divide an image into segments where each segment corresponds to some predefined class. It is used widely in medical imaging in organ segmentation or tumor detection. It is also possible to deploy a semantic segmentation in the case of autonomous driving for track prediction[16]. The main purpose is to semantically distinguish each pixel in an image and assign it a class as can be seen in Figure 2.1.

Nowadays, the common approach is based on deep learning architectures. Semantic segmentation is often performed with fully convolutional neural networks or transformer-based architectures[17].

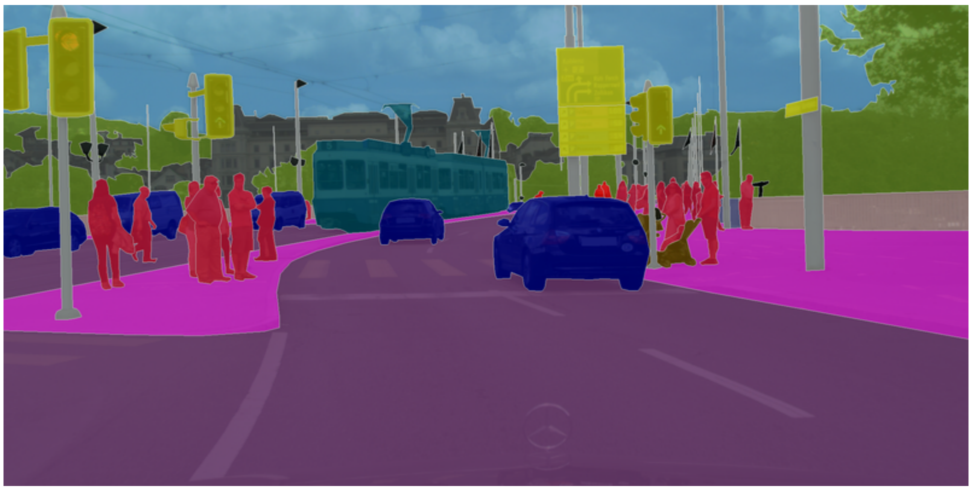


Figure 2.1: Example of semantic segmentation result [5]

2.1.1 U-Net architecture

U-Net is a fully convolutional neural network used for semantic segmentation [18]. The architecture is shown in Figure 2.2. UNet uses the encoder-decoder

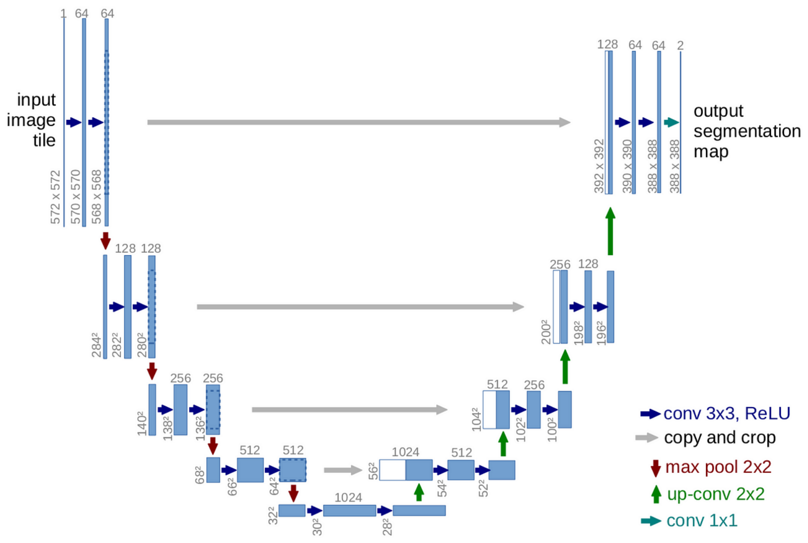


Figure 2.2: U-Net architecture [6]

approach. The left part of the U-shaped architecture is called the contracting path (encoder) and the right part is called the expansive path (decoder). The first four layers in the contracting path consist of two 3×3 convolutions and after each convolution, the ReLu function is applied. As can be seen in Figure 2.2, the max pool operation is then performed which results in a loss of spatial resolution. The output of the contracting path are low-level features that lack spatial resolution. Since we need to obtain a segmentation mask in the original input resolution, we need to gain back the spatial resolution. The expansive path creates a segmentation mask from low-level features. One of the advantages that UNet use are skip connection. Skip connections allow forwarding the output from the encoder layer to the decoder layer (the grey lines in Figure 2.2). Each layer in the expansive path consists again of 3x3 convolutions, after them the ReLu function is called. The transposed convolutions and forwarded data from skip connections help to increase spatial resolution[18].

2.1.2 ResNet architecture

For predicting boundary points of the race track we used ResNet, so we will also describe its architecture in more detail. ResNet is a very deep feedforward neural network with different numbers of layers. [19] There are many ResNet variants that are distinguished by the number of layers: ResNet 18, ResNet34, ResNet 50, ResNet 101, and many more. The image describes the architecture of ResNet 34.

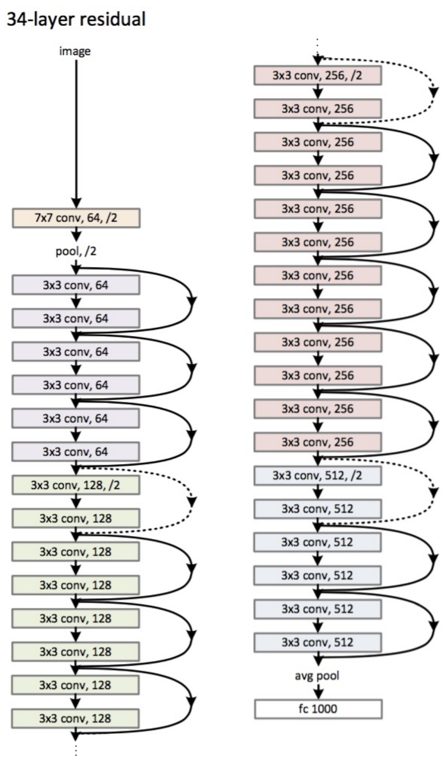


Figure 2.3: ResNet architecture [7]

Before ResNet was introduced, deep neural networks suffered from bad performance due to vanishing and exploding gradients. Since in deeper networks after many layers the gradients were zero due to chain rule and backpropagation, the weights were not efficiently updated. ResNet tackles this problem and introduced Residual Blocks.

Residual block

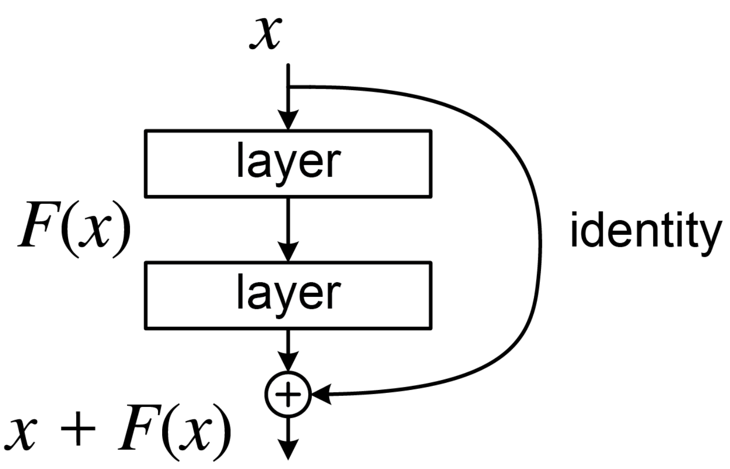


Figure 2.4: Residual block [8]

The residual block allows the information to be passed from the beginning of the layer to the end. This forwarding tackles the vanishing gradient problem so deeper networks do not suffer from vanishing or gradient problems anymore. The residual block with identity mapping can be seen in Figure 2.4, however, convolutional mapping and other types of mappings are implemented.

2.2 Fine-tuning

Since in our proposed method, we utilized fine-tuning to obtain better results, we will briefly explain its underlying concepts. Fine-tuning is a commonly used method for obtaining better performance in neural networks. As there are cases when the dataset is not large enough, the idea is to pre-train the model on a dataset of similar characters. Since the datasets are similar, the network could learn features that could be useful for the target dataset. Typically, when the pre-trained weights are used, most of the layers of the network are set as untrainable, and only n last layers are set as trainable [20].

Chapter 3

Proposed methods

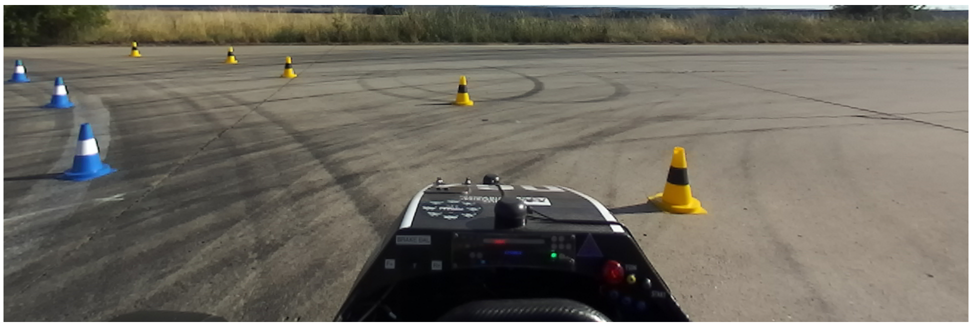
In this Chapter, we will describe the methods that we designed and implemented for the localization of the race track. We will explain three different approaches.

3.1 Prediction of segmentation masks

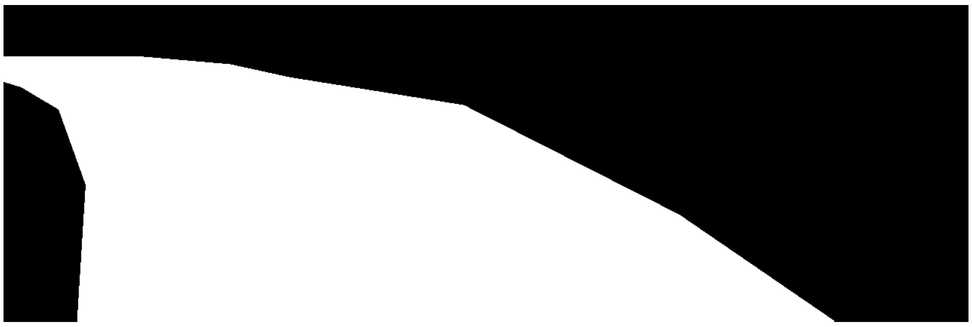
One of the methods that can be used for the localization of the race track is to classify each pixel in the image as either the outer part of the race track or the inner part of the race track. The approach can be understood from Figure 3.1, where Figure 3.1b contains a segmentation mask for Figure 3.1a. On the mask, the white pixels represent the inside of the track and the black pixels represent the outside of the track.

