DeepField — Semi-Supervised Learning for RoboCup Soccer Field Recognition

Gutsche, Jan
Fachbereich Informatik, TAMS
Universität Hamburg
22527 Hamburg, Germany
7gutsche@informatik.uni-hamburg.de

Vahl, Florian
Fachbereich Informatik, TAMS
Universität Hamburg
22527 Hamburg, Germany
7vahl@informatik.uni-hamburg.de

***Abstract*— In the RoboCup Humanoid Soccer League, the detection of the soccer field as part of the robot's visual perception is critical i.e. to distinguish between the relevant field area and the undefined surrounding area. We propose a semi-supervised, data-driven, deep learning approach to solve this challenge. The goal is to get a system, that is more robust against natural lighting and needs less configuration compared to the currently popular model-based approaches while keeping the annotation effort low. For this task, several neural network architectures have been trained and evaluated.**

***Index Terms*— Semantic Segmentation, Neural Networks, RoboCup**

I. Introduction
WRITTEN BY JAN GUTSCHE

The visual perception system of autonomous robots is still an ongoing field of research [4], [7], [9]. The RoboCup competition started around the early 1990s to promote and motivate research and development of autonomous mobile robotics and artificial intelligence (AI) [19]. During a game of RoboCup soccer, it is critical for the autonomous mobile robot to sense its environment and furthermore distinguish between the soccer field area and the surroundings. Therefore detecting the boundary between these areas (afterwards called field boundary) is necessary. As this boundary can be interrupted by obstacles on the field, we often use the convex field boundary, which is the convex hull over the interrupted field boundary. A field boundary can be converted into and derived from a binary image mask representation with pixels below or above the boundary marked as field or undefined.

This field boundary is important e.g. to exclude false positives of the robot's ball-, line- or goal-detection in the undefined area outside of the field. It is also used to exclude true positives detected on a secondary soccer field in the same camera image. Furthermore, the field boundary can be used as input data for later self-localization processes, as in the localization package by Hartfill [13].

Various approaches exist, trying to solve the problem of detecting the convex field boundary. They can be differentiated into the classes of model- and data-driven approaches. Those will be presented in detail in Section II.

With *DeepField* as described in Section III, we propose a semi-supervised deep learning approach for semantic segmentation of RoboCup soccer fields in camera images. We are using several neural network architectures of a Keras-based framework for semantic segmentation by Divamgupta [23] with semi-automatic generated labels from a model-based approach by Fiedler et al. [8]. Those will be compared to each other in regard to their accuracy and runtime. We want to address the high manual configuration effort and incomplete model definitions of model-based approaches as well as the high effort of manual label annotation of data-driven approaches.

In Section IV we evaluate the results of our approach and further discuss those in Section V. Finally, we conclude in Section VI and give an outlook on future work on this topic in Section VII.

II. Related Work
WRITTEN BY JAN GUTSCHE

In this Section, we present model-based and data-driven approaches for solving the problem of detecting the convex field boundary of a RoboCup soccer field.

A. Model-based approaches

The model-based design uses a model definition, which consists of a set of rules e.g. mathematical formulas or algorithms. By analyzing, verifying and validating, the model gets iteratively improved until the design requirements of the dynamic system are met [12].

One such model-based design approach is the field boundary detector of the vision pipeline of the Hamburg Bit-Bots Humanoid League Kid-size RoboCup team by Fiedler et al. [8]. This approach is presented in detail below since our DeepField approach depends on it. This field boundary detector uses a so called color detector to segment the image into the field and other objects based on their color. Color detectors either use segmentation algorithms based on HSV color representations or RGB color value lookup tables.

For the HSV method, the color detector uses predefined minimum and maximum values, to segment an object. In the case of field segmentation, this method is not suitable, since the field consists of various colors e.g. noise in the artificial grass, that can not be represented by such a filter. Especially when considering natural lighting conditions, this

method requires several manual configurations during a game to work properly, which contradicts the fundamentals of an autonomous robot.

The method based on lookup tables is more suitable for field segmentation since the filter uses a table that consists of predefined RGB color values, that are manually classified as the field. This allows a much more precise configuration and is, therefore, more accurate. Optionally, additional color values of pixels surrounded by other pixels with known field color can temporarily be added to this lookup table to adapt this method dynamically to natural lighting changes. The downside of using lookup tables is the high manual configuration effort by selecting color values to classify them as the field. As the circumstances require, this configuration process could be necessary prior to each game and for each soccer field.

The field boundary detector uses the resulting segmented binary images of the color detector. Multiple implementations of field boundary detectors exist, but all follow the same principle of analyzing the binary image column-wise for areas classified as the field. For each column, the algorithm searches for the border between image areas with at least a certain amount of field pixels and areas with a lower amount (defined by a threshold). The field boundary algorithm either searches from top to bottom or from bottom to top. The second search method helps to reduce errors due to multiple fields in the image since the relevant field is always at the bottom. The field boundary detector blurs the thresholded mask to fill in gaps and to compensate for lines that are typically not classified as a field color. Points at the left and right end of the resulting convex field boundary can be misleading because the convex field boundary can not accurately represent the truth in case of obstacles covering field area at the edges of the image.

