Applying Monocular Depth Estimation in RoboCup Soccer

Niklas Fiedler* Florian Vahl* Jan-Niklas Feldhusen Jianwei Zhang

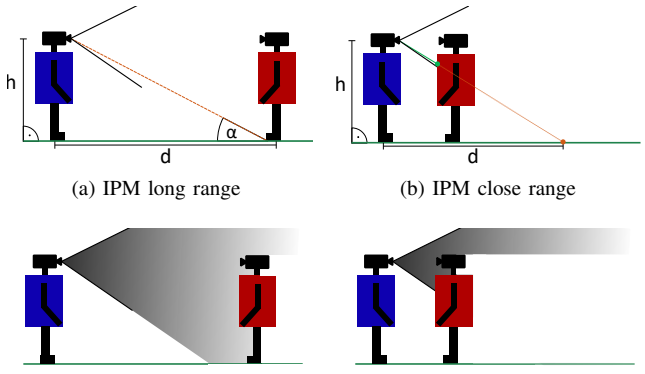
***Abstract*— We showcase a pipeline to train, evaluate, and deploy deep learning architectures for monocular depth estimation in the RoboCup Soccer Humanoid domain. In contrast to previous approaches, we apply the methods on embedded systems in highly dynamic but heavily constrained environments. The results indicate that our monocular depth estimation pipeline is usable in the RoboCup environment.**

I. Introduction

In the field of robotics, deep learning based approaches are well-proven and have been refined in recent years. This is especially true in the fields of robot vision [1] and motion [2]. While the topic of visual object detection can be regarded as solved for many domains, including RoboCup, we want to focus on depth perception for 3D-object localization and collision avoidance. Knowing the location of an object in the two-dimensional image space is usually not sufficient, as the robot interacts with its three-dimensional environment. So a link between the detection in image space and the object in Cartesian space needs to be inferred. In the following, we motivate our approach by discussing the shortcomings of conventional approaches in comparison to monocular depth estimation.

In the RoboCup Soccer Humanoid context, the robots need to measure the relative positions of objects (e. g. the ball and other players) on the field plane. Most teams approach this by using inverse perspective mapping (IPM). This is possible due to the flat surface of the field of play, which enables the robot to assume that all objects have their foot point on this plane. The camera's pose relative to this plane can be determined using forward kinematics and optionally the robot's inertial measurement unit for a more accurate orientation estimation. Therefore, the foot point of a given object in the image space can be easily mapped onto this field plane by combining the position in the image with the camera's intrinsic calibration. However, this approach has several drawbacks.

As demonstrated in Figure 1 (a), the foot point of a given object needs to be visible in the image to extract meaningful depth information. Additionally, an exact estimation of the camera's pose is needed, as minor errors in its orientation can result in highly inaccurate measurements. This is most relevant for small robots or large distances, as the error increases based on the inverse of the angle between the ray connecting the camera and the object and the ground plane.



(a) IPM long range (b) IPM close range

(c) MDE long range (d) MDE close range

Fig. 1. Schematic comparison and edge cases of inverse perspective mapping (IPM) in the upper row and monocular depth estimation (MDE) in the lower row.

The visibility of the foot point cannot be warranted in some scenarios, exemplarily shown in Figure 1 (b), where a robot may only see the upper body of another robot, especially if the robots are very close to one another. It is evident, that in a depth estimation those problems do not occur and the distance can be estimated regardless of the position of an object (see Figure 1 (c) + (d)). This comes at the cost of only providing an estimation instead of a measurement.

Intuitively, the problem of projecting detections from image space into Cartesian space would be approached by using an active RGBD-camera system. However, outside of RoboCup, there are many situations in which these cameras cannot be applied for various reasons such as bright sunlight, size, or high costs. Meanwhile, the RoboCup Soccer Humanoid rules encourage teams to use only sensors that have an equivalent in human senses [3]. This means that active depth camera systems are generally not allowed. To comply with this rule, teams need to rely on monocular or stereo vision systems.