3.2 Prediction of the race track boundary coordinates

The alternative approach is to formulate the localization of the race track as a regression problem. It is possible to predict points that correspond to the boundaries of the race track, thus approximating the boundary as a piecewise linear curve. In principle, this approach is an easier problem



(a) : Image captured during autonomous driving



(b) : Corresponding mask

Figure 3.1: Illustration of segmentation approach

than the production of a segmentation mask since only 20 points for each boundary of the race track need to be created. In contrast, the segmentation mask evaluates every pixel in the image. We provide a illustration of which points are predicted in this method for easier understanding. In Figure 3.2, the blue line represents the left boundary of the race track and the yellow line represents the right boundary of the race track. The points that are being predicted are formed by the intersections of the race track boundaries with the horizontal grey lines. These points are highlighted with blue and yellow points in the image. For further explanation, it is important to note that the values on the y-axis of the blue and yellow points are defined by the horizontal segments, so our method predicts only the values of the points on the x-axis. If the horizontal gray line does not intersect with the blue line, the x coordinates of the ground truth points are set to 0, i.e. the points lie on the left edge of the image and, correspondingly, for the yellow line, the value of x coordinates is set to width of image, so the points lie on the right edge of the image. From the obtained predicted points, it is possible to reconstruct the left boundary and the right boundary of the race track. It is also possible to create a polygonal mask similar to the mask in Figure 3.1b.

There was a problem with the track localization via segmentation masks that meant that the track was sometimes erroneously detected outside of the delineated zone, splitting the track into two sections. This problem is illustrated in Figure 3.3. The prediction of track boundaries eliminates that

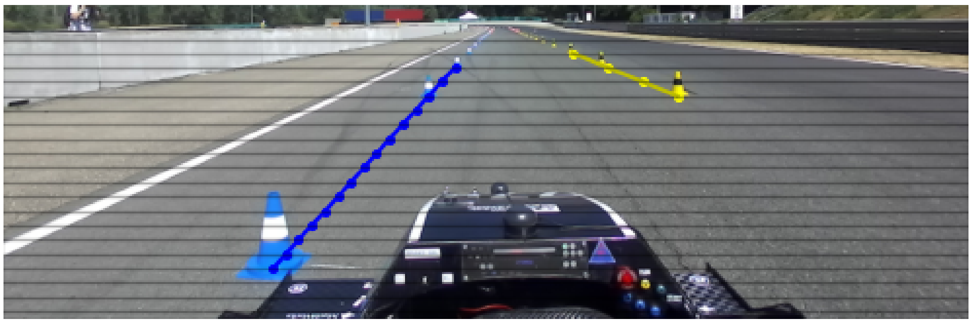


Figure 3.2: Ground truth labeled image. The blue and yellow lines are ground truth labels. The corresponding blue and yellow points are being predicted in this method. The grey lines are added for illustration.

problem by ensuring that the track does not split, since for each grey line from Figure 3.2, we can predict only one value on the x-axis for each boundary.

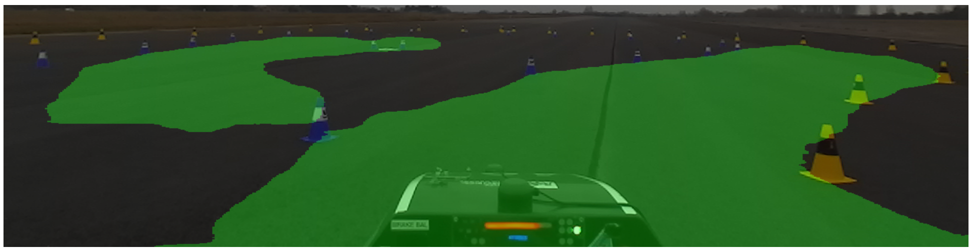


Figure 3.3: Segmented race track with mispredicted pixels in the left part

3.3 Prediction of the race track boundary via heatmap regression

To clarify what heatmap regression is, let us briefly describe this method. In heatmap regression problems such as facial landmark detections, instead of predicting the exact coordinates of the landmark points, the heatmap for each landmark is predicted, reflecting the probability of occurence of the point.

In Figure 3.4 in the leftmost image, we can see the annotated image with facial landmarks. The other five images contain corresponding heatmaps for each facial landmark. Typically, the heatmaps are generated from ground truth annotated points by applying a Gaussian kernel, and the value of each pixel in the image can be interpreted as a confidence that the landmark is located at that position[21]. It has been shown that, in many cases, heatmap regression methods outperform direct regression methods and achieve state-of-the-art results [21]. Because of that, we will utilize this method to potentially improve the accuracy of the predictions.

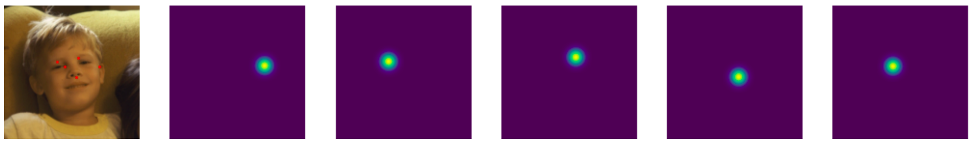


Figure 3.4: Ground truth annotated image with five heatmaps [9]

Now we will explain how we predict the heatmaps in the problem of predicting boundaries of the race track. For this, we will also use a visual approach.

Instead of predicting $2n$ values on the x-axis as we presented in the previous section, in this approach, we predict n one-dimensional heatmaps for each boundary of the race track. As in the previous method, we can approximate the race track boundaries with piecewise linear lines. The single one-dimensional heatmap for the left boundary of the track can be seen in Figure 3.5, where we rescaled the height of the image for clearer visibility.

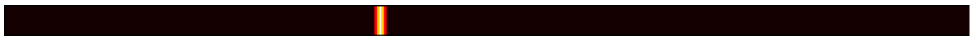


Figure 3.5: One dimensional ground-truth heatmap for the left boundary of the track representing the confidence of pixels being on the position of left boundary of the race track

Each one-dimensional heatmap has a predefined value on the y-axis in the image and these y values of the one-dimensional heatmaps are the same as the y values of the grey lines in Figure 3.6. Figure 3.7 displays the unified heatmaps in one image. The focal points of the heatmaps are at the same positions in the image as predicted points from the method, where we predict exact x-axis coordinates of the points in the image. For clarification, when there is no intersection between the boundary of the race track and the grey line, the heatmap corresponding to that grey line contains only zero values. Overall, in this method, we predict a tensor of shape $(20 \times \text{width of the image})$ for each boundary of the race track.

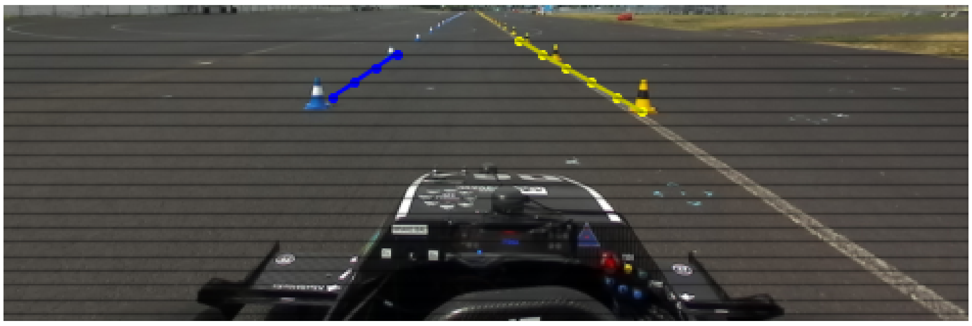


Figure 3.6: Ground truth annotated image

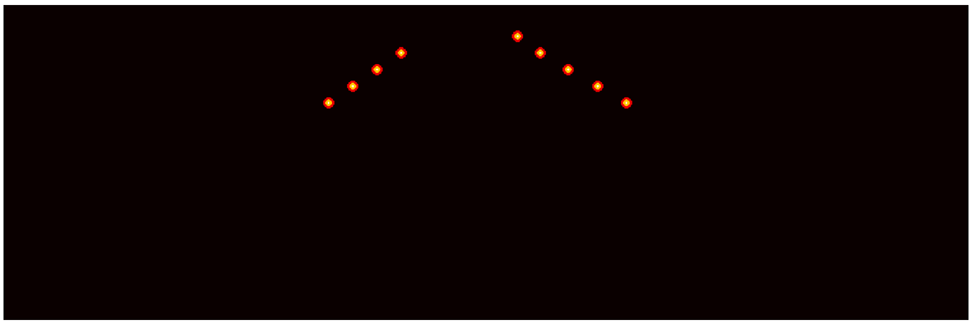


Figure 3.7: Merged ground truth heatmaps into one image

Chapter 4

Data preparation

In section 4.1, we describe the dataset we use for training the models. In section 4.2 we describe the annotation process and how we utilized a semiautomatic labeling tool to speed up the labeling process.

4.1 Dataset

There are two possible approaches how to detect the cones: one of them is to detect the bounding boxes of the cones, and the other is to semantically segment the cones. For these approaches, the publicly available FSOCO dataset can be used, which contains a large dataset with corresponding annotations [10]. The author of this thesis conducted research and found no papers that dealt with the segmentation of the race track of Formula Student Driverless competition in a similar spirit. Also, there is no publicly available annotated dataset for this problem. Due to the unavailability of the labeled dataset, we needed to create our own dataset and annotate it. Now we will describe the datasets that we collected.

4.1.1 eForceSeg dataset

The logging system of the eForce Driverless formula saves the RGB frames from races, so it is possible to retrieve those frames from logs and annotate those frames. The logger saves 9 frames per second during racing. For our purposes, we included every fifth frame from the log files. It was mainly because the frames collected at a similar time have very high similarity, hence will not provide significant information for the training of the network. The next reason is that labeling is a time-consuming process. Because we will later on use other datasets for semantic segmentation, for the distinction between these datasets, we will reference this dataset as the eForceSeg dataset. Overall, the eForceSeg dataset contains over 900 RGB images in resolution 1280×720. All images were taken from the ZED 2 camera mounted on the main hoop of the car.

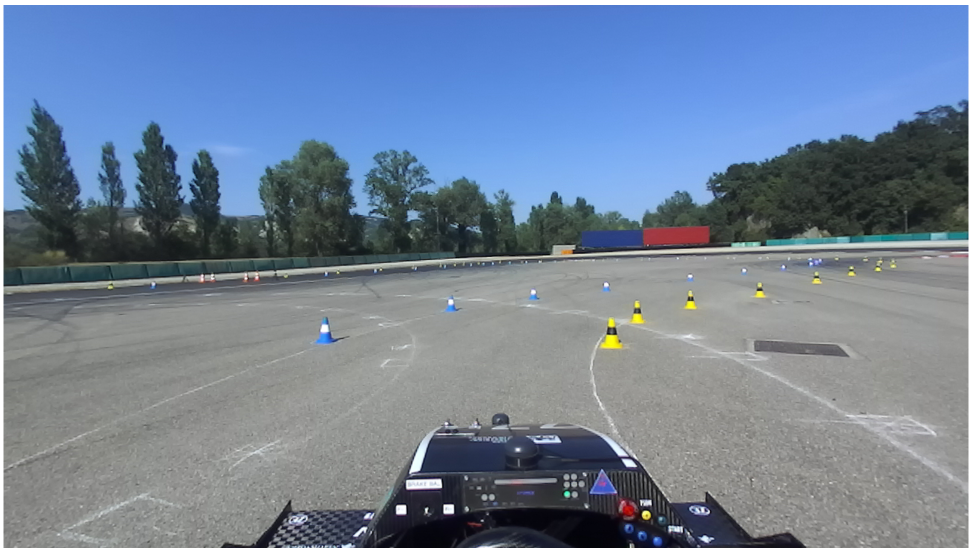


Figure 4.1: image from the eForceSeg dataset captured during Formula Student Auto Cross event in Italy

4.1.2 FSOCO dataset

FSOCO dataset (Formula Student Objects In Context) is a publicly available dataset of images containing cones from the Formula Student Driverless competition. Since annotating the images is a tedious task for newer teams, multiple Formula Student teams provided their dataset to help other teams to develop a perception system. The FSOCO dataset consists of 11 572 images with corresponding bounding boxes and 1517 images annotated with segmentation masks of traffic cones [10]. The majority of the images in the FSOCO dataset consists of images that do not contain a race track, so these images are not important for our task as we need only the images that contain the race track. The images are collected from different viewpoints. A typical image representing the FSOCO dataset can be seen in Figure 4.2 and 4.3. In these images, there are cones that do not represent the race track.