The model-based approach of the RoboCup standard platform league team HULKs [3] uses two distinct methods to classify pixels of the image as the field. First, the team uses predefined thresholds for chromaticity in the YCbCr color representation similar to the HSV color detector above. Second, a derived version of k-Means [17] as a clustering algorithm to find clusters of similar colors gets used. An algorithm for one-dimensional edge detection gets combined column-wise with the field color classification to determine the field boundary. Those methods may work well in the domain of the RoboCup standard platform league, but also require some configuration and calibration before the game and may fail under the impact of natural light.

B. Data-driven approaches

Data-driven approaches are based on a large number of question / answer pairs, the training data. The training process tries to optimize a model in such a way, that it mimics the answer to a given question. After successful training, the model is able to inference the answer of a unknown question on its own [6].

The RoboCup Humanoid KidSize League team MRL has developed a data-driven approach to solve the problem of detecting the convex field boundary. Their assumption is, that the convex field area of an image could always be described as a polygon of five points. Therefore, the output or answer of the model can be described as a five-tuple of points. As input data or questions respectively, images in the HSV color space were chosen. [18]

The MRL team ”designed a custom base network that performs feature extraction. [...] [They] also adopted two layers of fully connected neurons on top [...] performing regression of the outputs”. [18] The team described the initial result as promising but wants to improve this approach by further investigation.

We have no information about how large the used dataset was, but typical for a data-driven approach, it is necessary to annotate each and every image with the corresponding polygon, which requires lots of manual work.

III. Approach
written by Florian Vahl

To solve the challenges described in Section I, we trained several convolutional neural networks, that are built for the task of semantic image segmentation.

In our case, these networks are trained using semi-automatically generated labels. The generated binary label is covering the image regions which represent the field in the image. The human input is restricted to the configuration of the labeling model, which is based on the approach of the Hamburg Bit-Bots Vision Pipeline [8] (described in Section II) and the filtering of edge cases where the labeling model fails because of a scenario that is not properly covered or realistically coverable in the explicit model definition.

A. Dataset

The dataset which is used for this project contains 9188 RGB images which were recorded at multiple RoboCup competitions from 2016 to 2019 and our laboratory.

The images are taken with various cameras including mostly footage from a Basler acA2040-35gc with 12 mm focal length optics and a Logitech c910, besides that, some image sequences are taken from regular smartphone cameras.

Our dataset also contains 2184 contextless in- and outdoor images as negative data to prevent overrepresentation of images where the field always covers the bottom half of the image.

The image sequences in this dataset are organized by their recording session to differentiate images taken at different scenarios, which allows an easier configuration for each scenario separately.

B. Label generation

To train the neural networks on the given dataset, some kind of ground truth is needed. Each of these labels consists of a binary image with the same vertical and horizontal resolution as the corresponding image. They can be provided by humans who use their knowledge, to annotate the whole dataset with the intended output. This supervised approach is very laborious because many manual annotations are needed.

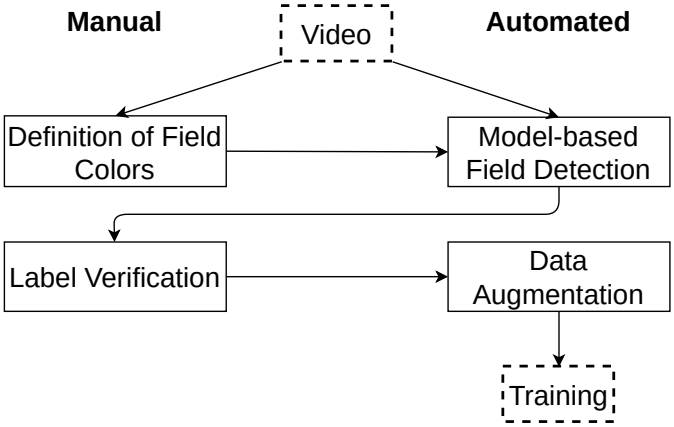

Figure 1. Data flow of the proposed training and labeling pipeline. Steps that require manual input are located on the left side of the figure while automated ones are on the right. The video input on the top stands for the image input from the dataset. The bottom right box stands for the output into the neural network training.