Stereo vision systems measure depth information by recording two images simultaneously with a known distance between each other. Then, corresponding features between both images are located. Finding the corresponding features in the two images can pose a major challenge, especially in environments with a too large or too low feature density, like

*Equal contribution

The authors are with group Technical Aspects of Multimodal Systems (TAMS), Department of Informatics, Universität Hamburg, Germany {niklas.fiedler, florian.vahl, jan-niklas.feldhusen, jianwei.zhang}@uni-hamburg.de

This research was partially funded by the Ministry of Science, Research and Equalities of Hamburg as well as the German Research Foundation (DFG) and the National Science Foundation of China (NSFC) in project Crossmodal Learning, TRR-169 and the European project Ultracept (778062).

the field of play in RoboCup [3]. Using the distance between the cameras as the baseline, their intrinsic parameters, and the feature's pixel coordinates in both images, the feature position can be triangulated in Cartesian space. Still, the usage of these systems introduces many challenges by itself. Powering two cameras increases the robot's power consumption. To achieve good estimation quality, the distance between the cameras should be as wide as possible. Nevertheless, this aggravates the correspondence problem and might not fit into the robot head. Furthermore, a very precise extrinsic calibration (the positioning of the cameras and their baseline) is required for usable measurements. This is often not practical in RoboCup Soccer Humanoid because the robots have to sustain falls. High accelerations during the impact of a fall could invalidate the extrinsic calibration of the cameras resulting in significant measurement errors. Additionally, the weight of a second camera in the robot's head moves the center of mass further up and can result in a less stable walk.

Regarding research in depth perception by humans and animals, it was found at an early stage that a switch from binocular to monocular vision can be quickly compensated [4]. The findings indicated that both humans and animals are able to completely rely on monocular depth estimation. Deep learning based monocular depth estimation methods mimic this capability. Their estimation utilizes several clues from the input image, such as perspective, occlusion, and knowledge regarding object proportions. In comparison to deep-learning based stereo vision, the monocular depth estimation is slightly outperformed [5]. Still, it is more beneficial to use monocular vision on a bipedal robot, as the mentioned downsides from the usage of a second camera on the robot have a bigger impact on the overall performance of the robot than a slightly better vision with a stereo depth estimation. Thus, we deemed an investigation of approaches to monocular depth estimation in the RoboCup Soccer domain necessary.

II. Related Work

The shortcomings of both passive (e.g. IPM. stereo vision) and active (e.g. RGBD cameras) depth measurement methods yield a great interest in monocular depth estimation throughout the fields of robotics and automation. Such methods usually aim to optimize the depth estimation performance on the standard datasets such as NYU Depth V2 [6] or KITTI [7]. Early approaches such as [8] were outperformed by more and more powerful and larger architectures [9], [10], [11]. Some of them even incorporate additional features such as surface normal estimation and semantic classification [9]. While their depth estimation quality increased in precision and detail, the computational requirements also grew significantly. In the official ranking list of the KITTI Benchmark [12], it is evident, that the topic is still very actively researched [13], [14], [15]. Especially due to an interest in depth estimation on embedded devices such as smartphones, architectures were developed with a focus on runtime performance while sacrificing estimation accuracy. An early and until today most prominent (measured by the number of citations) approach to monocular depth estimation on embedded systems is FastDepth by Wofk et al. [16]. They set out to approach the issue with high computation efficiency in mind and applied this to their fully convolutional encoder-decoder architecture. Rather than using high-accuracy encoders like AlexNet [17] or VGG [18] as it was done in the work of Eigen et al. [9] or ResNet-50 [19] utilized by Laina et al. [10] and Xian et al. [11], they used MobileNet [20]. Further works such as [21] extended this idea by using reinforcement learning to develop an efficient pruning strategy and thus further increased the runtime performance. However, this process is very delicate and prone to errors, discouraging us from using it. Another more sophisticated approach utilizing a novel guided upsampling block is presented by Rudolph et al. in [22]. This was released shortly before finishing this work and provides an interesting option for future work.