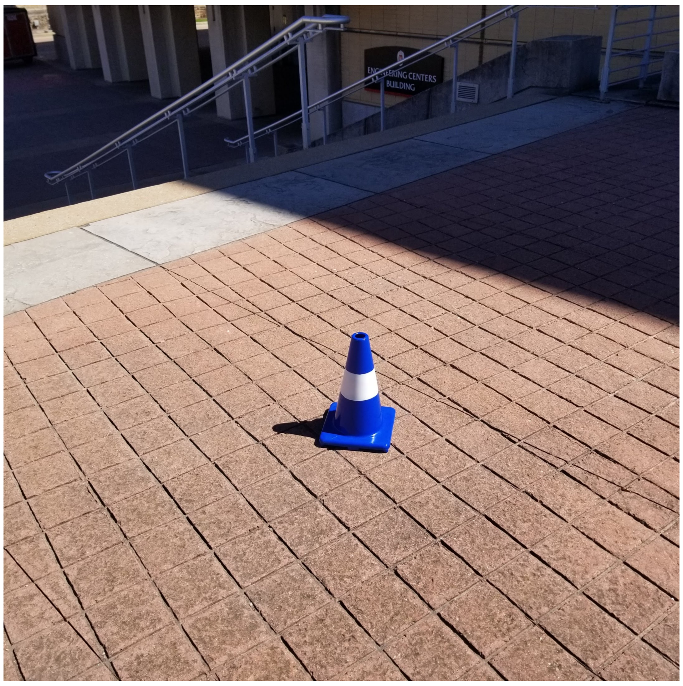


Figure 4.2: image from FSOCO dataset provided by Wisconsin team [10]

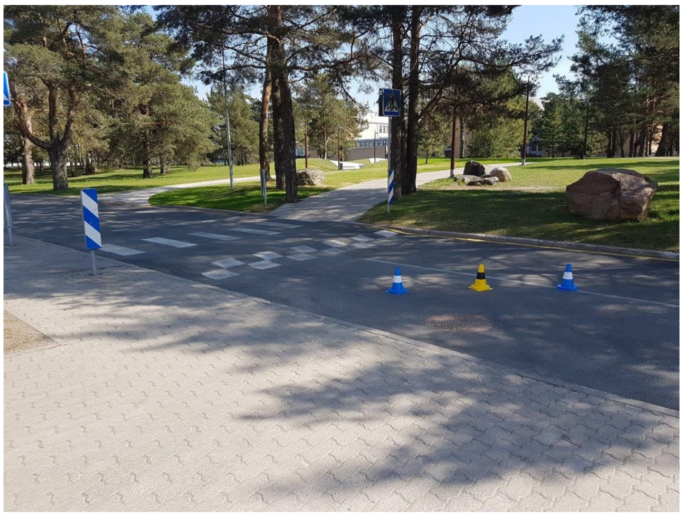


Figure 4.3: image from FSOCO dataset provided by Tallin team [10]

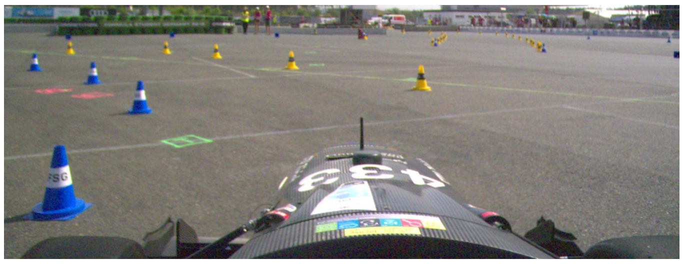


Figure 4.4: image from FSOCO dataset provided by AMZ team that form the race track [10]

After we filtered out useless images, we identified over 300 images that represent the race track and hence could be used.

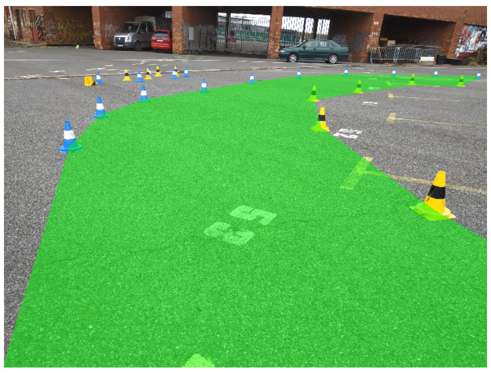
4.1.3 eForceSeg and FSOCO dataset distribution

The eForceSeg dataset comprises the images collected during the test days at the airfields in Milovice and Panenský Týnec, as well as during the race days that were held at the autodrome in Most and Varano de' Melegari in Italy. Since the dataset contains images from only four race tracks, ideally, we would like to enlarge the dataset so that the dataset contains images from many different places collected under different conditions. Based on the visual analysis of eForceSeg dataset, we have observed that there are not enough images capturing race turns. Also, the eForceSeg dataset does not contain many images with cones that are not a part of the track. That could potentially lead to mispredictions when we test our localizer on a more complex race track that contains cones that are randomly distributed near the track. More than 200 images from the eForceSeg dataset were gathered from the Acceleration event. These images are not very diverse, hence useful, since the race track is straight.

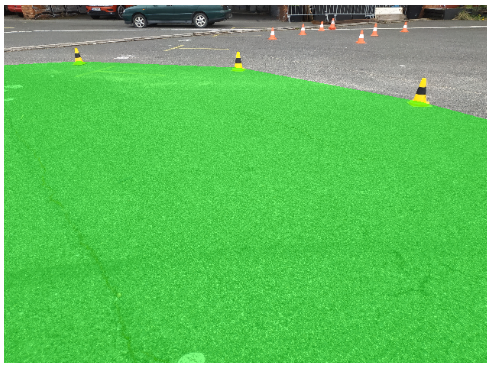
Despite the smaller size of the FSOCO dataset in comparison to our collected eForceSeg dataset, the FSOCO images are more diverse, contain images of various weather conditions and different mounting angles of the camera. Also, the FSOCO dataset contains more images on which the race track has randomly distributed cones nearby that are not part of the track. These images are useful, because the model needs to learn to filter out cones that are not part of the race track. Since the upper part of the images contains only the sky, we crop these parts.

4.1.4 Increasing image diversity by manual collection

During the winter of 2022 and spring of 2023, we were unable to collect images of the race track from the formula, because the car was not functional due to an accumulator failure. Since the cardinality of the eForceSeq dataset and the used part of the FSOCO dataset is still relatively small, to enhance the diversity of the dataset, we gathered and annotated more than 400 images. Those images were not taken from the camera mounted on the formula. Instead, we took some photographs using a mobile phone camera from a similar angle to the images captured by our formula. During the collection of the images, we specially focused on building race tracks of greater difficulty with randomly distributed cones of various colors near the race track. We also collected the images that contain turns to improve the accuracy of the model. Because the images we collected in this dataset contain race tracks that could potentially be challenging for our model, we added them to the training set. In Figure 4.5, we can see images collected in Strahov.



(a) : Image containing yellow and blue cones that are not part of the race track but are close to it

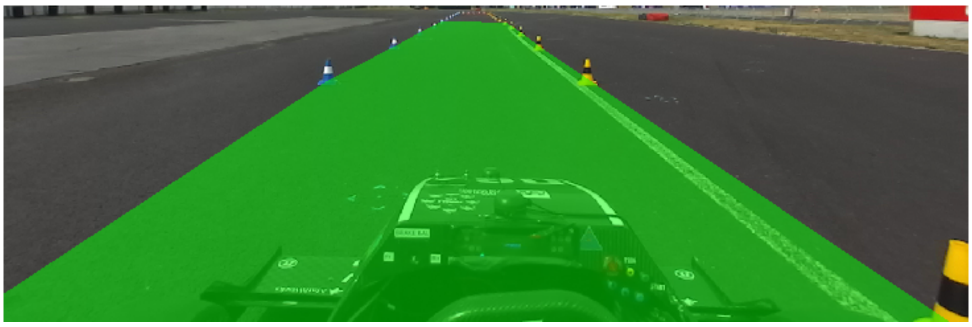


(b) : Image containing race turn delineated by yellow cones. In the distant part, there are small orange cones that are not part of the race track

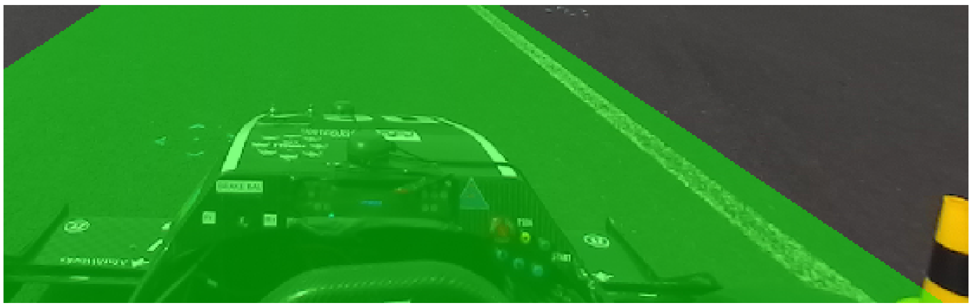
Figure 4.5: Images collected in Strahov

4.1.5 Data augmentation

Data augmentation is a regularization tool used to reduce overfitting. It consists of increasing the dataset size artificially by applying random transformations [22]. Because our dataset is relatively small, data augmentation could improve the accuracy of the model. Image transformations such as random cropping, horizontal flipping, and luminant transformations were tried. However, we did not use random cropping since there are cases when this method produces semantically incorrect masks as can be seen in Figure 4.6b, where the image was created by random cropping of image 4.6a. Even though image 4.6b contains no cones in the distance, the race track is annotated there, which is undesirable.



(a) : Image containing yellow and blue cones that are not part of the race track but are close to it



(b) : Image containing race turn delineated by yellow cones. In the distant part, there are small orange cones that are not part of the race track

Figure 4.6: Images collected in Strahov

To increase the number of images containing race turns, manual cropping of images was performed. Since the color of the cone is an important feature in the image for the segmentation of the race track, we did not apply horizontal transformations. The luminance transformations that simulate the weather changes were retained to obtain better results in different weather conditions.