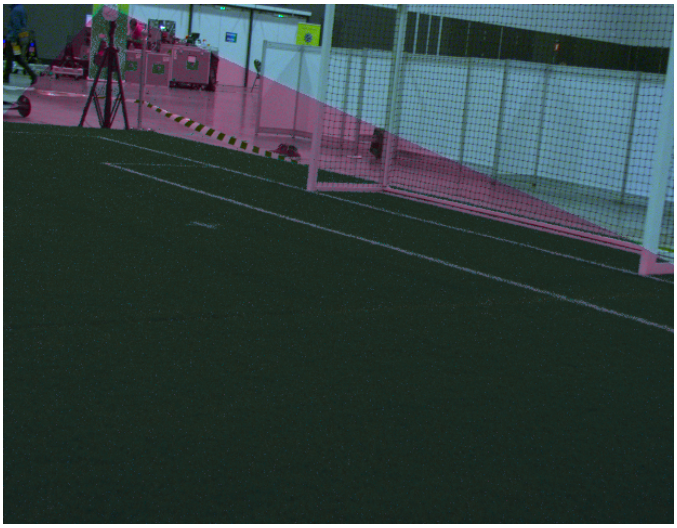

Figure 2. The auto-generated label (purple overlay) shows a significant offset relative to the real world field boundary. This happened because a wrong color, in this case, the black tone of the wifi stand, accidentally occurred in the lookup table.

To reduce the work done by a human we use the existing field boundary detection model of the Hamburg Bit-Bots [8] to generate the labels. The used model-based detection is discussed and described in Section II. The training concept's dataflow is visualized in Figure 1. Because the model needs some manual configuration for different lighting conditions, cameras, and scenarios a new configuration is created for every scenario in our dataset. If the configuration is finished the corresponding image sequence can be annotated automatically.

In the configuration, a tool called color picker is used to select the unique colors of the field. It is a part of the Hamburg Bit-Bots vision package[1]. The tool shows the video stream, while colors can be added or removed by clicking on them with a resizable selector. It outputs a lookup table file in the *.pickle* file encoding. This lookup table is then passed to the field boundary detection model for the automatic annotation of all pictures in that sequence.

The task of selecting the field colors by clicking on them takes less human effort than annotating each image individually. This is especially useful if the image sequence is very long because adding images from the same scenario doesn't add more human effort in this step. But there is a trade-off between the diversity of scenarios and the amount of human configuration because more scenarios need more human-made lookup tables. This challenge is addressed in Section III-C by adding data augmentation [21] to our labeled dataset.

The automated annotation is done by a heavily modified version of the Hamburg Bit-Bots vision pipeline[2] which loads all image sequences with the corresponding configuration, executes its field boundary algorithm and saves the resulting label. This can be done for the whole dataset at once, so it takes very little effort to do so.

The field boundary detection model has some edge-cases in which the estimated field boundary is wrong. An example can be seen in Figure 2. This happens due to various reasons. First of all the model is also designed with maintainability and limited complexity in mind, so it is still understandable and adjustable by the developer. In this case, some errors are accepted due to a better cost/benefit ratio. Furthermore, a model which is covering all unlikely scenarios would be very hard to design if not unrealistic. There are also errors occurring due to environment/lighting changes in the scenario or slightly wrong configurations.

To suppress a transfer of this model flaws into our neural network model the auto-generated labels are manually reviewed. In contrast to the configuration, the manual effort for this review scales with the number of total images. But the effort spent on each image is still quite low because most images are labeled correctly (99.5 percent are correct) and the reviewer can watch the sequence with more than realtime speed until a wrong one is noticed. If such an image occurs it is deleted from the training dataset. It could also be manually annotated instead (see future work in Section VII for further details on this topic).

C. Data Augmentation

Data augmentation [21] is applied to use the available scenarios more efficiently. This should help training the neural networks on a wider variety of lighting conditions, field structures, and other environmental conditions without adding more scenarios in our dataset. This task gets more important when we consider running this system in natural light conditions, which require coverage of a greater variety of scenarios.

The data augmentation randomly applies a variety of filters and transformations. An example can be seen in Figure 7.

Some of the used filters are a Gaussian blur, an average blur, a median blur, sharpening kernels, a simplex noise filter,

[1]https://github.com/bit-bots/bitbots_vision
[2]https://github.com/bit-bots/bitbots_vision/tree/feature/projekt/autolabel

Figure 3. Image

Figure 4. Label

Figure 5. Augmented Image

Figure 6. Augmented Label

Figure 7. The data augmentation applies several filters and transformations on the input (upper row) which results in the augmented image and label (lower row)

an additive Gaussian noise, changes in the hue, saturation, brightness, and contrast of the image. Transformations such as elastic transforms, flips along the vertical axis, perspective transform or cropping are applied to the image and the label. For this task, a python library called *imgaug*[3] is used. Horizontal flipping is not applied since we want to keep the spacial correlation of multiple fields in image space, such that only the lower one is classified as the field.

D. Training

The labeled dataset is used to train multiple convolutional neural networks.

All networks are provided by and trained with the *image-segmentation-keras* [23] python library build on top of the popular *Keras* [5] and *TensorFlow* [1] frameworks. Every network we have used is built for the task of semantic segmentation, so they are capable of pixel-precise classification. All tested network models follow an encoder-decoder scheme and the framework allows all possible encoder and decoder combinations. The used model architectures are

- Vanilla (framework default),
- VGG16 [22],
- MobileNet [11]
- ResNet [10]

on the encoder side, and

- FCN-8 / FCN-32 [15],
- Segnet [2],
- U-Net [21],
- PSPNet [26]

on the decoder side.