4.1.6 Synthetic dataset

In order to obtain better accuracy of the model, it has been proposed to pre-train the network on a large synthesized dataset. As we stated before in section 3.2 about fine-tuning, we argue that the network could learn important features in initial layers such as cone shape and color of the cone that could be transferred and reused to help improve the accuracy of our model. Also, the pre-trained model could learn to filter out cones that do not form the race track.

The dataset that we use for pre-training the model was collected from the eForce Driverless simulator. The simulator was implemented by members of the eForce Driverless group. However, the author of this thesis did not contribute to it.

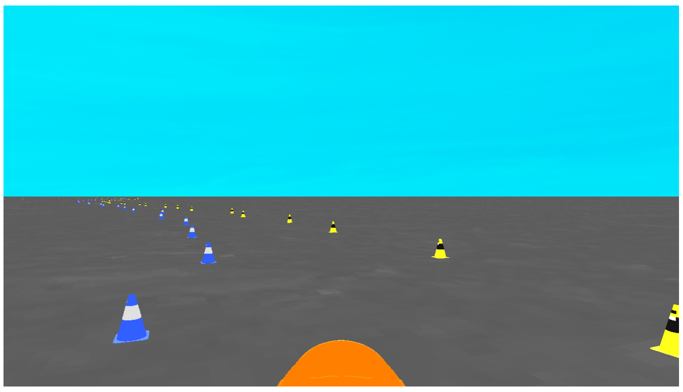


Figure 4.7: Screenshot taken from eForce driverless simulator

There are several advantages of having a driverless simulator.

1. Testing,
2. Safety,
3. Data generation.

The first advantage is the possibility to test the algorithms such as path planning before deploying them on the actual formula. This allows us to detect bugs efficiently, as the process is simulated. Effective testing of software is related to safety. Finally, with the simulator, it is possible to collect a large set of data that can then be used to train neural networks.

Collection of synthetic images

The simulator has two modes: a manual and an autonomous. Since we would like to automate the process of collecting synthetic images, we utilized the autonomous mode. We would like to have a diverse synthetic dataset where we have full control over the generation of the images. To collect various synthetic images, it is necessary to create various maps of the race track. For race track generation, we used the implementation of procedural race track generation available on GitHub, which is released under MIT license [23]. We slightly modified the implementation so that the script generates cones that form the race track and also generates the cones that are not related to the race track, so that the model learns to ignore the cones that do not form the race track.

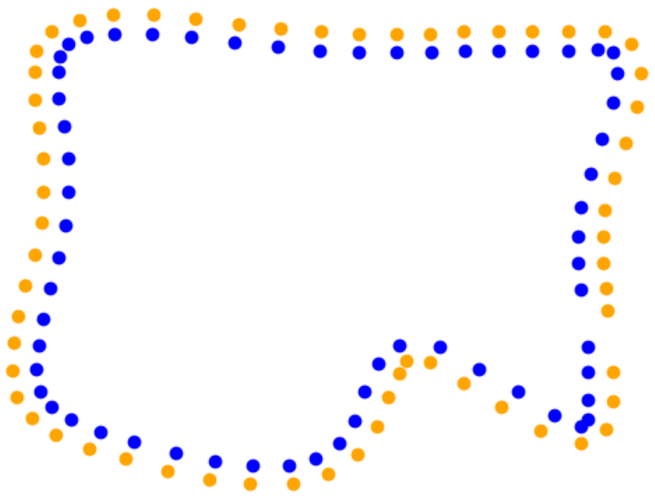


Figure 4.8: Generated race track

Then the image collection from eForce driverless simulator comes into play. The frames from racing were randomly captured and information such as car position and the position of visible cones are saved to be able to annotate the collected images fully automatically. With this method, we captured over 20,000 images.

4.2 Annotation process

This section is divided into two parts. First, we will describe the annotation process of the real-world images. Then, we will show how we annotated the synthetic images.

4.2.1 Annotation of real-world images

After we collected the images, the labeling process began. We started to annotate the images with the segmentation masks representing the inner part of the race track because first, we did not assume to use any other approach than prediction of segmentation mask. Later, for localization of the race track boundaries, the segmentation masks were manually converted into 2 labels: left boundary and right boundary of the race track. These labels are shown in Figure 3.2 with blue and yellow line. Here we will describe how we annotated images with the segmentation masks. We tried various segmentation tools such as Label Studio or the CVAT. Because of its user-friendly interface, we have chosen to annotate more than 100 images manually in the CVAT annotation cloud-based tool. We focused on parts of the race track in the image that are near the formula so even though there was a track defined 30 meters away from the formula that could be segmented, we did not annotate those parts for several reasons. First, the distant parts of the race track are sometimes difficult to recognize, since the cones in distant parts of the images are formed by only a few pixels. Secondly, there are some images in which the distant part of the track is ambiguous. The third reason is that distant parts of the race track are not important for a safe ride. Lastly, this approach enabled us to annotate the images faster. However, this process of annotating the images is slow, since we need to annotate the dataset manually. Ideally, we would like to use some semiautomatic annotation tool for labeling the images.

First, we utilized the implementation of the eForce cone detector and cone center estimator to detect cones on the images and obtain the centers of the cones. In our first version of the algorithm for semiautomatic annotations, we estimated the inner part of the race track heuristically, by comparing the distances of the cone centers that were obtained from cone detector directly from the camera image, so no transformations were applied.

As the CVAT annotation tool is an open source project released under the MIT license, we could implement this functionality into it. However, since CVAT is quite a large project and contains a lot of code, we searched for a smaller project due to simplicity and clarity. After some research, we found a Pangolin labeling tool written in PyQt5 [24]. It provides polygonal segmentation and bounding box labeling. The labeled data in the Pangolin tool can be exported in PNG, XML, and YoloV3 format. This graphical tool is also released under an MIT license, so we can implement our semiautomatic labeling functionality into it.

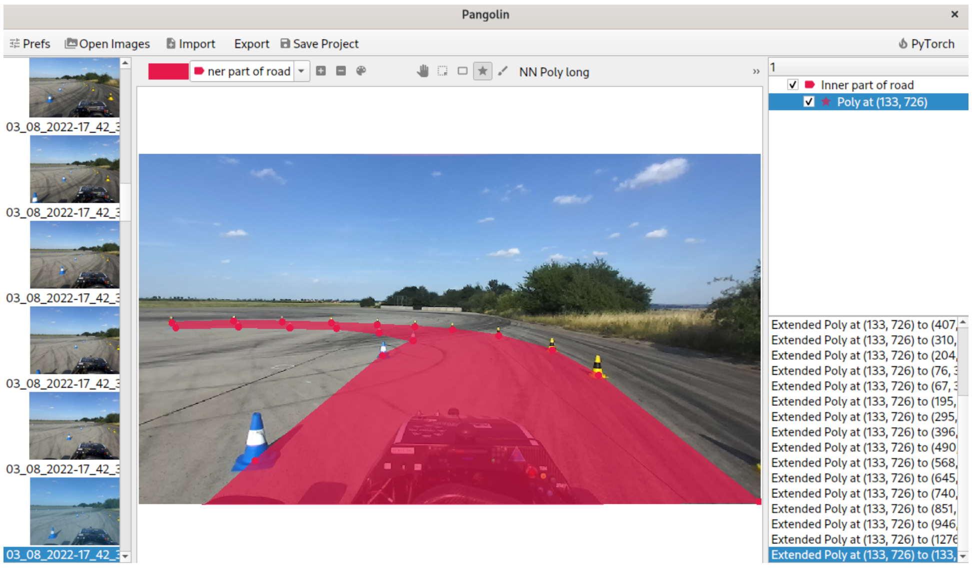

Pangolin
Prefs
Open Images
Import
Export
Save Project
PyTorch
ner part of road
NN Poly long
Inner part of road
Poly at (133, 726)
03_08_2022-17_42_3
Extended Poly at (133, 726) to (407,
Extended Poly at (133, 726) to (310,
Extended Poly at (133, 726) to (204,
Extended Poly at (133, 726) to (76, 3
Extended Poly at (133, 726) to (67, 3
Extended Poly at (133, 726) to (195,
Extended Poly at (133, 726) to (295,
Extended Poly at (133, 726) to (396,
Extended Poly at (133, 726) to (490
Extended Poly at (133, 726) to (568,
Extended Poly at (133, 726) to (645,
Extended Poly at (133, 726) to (740,
Extended Poly at (133, 726) to (851,
Extended Poly at (133, 726) to (946,
Extended Poly at (133, 726) to (1276
Extended Poly at (133, 726) to (133,

Figure 4.9: Pangolin annotation tool

However, sorting cones directly from bounding box positions does often produce incorrect results. If the image contains cones that do not form the race track, this approach will fail as this heuristic approach does not occlude any bounding boxes that are not part of the track. The distance between cones directly from the image is not a good metric for sorting since the task of searching for the nearest cone in the image is not equivalent to the task of searching the nearest cone from the bird's eye.

The rest of the eForceSeg dataset, that is, the next 800 images, was annotated with this approach and incorrectly annotated masks were manually corrected. For annotation of the part of the FSOCO dataset were used the ground-truth bounding boxes annotations. We heuristically created the polygonal shape representing the race track from those annotations. Now, we will describe our method to increase the efficiency of semiautomatic labeling.

Integrating the path planning algorithm in order to improve semiautomatic labeling efficiency

The second version of the semiautomatic annotation tool enhances the labeling the race track by utilizing a path-planning algorithm to obtain a polygonal segmentation mask. In this approach instead of using distances of cone centers taken directly from images, we projected the center of cones to local car coordinates and used a path-planning algorithm from which we can obtain the borders of the track by iteratively finding the closest cones in the scene. It is important to point out that the path-planning algorithm is not the author's work. At this point, the semiautomatic labeling tool utilizes the detections from the cone detector from which the cone centers are estimated. The centers are then projected with homography mapping onto local car coordinates. After obtaining the transformed centers of the cones, the algorithm for planning the path can give us the boundaries of the track, from which we can easily create a polygon.

This approach produces significantly better results as the planning algorithm has the capability of filtering out cone centers that are not part of the track most of the time. Still, some labels needed to be manually corrected, when the path planning algorithm fails or the cone detector is unable to detect some cones. However, with this approach, labeling is many times faster. With this method, it is a lot easier to collect and get corresponding ground truth labels from the images. We used this method later on for annotating the test dataset. Now that this tool is available, it will be much easier for us to annotate the data we collect in the future.

4.2.2 Annotation of synthetic images

As we previously mentioned in subsection 4.1.6 about the synthetic dataset, the frames from racing were randomly collected during simulations, and the information such as the position of the car in the simulator and the position of the visible cones were saved into JSON. From those saved data is then possible to annotate the images fully automatically with searching for correspondence between ground truth race track map and position of visible cones on the image. With this algorithm, we captured over 20,000 images with corresponding annotations.

Chapter 5

Implementation part

In this section, we will describe the training process, the search for hyperparameters, and the choice of appropriate loss function for each approach. We trained three different models:

1. UNet model predicting segmentation mask,
2. ResNet model predicting coordinates of the race track boundaries,
3. ResNet model predicting heatmaps, from which we can obtain the race track boundaries.

First, in section 5.1 we will describe the implementation parts that are common for each training process.