This allowed us to train all combinations for 60 epochs on our dataset using a simple script.

[3]https://github.com/aleju/imgaug

In the training process, a Hold-Out-Cross-Validation has been applied by splitting the dataset in a training and an evaluation dataset with a 1:2 ratio respectively. This allowed a non-loss based performance indication while training. For the comparison of the different architectures, a separate manually annotated dataset has been used.

The loss function itself is based on the Keras categorical crossentropy package. As an optimizer adadelta [25] has been used.

The hardware setup consists of two *Nvidia RTX 2080 TI* GPUs and a *AMD Ryzen Threadripper 2950X* CPU. It was provided by Ministry of Science, Research and Equalities of Hamburg as part of a student research program. The GPUs where mainly used for training and evaluation inference of the networks, while the CPU ran the parrallelized and model-based labeling pipeline. This allowed fast training, evaluation and labeling even at the same time. Most of the discussed models trained their 60 epochs in only a few minutes.

IV. Evaluation

To evaluate the real-world performance of our approach, every encoder-decoder combination has been evaluated automatically for every of its 60 trained epochs on a special evaluation dataset.

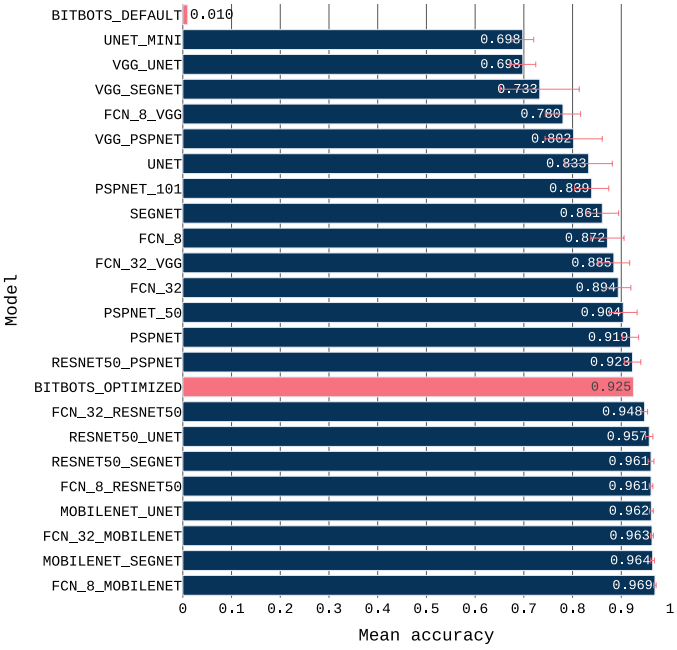

Figure 8. Each epoch of every covered model has been evaluated against our evaluation dataset. The bars show the mean accuracy and standard deviation of the top 30 percent epochs. The Jaccard Index between the manually annotated segmentation and the predicted segmentation has been used as the metric.

A. Evaluation dataset
written by Florian Vahl

The used evaluation dataset has been proposed by Fiedler et al. in the evaluation of their RoboCup vision pipeline [8]. The dataset consists of 709 images captured during a game in the 2018 RoboCup competition located in Montreal, Canada.

It includes footage of non-Bit-Bots robots playing an active game. Every image is manually annotated with all RoboCup relevant classes including all visible balls, robots, goalposts, obstacles, lines, penalty crosses, and the convex field boundary. In our case, the only relevant class is the field boundary. The other annotated classes lead to the opportunity of further evaluation if other features like lines or obstacles are included (see Section VII for further details). All images and scenarios covered in the evaluation dataset are excluded from the training dataset.

For the evaluation of natural light conditions, we propose an additional more demanding dataset consisting of 750 images taken at the 2019 RoboCup competition located in Sydney, Australia. For better comparability with the paper of Fiedler et al. [8] and a differentiation between current and future challenges, it is only used in part IV-D. The data set features bright sunlight shining at a low angle onto the field, resulting in different shadow patterns and brightness conditions. The dataset is also recorded from the robot's perspective, resulting in a more realistic scenario with an accurate point of view and robot parts that are obscuring the field.

B. Metrics
written by Florian Vahl

As a metric, the Jaccard Index, also known as intersection over union (IOU), is used to quantify the error of the predicted segmentation. The metric is used to describe the similarity of bounding boxes or image masks like segmentations, which makes it a commonly used metric (see [8], [14], [16], [24]) with build-in framework support.