5.1 Common part

The dataset for training and validation is split randomly into training (90%) and validation (10%) subsets. Except for the transformations we described in the data augmentation section, no other transformations were used. This ensures that more attention is given to the closest parts of the track with respect to the formula and the more distant track section is not so important. In our code, we utilized Weights & Biases library for logging and visualization

[25]. The main advantage of WandB is that it is possible to create new graphs from logged values and interactively view them. One of many tools that W&B provides and we utilized it is Sweeps. With this tool, it is possible to automatically try different hyperparameter values which are chosen from a predefined JSON configuration. The training was performed at the Research Center for Informatics clusters at CTU Prague on the graphics card Tesla V100-SXM2-32GB.

5.2 UNet training

For training, we modified an implementation of U-Net on GitHub that was implemented for the Carvana car masking challenge [26] The code is using the PyTorch framework and is provided under a GPL3.0 license. The dataset used for training and validation consists of eForceSeg, FSOCO, and Strahov datasets. The eForceSeg dataset, after cropping the part of the images that contain the sky, has a resolution of 1280×720. However, the FSOCO and Strahov dataset consists of many images of different resolutions (1920×700, 1920×1080, 1000×750). Since we would like to use a mini-batch size bigger than one, after visual analysis, we decided to rescale all images to the resolution 512×384 with bicubic interpolation, which preserves the shape of cones so it is still easy to segment the track.

To speed up training, we scaled the image to 0.5 of the resolution so the shape of matrix that is fed into the netTwork is (256, 192). With this shape, the max batch size due to memory limits is 32.

The mispredictions are penalized by binary crossEntropy with logits. The difference between BCE with logis and the BCE loss function is that BCE with logits implicitly modifies the data with a sigmoid layer. The BCE loss function penalizes the highly confident mispredictions and also less confident correct predictions. The loss function is computed as follows:

$$\mathcal{L}(y_{i,j}, \hat{y}_{i,j}) = -\left[y_{i,j} \cdot \log(\hat{y}_{i,j}) + (1 - y_{i,j}) \cdot \log(1 - \hat{y}_{i,j})\right]$$

The variable $y_{i,j}$ denotes the class of the true label for the pixel at coordinates i, j. The variable $\hat{y}_{i,j}$ denotes the predicted probability for the positive class of pixel at coordinates i, j [27].

However, we tried more loss functions like the Negative log-likelihood loss function performed with similar accuracy and convergence rate. Also, the Mean squared error was tried. It performed similarly in accuracy but had slower convergence.

For evaluating the validation accuracy of our model, the Validation dice score is used, since it was already implemented in the template code. The dice score is computed as follows, where A represents the predicted region and B represents the true region [28]:

$$\mathrm{Dice}(A, B) = \frac{2|A \cdot B|}{|A| + |B|}$$

Later for evaluating the test dataset, we used Intersection over Union metric. Here, the A and B represent the same regions as in dice score [28].

$$\mathrm{IoU}(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{5.1}$$

We did an experiment to compare Adam, Adagrad, Adadelta, RMSprop, and Stochastic gradient descent optimizers. The default values of arguments for optimizers were used. The batch size used in this experiment was 16 and training was set to 40 epochs. From those optimizers, the SGD performed the worst with a 0,37 validation dice score. The other optimizers had similar validation dice scores - over 0,95 and since Adam had a dice score of $0,96$, we chose to use he Adam optimizer in all of our experiments.

5.3 ResNet training

In this section, we will first describe how we trained the ResNet for predicting the coordinates of the race track boundaries. Then, we will describe the training of ResNet where we predicted heatmaps instead of the coordinates.

5.3.1 Prediction of coordinates

Initially only the eForceSeg dataset was used for training and evaluation, since we needed to modify the labels from polygonal masks to labels representing the left boundary of the race track and the right boundary of the race track. Our dataset for this task contained 850 images and if the network performs well enough, we would also label the rest of the images from FSOCO and Strahov dataset. For training, we used ResNet 50 model imported directly from PyTorch and at the end we modified the fully connected layer so the output layer's shape is 40 (predicting 20 points for the left boundary and

20 points for the right boundary). First we tackled issues with training the network, because it performed poorly and it was even difficult to overfit on the training dataset. This changed significantly when we modified the Adaptive average pool layer size. Initially, the adaptive average pool layer produced 1×1 output size for each channel. We modified the architecture, so the Adaptive average pool layer produce 7×7 output tensor for each channel. With this modified architecture, the value of validation loss (MSE) for input images of size 1280×420 was in average 1352.6 after 10 epochs. Before this modification, the value of validation loss was in average 5431.8. We trained the model for 70 epochs with a batch size of 16. The mean squared error loss function was used for the penalization of incorrectly predicted points. In the formula for MSE, the x_i denotes the predicted value. The y_i denotes the ground truth value [27].

$$L = \sum_{i=1}^{40} (x_i - y_i)^2$$

5.3.2 Heatmaps prediction

The architecture of ResNet was again slightly modified from the original imported version. Since the output layer predicts a tensor of shape 40×(size of one heatmap), the number of predicted points is bigger, thus the memory requirements are higher so it is not possible to use ResNet 50. Instead, we used ResNet 18 with an Adaptive Average Pooling layer that produces 5×5 tensor for each channel. The training process of ResNet Heatmaps is similar to the training process of ResNet we described in the previous section, so we will not describe them again.

We implemented scaling of the tensor so the output layer produces tensor of shape (40, (1280 / scaling factor)) instead of predicting tensor of shape (40, 1280). The borders of the race track are then reconstructed from *argmax* of every one-dimensional heatmap.

Chapter 6

Results

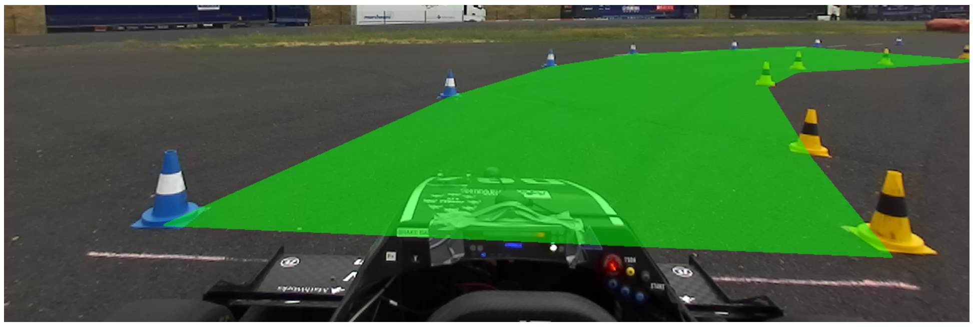
In this Chapter, we present the results of models trained on UNet and ResNet architectures. The evaluation is performed on real-world datasets and also on a synthetic datasets collected from the eForce simulator.

6.1 Test dataset

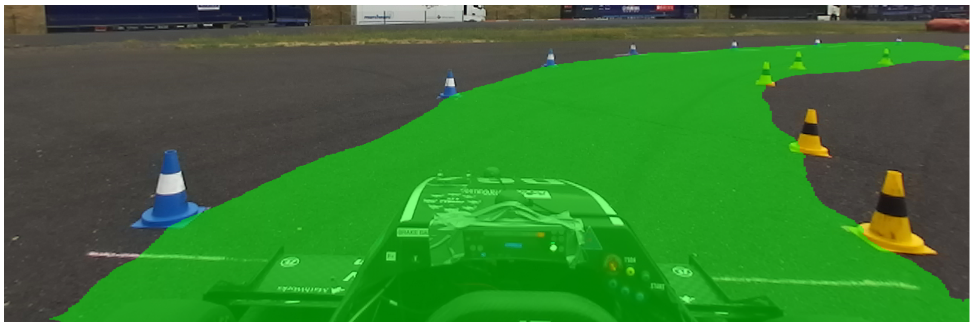
The real-world test dataset was created from 3 races: Formula Student Autocross event from the Czech Republic in Autodrom Most (208 images), Autocross event from FSG (118 images), and from the testing day in the airport in Milovice (21 images). The test dataset differs from the training and validation datasets. The images from the test set contain different weather conditions and the race tracks that were not included in the training and validation set. The synthetic dataset (336 images) was collected from variously generated race tracks in the eForce driverless simulator.

6.2 Measuring accuracy of predicted tracks

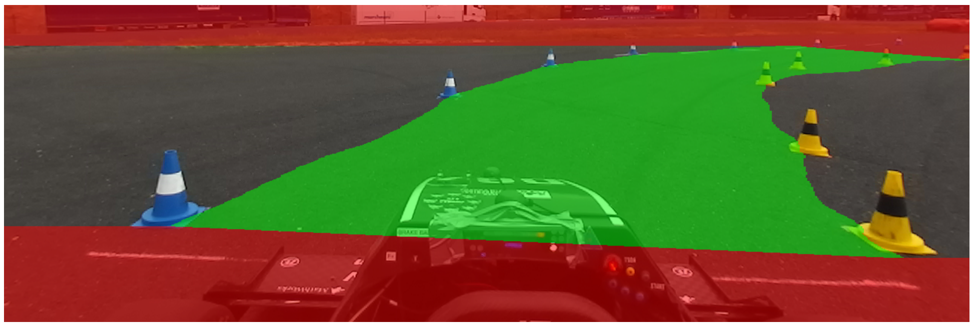
The output from our models is or can be modified into a segmented inner part of the race track that has a polygonal shape. In Figure 6.1b, we can see a raw segmented map produced by UNet model.



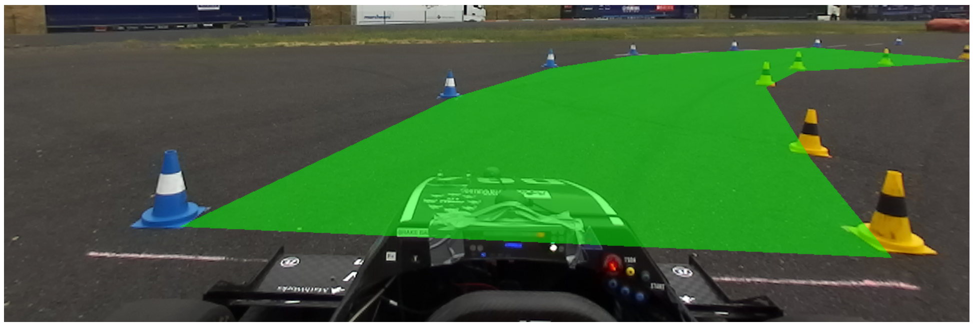
(a) : Ground-truth annotated mask



(b) : Raw segmentation mask produced by UNet



(c) : Postprocessed segmentation mask produced by UNet model



(d) : Mask produced by baseline approach

Figure 6.1: Comparison of baseline approach and UNet segmentation

Since the path-planning algorithm starts forming a left boundary of the track from the first blue cone and a right boundary from the first yellow cone, the comparison between the UNet mask and mask created from the planning algorithm would lead to incorrectly classified pixels at the bottom of the image. These pixels are marked with the red color in the bottom in Figure 6.1c. So we decided to ignore those bottom pixels as they do not matter when navigating through the race track. The ground truth annotated masks were annotated with the semiautomatic labeling tool and inaccuracies were manually fixed. In the following experiments, we want to focus on the accuracy of the nearest parts of the race tracks, as these are most important for navigating through the track, so not only the bottom pixels were excluded, but also pixels representing more distant parts of track were occluded from comparison.

In the synthetic dataset, we compare the path-planning algorithm without vision, since we do not have an available cone detection model for the synthetic dataset. Due to the fast fully automated annotation process, we will compare the race track up to 40 meters.

For evaluation, we used IoU. Another approach would be to compare the ground truth centerlines of the race track with the centerlines created by our methods. It is possible to extract the centerlines for methods that predict boundary of the race track by, however we did not find a reliable method for extracting centerlines of the track from segmentation mask due to complex shapes of the mask representing inner part of the race track, thus the impossibility to clearly identify the left boundary and the right boundary of the track. Also, it is difficult to annotate ground-truth centerline of the race track accurately. Due to these reasons, we decided to measure the accuracy with IoU.

6.3 Chosen models

As the final models we use for the evaluation of the test dataset, we have chosen the models with the best validation score. To clarify, all fine-tuned models were pre-trained on synthetic images from the simulator. For fine-tuned UNet models, we experimentally tried to freeze different layers and measured the validation accuracy.

The best accuracy have models that have no frozen layers and the worst accuracy have models with the most frozen layers. The experiment was performed also on fine-tuned ResNet model. The fine-tuned ResNet model used in the following experiments has 9 frozen layers.

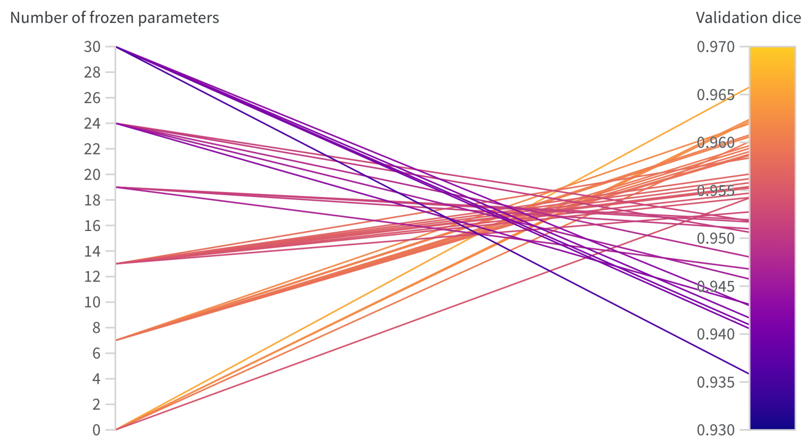


Figure 6.2: Dependency of number of frozen layers on validation dice score - UNet

6.4 Results

We will now present three tables showing the evaluation results. The metric used for evaluation is IoU. Table 6.1 shows a comparison of the methods on the synthetic dataset.

Method	IoU Accuracy (%)
Only planning	**94.3**
ResNet	87.7
UNet	90.4
ResNet - Heatmaps	75.8

Table 6.1: Comparison of baseline and our proposed methods on the synthetic dataset

As can be seen, the planning algorithm performs the best in this experiment [15]. We argue that the reason for this is that we measured the accuracy up to 40 meters, so generally it was more difficult for UNet and ResNet approaches to recognize the race track in more distant parts since the cones from distant parts are represented just by a few pixels. In contrast, the planning algorithm did not need to recognize the cones, as the detections were simulated, and all available cone positions were forwarded to the planning algorithm.

The next two tables show the evaluation accuracy of the real-world dataset. We split the test dataset and measured the accuracy of frames taken from

Formula Student Czech Autocross separately from the other frames since the frames taken from the FS Czech race were part of the training subset for ResNet and Heatmap ResNet. Thus, Table 6.2 shows the evaluation accuracy only for the current approach and for UNet.

Method	**IoU Accuracy (%)**
Cone detections + planning	91.2
ResNet	59.8
UNet	**92.4**
Fine-tuned ResNet	61.5
Fine-tuned UNet	92.3
Heatmaps ResNet	58.5

Table 6.2: Comparison of baseline and our proposed methods on the Real World Test Dataset (FSG + Milovice)

The UNet shows promising results on the real-world dataset and is slightly better than the current approach. The fine-tuned UNet shows similar results to UNet without fine-tuning, so the fine-tuning did not enhance the accuracy of the model which was expected from the results in Figure 6.2

Despite the ResNet model showing promising results on synthetic dataset, the model performs poorly on the real-world dataset. In the future, we could try another architecture for training instead of ResNet. In comparison the fine-tuned ResNet shows marginal improvement upon the ResNet without fine-tuning, so the fine-tuning had a small effect on the accuracy. The heatmap approach performs the worst of all the compared methods.

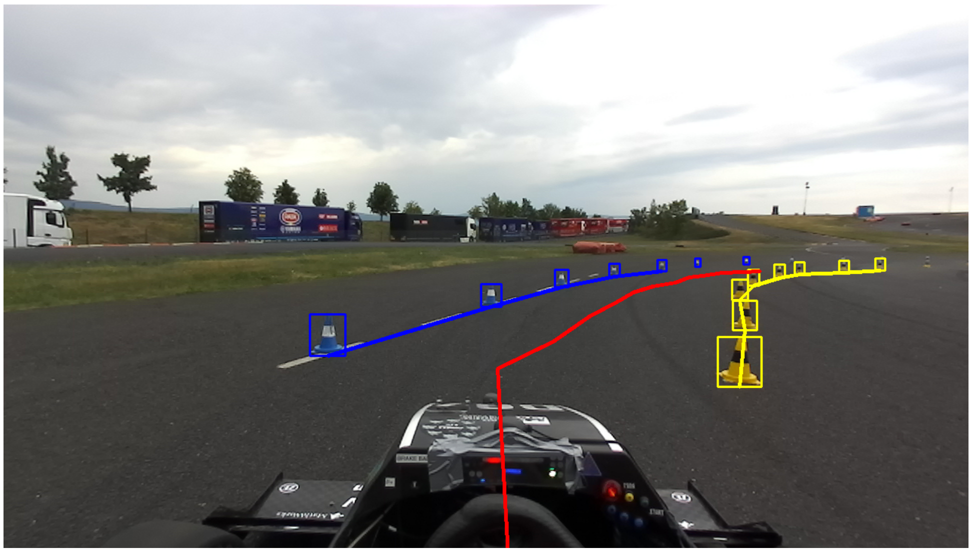
Now we will discuss the accuracy of the FS Czech Autocross test dataset.

Method	**IoU Accuracy (%)**
Cone detections + planning	75.8
UNet	**93.6**
Fine-tuned UNet	93.4

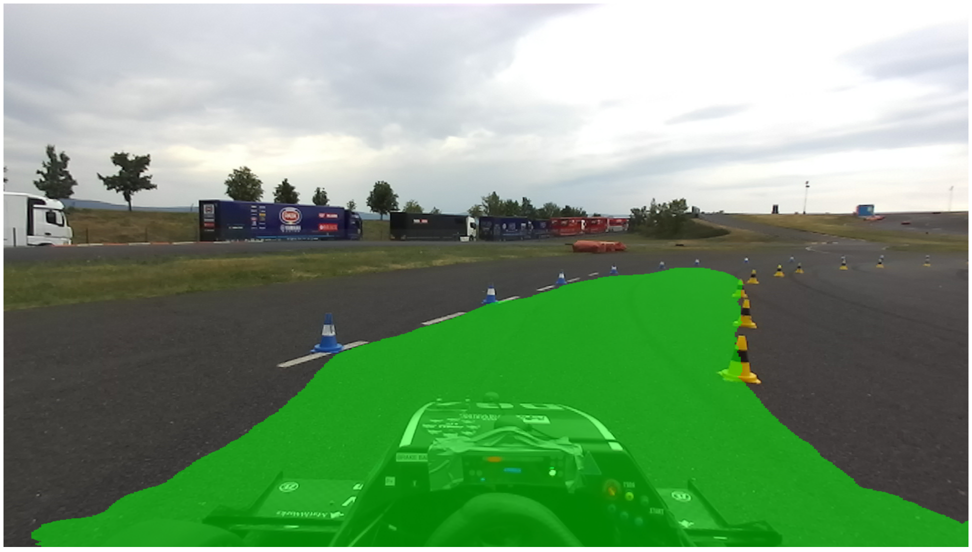
Table 6.3: Comparison of baseline and our proposed methods on the Real World Test Dataset (FS Czech)

On this dataset, the UNet model outperforms the baseline approach significantly. Again, fine-tuned UNet shows similar results to UNet without prertaining. The reason for the poor performance the of the current approach is that the cone detector did not predict some cones when the camera was oriented towards the sun. Also, the camera was not properly calibrated and since there were cones on the image that did not represent the current race track, the path planning algorithm could not correctly predict the race track. Since only the UNet model performed accurately enough, we provide an FPS benchmark only for UNet. The experiment consisted of processing of 1382 images again on RCI cluster on a graphics card Tesla V100-SXM2-32GB. The Average processing time is 58.033 FPS.