C. Accuracy
written by Florian Vahl

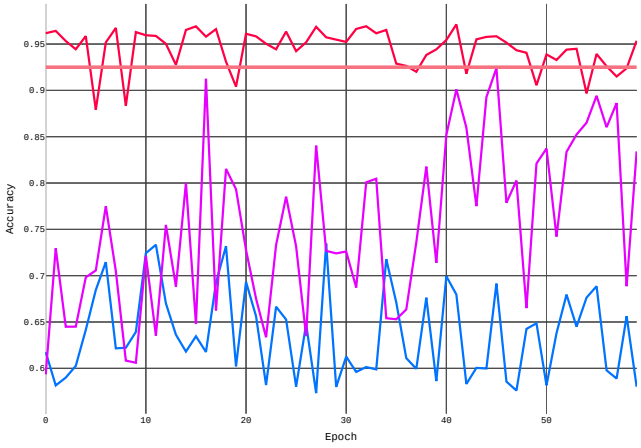

Figure 9. This graph shows the accuracy of the architectures measured as the mean Jaccard Index between the manually annotated segmentation and the predicted segmentation for a subset of the tested architectures. The blue line represents the U-Net Mini. The purple line represents the Segnet decoder with a vanilla encoder. The red line represents also a Segnet decoder, but with a MobileNet encoder. The straight light red line marks the performance of the manually optimized model-based approach of the Bit-Bots Vision [8].

The evaluation of all network architectures at every epoch resulted in interesting results. A subset of the evaluated architectures can be seen in Figure 9.

While the U-Net-Mini fails to reach an accuracy near that of the conventional model-based approach, it also shows no converging to a maximum value with progressing epochs. A manual inspection of the predicted segmentation showed many artifacts all over the image. One explanation could be a not feasible model architecture for this kind of problem. Another one could indicate a problem or bug while training the network. This leaves room for further evaluation and testing. For example, other U-Net based architectures resulted in much better results as seen in Figure 8. The MobileNet-Segnet network achieved a constant very high accuracy, outperforming the vanilla version in Figure 9 and the labeling model (indicated in Figure 9 as a red line) beginning already at epoch 0. This raises the question of how much influence the encoder choice makes in terms of accuracy.

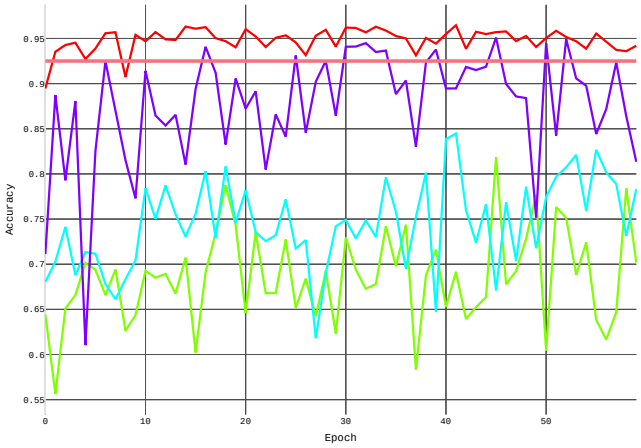

Figure 10. This graph shows the inference accuracy measured as the Jaccard Index between the manually annotated segmentation and the predicted segmentation on the evaluation dataset for the different encoders for each of the 60 trained epochs. The straight light red line shows the performance of the manually optimized model-based approach of the Bit-Bots Vision [8]. The other red line represents the mean performance of all the MobileNet encoder based networks. The purple one represents the ResNet based ones, the turquoise line shows the vanilla based encoders and the green line the VGG based ones.

In Figure 10 can be seen that the encoder makes a big difference regardless of the used decoder. Networks using the MobileNet encoder produced the highest accuracy and most consistent results.

A manual inspection of all evaluation predictions showed that the MobileNet based approaches could distinguish the field the robot is standing on from the irrelevant neighbor fields. Which is a frequent situation at RoboCup competitions.

The overall best performance showed the FCN-8-MobileNet encoder-decoder combination. It achieved a Jaccard Index of 0.969 which is a significant improvement in contrast to the model-based approach which achieved a score of 0.925 as seen in Figure 8. Detecting every image in the evaluation dataset completely as the field would result in an IOU of 0.572, therefore relatively small changes can result in huge performance differences. Another manual inspection of the results showed, that the model-based approach, which was also used for the label generation, always had a small offset in the upwards direction relative to the ground truth. This resulted in

a little offset in the labels which the neural networks adopted. Removing this offset could further increase the performance of the resulting networks, but for now, this offset seems small enough to be ignored in real-world applications.

In Figure 8 all encoder-decoder combinations are compared to each other. The mean value and standard deviation of only the top 30 percent of all the epochs are included to reduce outliers while considering the fact that the performance of different networks increases at different rates and also maybe decreases in later epochs due to over-fitting. The figure also includes the optimized vision performance of the paper from Fiedler et al. [8] and the performance with default values which result in no reliable detections due to the need for manual configuration. Furthermore, eight networks outperformed the model-based approach. As also indicated in Figure 10 non of them use a VGG or Vanilla encoder.

D. Natural light accuracy
written by Florian Vahl

The handling of natural light conditions will be critical in future RoboCup competitions. Therefore, we introduced a special dataset which is described in Section IV-A. We have only evaluated neural networks that outperformed the model-based approach under normal conditions (as seen in Figure 8). As can be seen in Figure 11 the MobileNet-U-Net combination performed the best under these conditions. It should be noted that there are no natural light conditions covered in the training dataset at the moment. Adding these would probably result in a further performance increase of these data-driven approaches. The model-based approach needed configuration using a color lookup table for this specific scenario, which should be considered when the approaches are compared. Using the default color lookup table for the model is not suitable as seen in Figure 8.

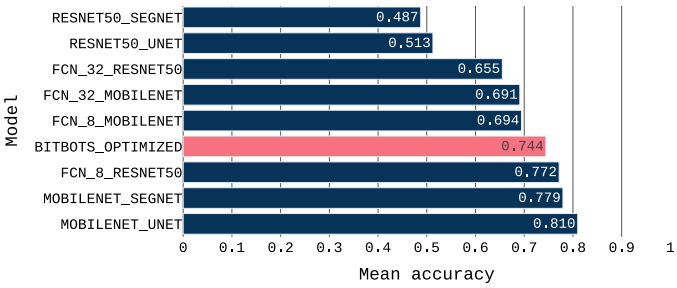

Figure 11. This graph shows the accuracy measured as the IOU between the manually annotated segmentation and the predicted segmentation on the natural light evaluation dataset for all architectures that exceeded the performance of the Bit-Bots vision in the normal evaluation.

E. Runtime
written by Jan Gutsche

Generally, low runtime values are preferred, because in a closed-loop system this means, the system could react faster to the environment, achieve a higher cycle frequency or could use the available computing resources for other tasks. A common concern of data-driven neural network approaches is their higher inference time compared to model-based ones. Therefore a comparison between absolute timing values of the used neural network architectures and the currently used model-based approach is needed. These absolute values are only meaningful when our DeepField approach is fully integrated into the robot's software and hardware stack since the real-world performance can vary significantly after applying software- and hardware-optimizations and integration. Those steps are not completed for the time of writing (see Section VII), as a result of that, this comparison will focus on relative performance differences between the model architectures, not the absolute inference time.

All timing measurements were taken on the same evaluation dataset with an Intel CPU Core i5-8259U @ 8x 3.8GHz integrated in the Intel NUC 8i5bek2 without any software- or hardware-optimizations. The runtime measurement of the Bit-Bots vision pipeline only represents the processes involved with detecting the convex field boundary. The Intel NUC computer family is commonly used in the RoboCup Humanoid Leagues, as it is the main computer model of many robot platforms [20]. The inference time does not change significantly for different epochs of one model, therefore we have used untrained models for timing measurements.

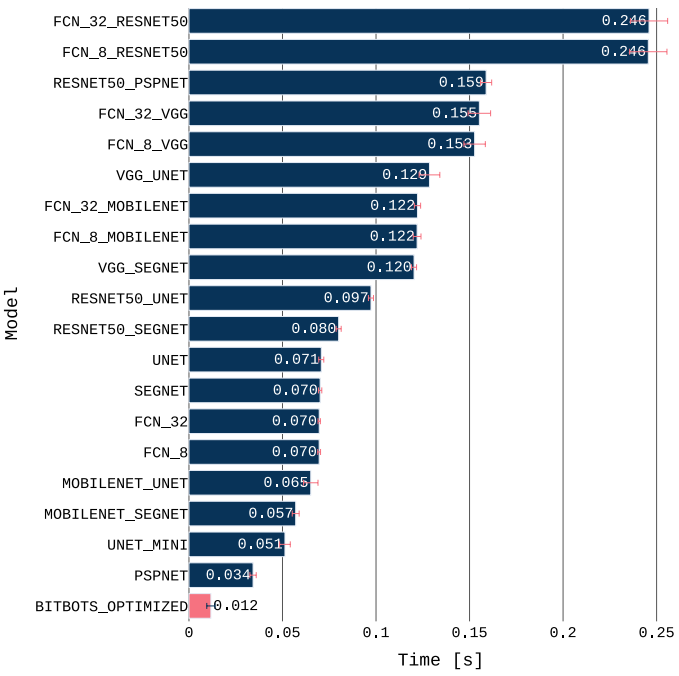

Figure 12. This graph depicts the mean inference time per image over the evaluation dataset of each model alongside of their standard deviation. The model-based Bit-Bots vision pipeline approach with specific configuration optimized for the dataset is shown in red. The data-driven models are shown in blue.

As can be seen in Figure 12, with 0.012s needed to calculate the convex field boundary of one image, the model-based Bit-Bots vision pipeline is at least 2.8 times faster than every data-driven architecture. Compared to FCN8-ResNet50 or FCN32-ResNet50 the model-based approach is over 20 times faster. The fastest neural net architectures in our comparison are

PSPNet (0.034s), U-Net-Mini (0.051s) and MobileNet-Segnet (0.057s) with low standard deviation.