6.4.1 Visual comparison UNet vs Current approach

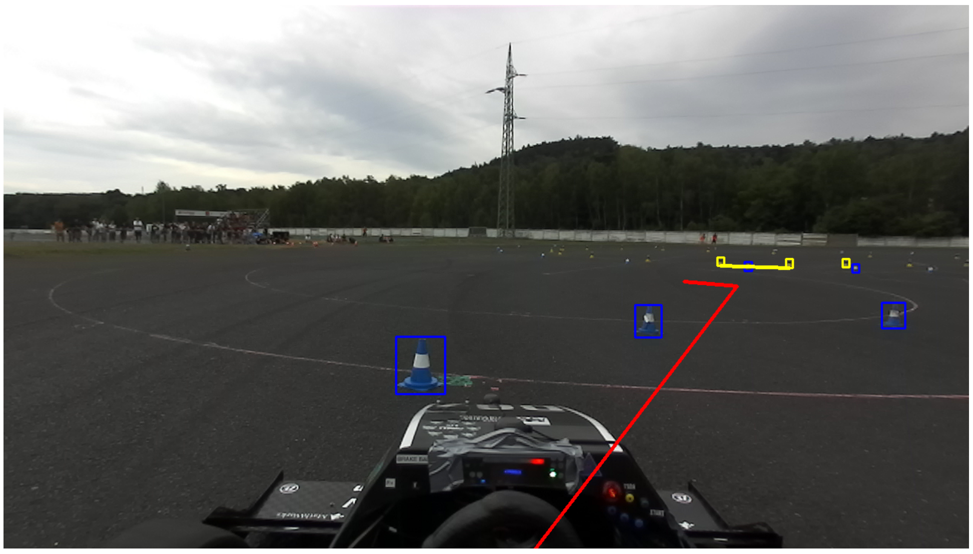


(a) : Baseline approach

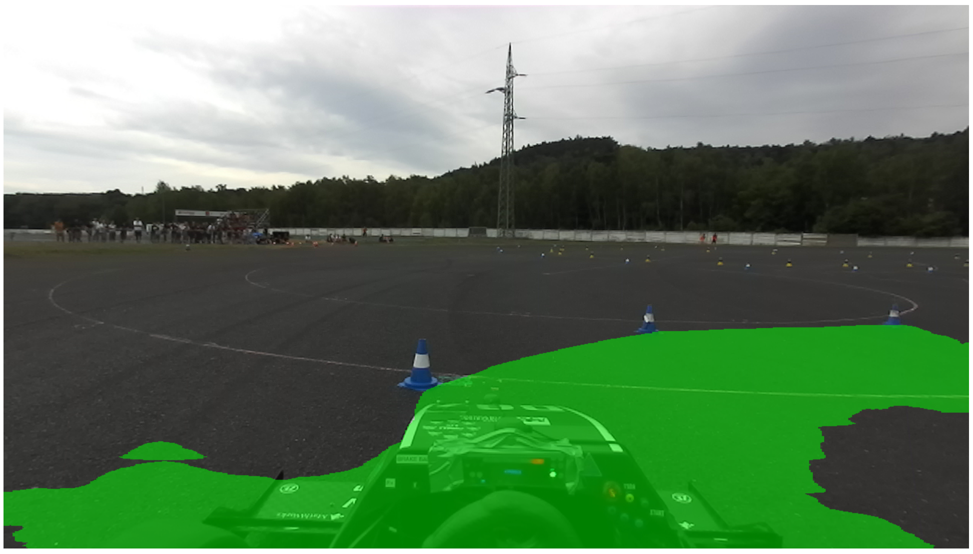


(b) : UNet segmentation

Figure 6.5: Comparison of baseline approach and UNet segmentation

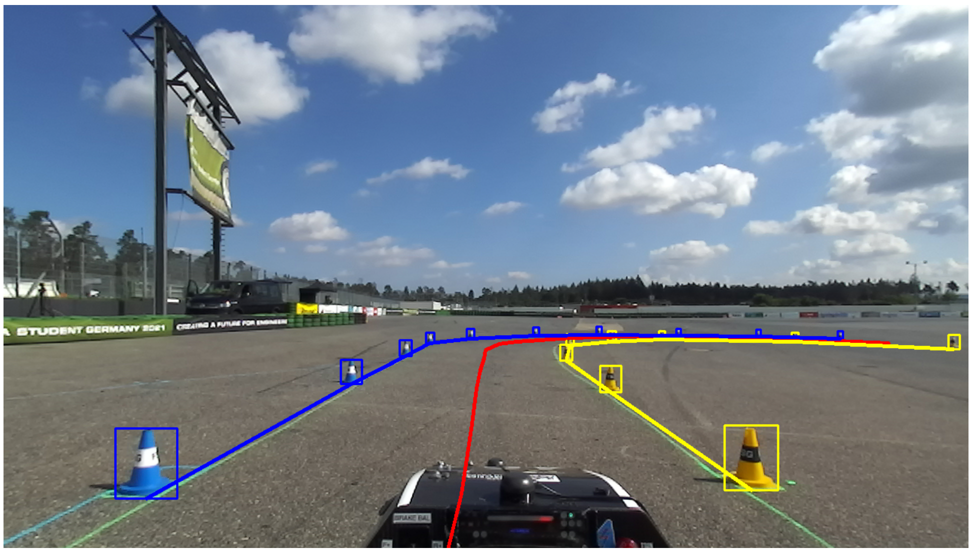


(a) : Baseline approach

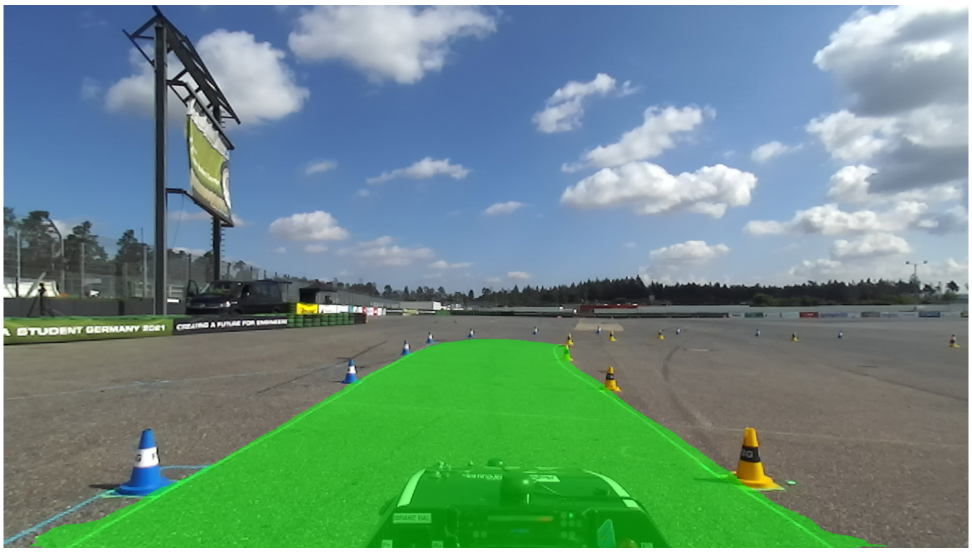


(b) : UNet segmentation

Figure 6.6: Comparison of baseline approach and UNet segmentation

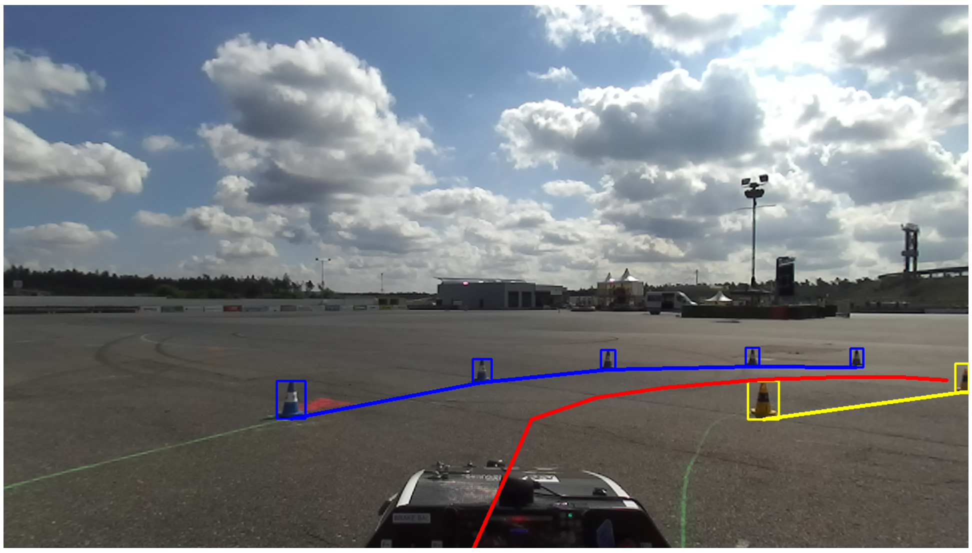


(a) : Baseline approach

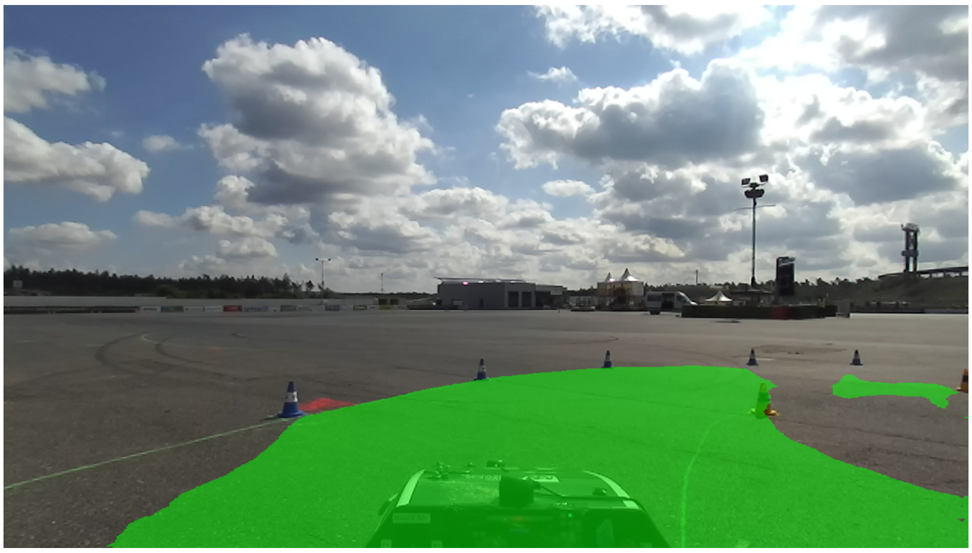


(b) : UNet segmentation

Figure 6.7: Comparison of baseline approach and UNet segmentation

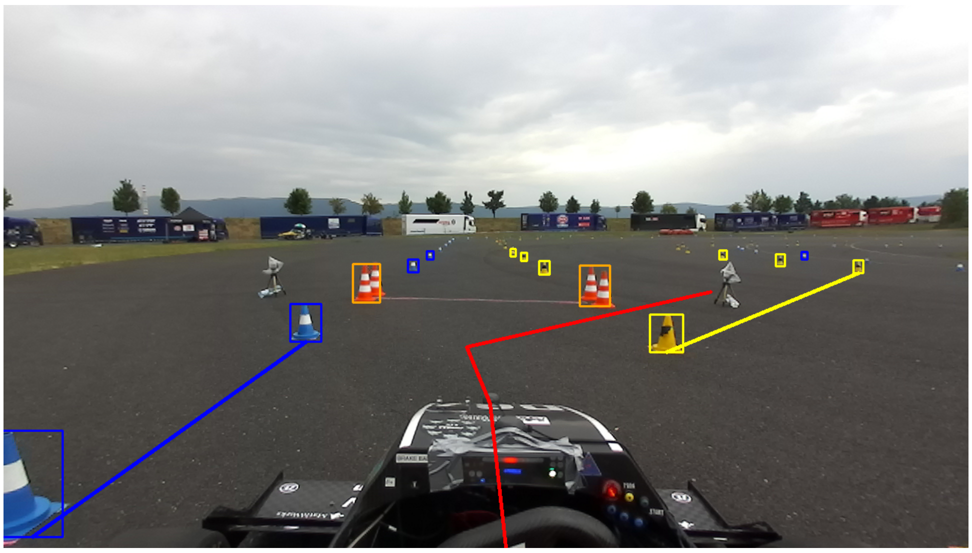


(a) : Baseline approach

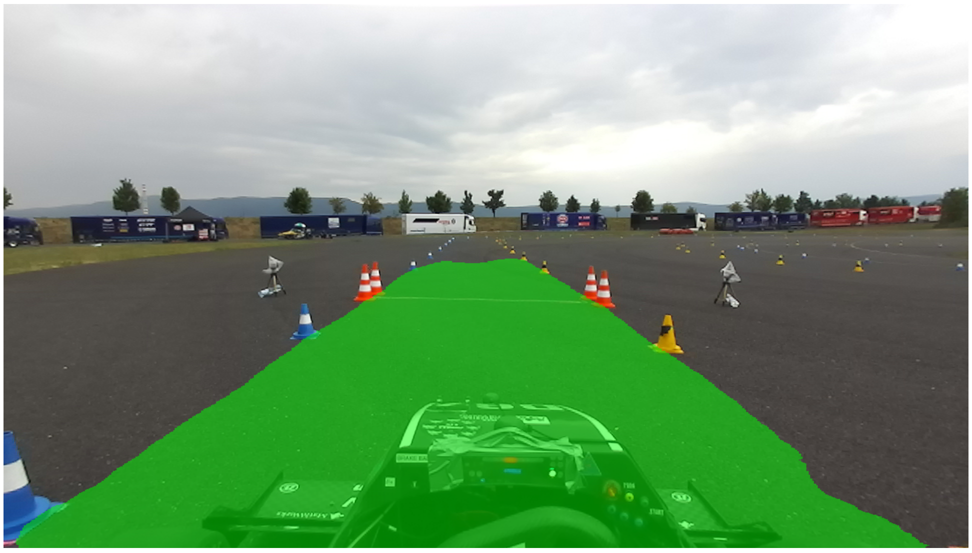


(b) : UNet segmentation

Figure 6.8: Comparison of baseline approach and UNet segmentation

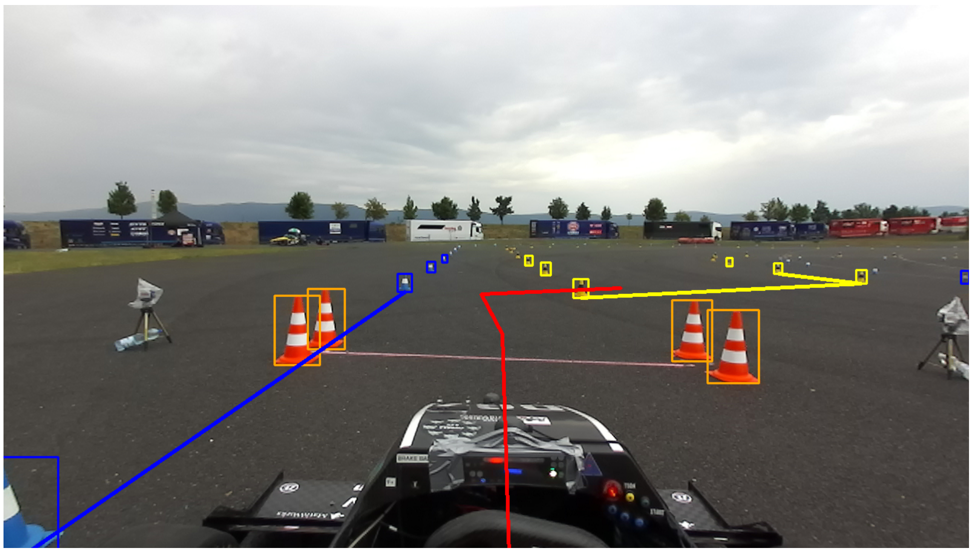


(a) : Baseline approach

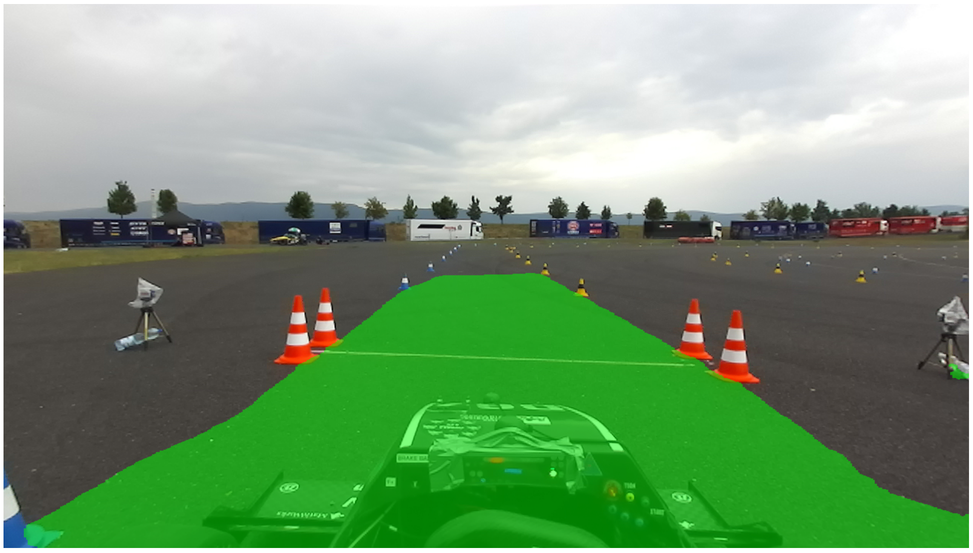


(b) : UNet segmentation

Figure 6.3: Comparison of baseline approach and UNet segmentation



(a) : Baseline approach



(b) : UNet segmentation

Figure 6.4: Comparison of baseline approach and UNet segmentation

Chapter 7

Conclusion

In this thesis, we presented the localizer of the race track that segments the track into inner and outer parts. U-Net-based localizer accuracy outperforms the method that is currently used in our formula. The UNet race track localizer average IoU on test dataset is 0.93 and the model runs at 59 FPS. It has been shown with illustrations that in cases where the baseline approach fails to localize the race track, our localizer segments the track correctly. Unfortunately, the localization of the race track via ResNet and ResNet heatmap approach is not accurate enough to be used in future.

The future work will include adding labels for the left boundary of the race track and the right boundary of the race track to the ground truth segmentation masks so we could easily create a centerline for navigating the formula as it is not possible accurately just with a polygon mask representing the inner part of the track. Also, we would like to collect a bigger dataset in the future to enhance the accuracy of predictions. It will be easier with our semiautomatic labeling tool. Finally, we would like to deploy and test the UNet model in our formula in the future.

Appendix A

Bibliography

[1] Miroslav Martínek. eforce dv.01 formula, 2022. non-publicly available image.

[2] Formula student teams. http://uwbracing.cz/wp-content/gallery/formula-student-czech-republic-2018/DSC3685-Edit.jpg. Online; accessed May 08, 2023.

[3] Formula Student Germany. Acceleration event schema. https://www.formulastudent.de/fileadmin/user_upload/all/2023/important_docs/FSG23_Competition_Handbook_v1.0.pdf, 2022. Online; accessed May 05, 2023.

[4] Formula Student Germany. Skidpad event schema. https://www.formulastudent.de/fileadmin/user_upload/all/2023/important_docs/FSG23_Competition_Handbook_v1.0.pdf, 2022. Online; accessed May 05, 2023.

[5] Zhao. Result of semantic segmentation. https://thegradient.pub/content/images/size/w800/2018/05/Zurich-Cityscapes.png. Online; accessed May 09, 2023.

[6] Unet architecture. https://miro.medium.com/v2/resize:fit:720/format:webp/1*f7YOaE4TWubwaFF7Z1fzNw.png, 2013. Online; accessed May 09, 2023.

[7] 34 layer resnet architecture. Online; accessed May 16, 2023.

[8] Residual block with identity mapping, 2013. Online; accessed May 16, 2023.

[9] Elte Hupkes. Heatmap example, 2021. Online; accessed May 13, 2023.

[10] Niclas Vödisch, David Dodel, and Michael Schötz. Fsoco: The formula student objects in context dataset. *SAE International Journal of Connected and Automated Vehicles*, 2022.