F. Cost-Benefit
written by Jan Gutsche

The goal is to choose a neural net architecture, that satisfies our needs. As DeepField should be used in the RoboCup domain, we want a reliable data-driven solution, that performs accurately in various scenarios with limited computing resources. Therefore, we have to combine the metrics of accuracy and runtime. The cost-benefit ratio combines them by dividing the mean inference time (s) by the accuracy (IOU) of the model, so we get seconds per IOU (s/IOU), which gives an overview of each model's performance. The goal is to achieve a low cost-benefit ratio, as this means to get high benefit with low cost.

As the evaluation of the runtime (Section IV-E) showed, we can not expect better runtime performance from our data-driven approach than the model-based, but we want the best accuracy at a minimum runtime. For this reason, we have selected all neural net architectures with a higher accuracy than the model-based approach by the Hamburg Bit-Bots and compared them in Figure 13.

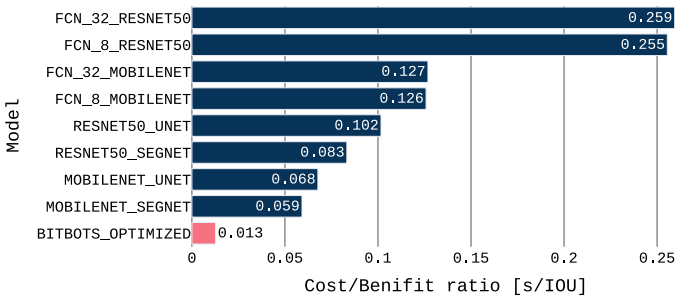

Figure 13. This graph shows the cost-benefit ratio (s/IOU) of the most accurate data-driven models (blue) in comparison to the model-based Bit-Bots vision pipeline approach (red) with specific configuration optimized for the dataset.

As can be seen, the cost-benefit ratio of the Bit-Bots vision pipeline can not be reached by any neural net architectures, since it is faster. MobileNet-Segnet is the best performing data-driven architecture with MobileNet-U-Net and ResNet50-Segnet performing slightly less.

V. Discussion
WRITTEN BY FLORIAN VAHL

A. Performance

As seen in the evaluation (Section IV) in Figure 9 some data-driven models perform even better than the Bit-Bots vision model which was used for their label generation. This raises the question if the manual label verification is the reason for the better performance. Another explanation would be the usage of data augmentation. Training the networks without data augmentation resulted in the same accuracy on the validation data set, but produced less accurate results on e.g. natural light condition scenarios that are not covered in the vision evaluation set from Fiedler et al. [8]. Therefore the manual verification is working and helps the data-driven approaches outperform their labeling model.

B. Comparison with other approaches

A direct comparison is difficult, due to the lack of evaluation data from the other approaches covered in Section II except the Bit-Bots Vision [8]. The description from the MRL RoboCup team indicates unsatisfying results, but without explicit data this is hard to quantify [18]. In general, the runtime of our data-driven approach has to be optimized to compete with the model-based ones. But considering the lower configuration effort, unexploited runtime optimization possibilities and the higher accuracy, it looks like a reasonable alternative.

C. Evaluation Dataset

A problem that occurred in the evaluation was a bias related to the main evaluation data set. While the data set allowed easy comparison with the paper of Fiedler et al. [8], it has significant problems for the purpose as an evaluation data set. First of all, it only contains images from one game in one perspective which only changes slightly. While the scenario is a typical gameplay scenario, a larger variety of scenarios in the evaluation dataset would represent a RoboCup soccer game more realistically. It includes images with two visible fields but lacks other scenarios. In contrast to the training set, the evaluation data set contains no images from the robot's point of view, for example, looking at its limbs. These special cases for the field detection, are therefore not covered in the evaluation dataset. One goal of the DeepField approach was to improve the accuracy in natural light conditions. Therefore this could not be measured with this dataset and a new dataset had to be introduced for this purpose. One bias that is covered correctly in the existing evaluation data set is the usage of non Wolfgang robots which are also not contained in the training data set. This leaves room for the creation of a more versatile RoboCup evaluation dataset in the future.

VI. Conclusion
WRITTEN BY JAN GUTSCHE

With DeepField, we presented a novel data-driven approach for semi-supervised RoboCup soccer field image segmentation with natural light conditions to minimize the manual configuration and annotation effort.

We have used an existing model-based approach by Fiedler et al. [8] to generate the training data needed to train several neural net architectures implemented in a Keras-based framework for semantic segmentation by Divamgupta [23]. All available architectures have been trained and evaluated by their accuracy and runtime performance. MobileNet-Segnet appears to be the best performing model in our cost-benefit analysis. This architecture also proved to outperform the model-based approach, especially in natural light conditions.

VII. Future Work
WRITTEN BY FLORIAN VAHL

Since the detection of the field in the camera image could be achieved with a high accuracy (see Section IV), adding

other classes such as lines or obstacles on the field seems like a plausible next step.

It could also be evaluated if a model trained on a big convex field boundary detection dataset could be used for transfer learning to train other models which include also lines and obstacles. The new model could not only profit from the ability to detect the field, but it could also use the fact, that the convex field boundary includes all obstacles and lines on the field. Therefore the model has probably already a representation of these classes in the form of the convex field boundary. This could speed up the training and lower the amount of data needed for the line and obstacle classes.

If further evaluation, testing and optimization works as intended, the proposed approach with the Segnet-MobileNet model could be integrated into our current vision pipeline [8]. This would allow us better performance in the future while playing in natural light conditions. It also would reduce the amount of configuration work needed by the team members during a competition.

Even if the current model architectures worked fine, they are mostly designed to handle more classes and more complex situations. A custom architecture could be used to improve the inference time.

Not only in the RoboCup domain, data-driven approaches become more popular in the recent years. They are replacing the previously used model-based approaches. Therefore training pipelines like these could be used outside the RoboCup context to convert model-based approaches to data-driven ones while keeping the annotation overhead low.

Acknowlegments

Thanks to the RoboCup team Hamburg Bit-Bots, especially to Timon Engelke and Jonas Hagge. This research was partially funded by the German Research Foundation (DFG), the Ministry of Science, Research and Equalities of Hamburg and the National Science Foundation of China (NSFC) in project Crossmodal Learning, TRR-169.

References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, and Zhifeng Chen. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015. Accessed 2020-02-28.

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

[3] Martin Borchert, Andrea Essig, Pascal Gleske, Chris Kahlefendt, Yuria Konda, René Kost, Konrad Nölle, Nicolas Riebesel, Maximilian Schmidt, Mayra Sharif, Felix Wege, and Bennett Wetters. Team Research Report - 2019. https://hulks.de/_files/TRR_2019.pdf, 2019. Accessed 2020-03-30.

[4] Mihai Bujanca, Paul Gafton, Sajad Saeedi, Andy Nisbet, Bruno Bodin, FP O'Boyle Michael, Andrew J Davison, Graham Riley, Barry Lennox, Mikel Luján, et al. Slambench 3.0: Systematic automated reproducible evaluation of slam systems for robot vision challenges and scene understanding. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6351–6358. IEEE, 2019.

[5] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015. Accessed 2020-04-06.

[6] Data-Driven Science. Data Science Terms Explained — Part I. https://medium.com/@data_driven/data-science-terms-explained-part-i-9eaf2cae6b16. Accessed 2020-04-05.

[7] S Tejaswi Digumarti, Lukas Maximilian Schmid, Giuseppe Maria Rizzi, Juan Nieto, Roland Siegwart, Paul Beardsley, and Cesar Cadena. An approach for semantic segmentation of tree-like vegetation. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 1801–1807. IEEE, 2019.

[8] Niklas Fiedler, Hendrik Brandt, Jan Gutsche, Florian Vahl, Jonas Hagge, and Marc Bestmann. An Open Source Vision Pipeline Approach for RoboCup Humanoid Soccer. In Stephan Chalup, Tim Niemueller, Jackrit Suthakorn, and Mary-Anne Williams, editors, *RoboCup 2019: Robot World Cup XXIII*, pages 376–386. Springer International Publishing, 2019.

[9] Junjun He, Zhongying Deng, Lei Zhou, Yali Wang, and Yu Qiao. Adaptive pyramid context network for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7519–7528, 2019.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[11] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[12] Jeff C Jensen, Danica H Chang, and Edward A Lee. A model-based design methodology for cyber-physical systems. In *2011 7th International Wireless Communications and Mobile Computing Conference*, pages 1666–1671. IEEE, 2011.

[13] Judith Hartfill. Feature-Based Monte Carlo Localization in the RoboCup Humanoid Soccer League. Master's thesis, Universität Hamburg, 2019.

[14] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1925–1934, 2017.

[15] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[16] Pauline Luc, Natalia Neverova, Camille Couprie, Jakob Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 648–657, 2017.

[17] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[18] Hamed Mahmudi, Alireza Fatehi, Amir Gholami, Mohammad Delavaran, Soheil Khatibi, Bita Alaee, Saeed Tafazol, Maryam Abbasi, Mona Doust, Asal Jafari, and Meisam Teimouri. MRL Team Description Paper for Humanoid KidSize League of RoboCup 2019. https://submission.robocuphumanoid.org/uploads//MRL_HSL-tdp-5c05a863d9671.pdf, 04 2019. Accessed 2020-03-30.

[19] RoboCup Federation. A brief History of RoboCup. https://www.robocup.org/a_brief_history_of_robocup. Accessed 2020-02-28.

[20] RoboCup Humonoid League. Qualified teams for RoboCup 2019. https://humanoid.robocup.org/hl-2019/teams/, 2019. Accessed 2020-04-04.

[21] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[22] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[23] https://github.com/divamgupta/. Image Segmentation Keras: Implementation of Segnet, FCN, UNet, PSPNet and other models in Keras. https://github.com/divamgupta/image-segmentation-keras. Accessed 2020-04-06.

[24] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. Learning a discriminative feature network for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1857–1866, 2018.

[25] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[26] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid Scene Parsing Network. *CoRR*, abs/1612.01105, 2016.