[11] Formula Student Germany. Formula student germany 2023 competition handbook. `https://www.formulastudent.de/fileadmin/user_upload/all/2023/important_docs/FSG23_Competition_Handbook_v1.0.pdf`, 2023.

[12] Roman Šíp. Visual detection of traffic cones for autonomous student formula. `https://dspace.cvut.cz/bitstream/handle/10467/101636/F3-BP-2022-Sip-Roman-main.pdf?sequence=-1&isAllowed=y`, 2022. Bachelor's thesis, Czech Technical University in Prague.

[13] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015.

[14] Michal Horáček. Finding the fastest trajectory for autonomous student formula. `https://dspace.cvut.cz/bitstream/handle/10467/101617/F3-BP-2022-Horacek-Michal-final_thesis.pdf`, 2022. Bachelor's thesis, Czech Technical University in Prague.

[15] Dmytro Khursenko. Algorithm to complete the first lap for autonomous student formula, 2023. Bachelor's thesis, Czech Technical University. To be defended.

[16] Gabriel L Oliveira, Wolfram Burgard, and Thomas Brox. Efficient deep models for monocular road segmentation. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016.

[17] Semantic segmentation models. `https://paperswithcode.com/methods/category/segmentation-models`. Online, accessed May 16 2023.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer International Publishing, 2015.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[20] Andrew Glassner. *Deep Learning: A Visual Approach*. No Starch Press, 2021.

[21] Zhengxiong Luo, Zhicheng Wang, Yan Huang, Liang Wang, Tieniu Tan, and Erjin Zhou. Rethinking the heatmap regression for bottom-up human pose estimation, June 2021.

[22] Tensorflow tutorials: Data augmentation. https://www.tensorflow.org/tutorials/images/data_augmentation. Online; Accessed May 18, 2023.

[23] juangallostra. procedural-tracks github repository, 2019.

[24] edluffy. pangolin. https://github.com/edluffy/pangolin. Accessed: May 21, 2023.

[25] Weights Biases. Weights Biases. https://docs.wandb.ai/guides. Online; accessed May 21, 2023.

[26] Milesial. Pytorch-unet. https://github.com/milesial/Pytorch-UNet.

[27] Deval Shah. The essential guide to pytorch loss functions. https://www.v7labs.com/blog/pytorch-loss-functions, 2022. Online; accessed May 23, 2023.

[28] Nghi Huynh. Understanding evaluation metrics in medical image segmentation. *Medium*. Online; accessed May 16, 2023.

Appendix B

Abbreviations

- **LiDAR** - Light Detection and Ranging
- **YOLO** - You Only Look Once
- **SLAM** - Simultaneous Localization and Mapping
- **FEE CVUT** - Faculty of Electrical Engineering, Czech Techical Unverisity
- **RGB** - Red Green Blue
- **DV.01** - eForce driverless 01
- **FSE.12** - eForce formula student electric 12
- **ZED** - Stereolabs ZED (camera model)
- **YOLOV3** - You Only Look Once Version 3
- **CNNs** - Convolutional Neural Networks
- **ReLU** - Rectified Linear Unit
- **ResNet** - Residual Network
- **FSOCO** - Formula Student Objects in Context
- **CVAT** - Computer Vision Annotation Tool
- **PNG** - Portable Network Graphics
- **XML** - Extensible Markup Language

- **MIT** - Massachusetts Institute of Technology
- **JSON** - JavaScript Object Notation
- **CTU** - Czech Technical University
- **WandB** - Weights & Biases
- **GPL3.0** - General Public License version 3.0
- **BCE** - Binary CrossEntropy
- **MSE** - Mean Squared Error
- **IoU** - Intersection over Union
- **SGD** - Stochastic Gradient Descent
- **Adagrad** - Adaptive gradient
- **RMSprop** - Root Mean Square propagation
- **FS** - Formula Student
- **RCI** - Research Center for Informatics

Appendix C
Code

The available code can be found in this Gitlab repository `https://gitlab.fel.cvut.cz/capurjos/semantic_segmentation_of_race_track`