III. Approach

We approach the problem of depth estimation in a pixel-wise manner with a fully convolutional neural network. As mentioned, multiple approaches to this task already exist. Our work focuses on selecting a suitable architecture, integrating it into our RoboCup vision pipeline, and evaluating whether the method is applicable in the domain. Pixel-wise depth estimation networks train a function $f_\theta : \mathcal{I}_{m\times n\times 3}\mathbb{R} \rightarrow \hat{\mathcal{D}}_{m\times n}\mathbb{R}$. The input image $\mathcal{I}$ of size $m \times n$ with three color channels (RGB) is mapped to an output depth map $\hat{\mathcal{D}}$ of the same size. Each pixel $\hat{d}_i$ of the depth map represents the estimated distance between the camera optical center and the object depicted by the corresponding RGB-pixel.

We evaluate existing neural network architectures for the application in the RoboCup Soccer Humanoid environment. This was done to provide a resource for teams with different computation capabilities available and to focus on the applicability and integration to the domain. Additionally, the used architecture can easily be exchanged.

A. Network Architectures

We compare two well established fully convolutional network architectures. The first architecture is U-Net presented by Ronneberger et al. in 2015 [23]. While it was originally designed for image segmentation tasks in the medical field, we deemed it usable for our application by using only a single output layer and employing a regression loss instead of one for pixelwise classification. The input type of an RGB-image ($\mathcal{I}_{m\times n\times 3}\mathbb{R}$) is kept. But instead of generating an activation map for each of the k segmentation classes ($\mathcal{A}_{m\times n\times k}\mathbb{R}$), we only infer a single output channel ($\hat{\mathcal{D}}_{m\times n}\mathbb{R}$) with values $\hat{d_{i,j}} \in \mathbb{R} : 0 < \hat{d}_{i,j} < 1$. U-Net consists of an encoding and a decoding stage. The encoding works like a typical convolutional neural network doubling the number of channels in every downsampling step. In the decoding stage, we upsample the feature map and halve the number of channels. To support the upsampling process, a concatenation with the corresponding feature map from the encoding layers is made. The network has a total of 23 convolutional layers. As a very early implementation

of a fully convolutional architecture, it is generally well known and already used in the RoboCup domain for object detection [24].

The second architecture we evaluate is the aforementioned FastDepth proposed by Wofk et al. in 2019 [16]. Ut utilizes the MobileNet [20] encoder resulting in an improvement of the runtime performance in contrast to other approaches, which mainly focused on estimation precision. The structure is overall similar to the U-Net architecture. It aims to achieve low latencies so that it can run in real-time on small systems. The MobileNet encoder architecture is a proven and efficient state-of-the-art network. It utilizes depthwise separable convolutional layers to reduce computational costs. They consist of a depthwise and a pointwise convolution, which are alternately used. If those layers are counted separately, the MobileNet has 28 layers. In the decoding stage, depthwise and pointwise layers are also used alternately, and additive skip connections from the encoding layers to the outputs of the middle three layers are used to support the upsampling process. This is done to avoid an increase in feature map channels. The upsampling itself is done using a simple nearest-neighbor interpolation, doubling the size of the feature map size. In total, FastDepth has 41 layers.

For both networks, the input size was set to 128×128 pixels. In our use case, high global estimation accuracy is more important than correctly replicating local features in the depth maps, so the benefits in runtime performance are prioritized.

The loss for the full depth maps generated in simulation (see Section III-B) consists of two components: The distance loss (mean absolute error) and a gradient loss which rewards an accurate structural estimation of edges without regard to the absolute distance. A weighting factor between the two is optimized using optuna (see Section III-D). For the sparse ground truth depth maps captured in the real world, the mean square error is used, ignoring regions without measurements. Labels are mapped from the interval $[0, \infty]$ to $[0, 1)$ with $\hat{x} = 1 - \frac{1}{x+1}$ prior to the loss calculation.

B. Data Collection

Both training and evaluation of the system are mainly conducted on data generated in the official RoboCup Soccer Humanoid simulation environment as the TORSO-21 dataset [25] was used. The main reason for relying on the simulation environment is the availability of reliable ground truth data while keeping the environment authentic to the competition. The TORSO-21 dataset provides depth maps for all RGB-images recorded in simulation.

For technical reasons, the depth measurements in the dataset are limited to a maximum distance of 10m. The main issue is that there is no depth data available for the background high dynamic range image (HDRi) sphere. Thus, this will also apply to our resulting network. Also, the goal nets, which are visually partially transparent, are modeled as solid objects in the depth channel. This can cause erroneous ground truth data as a network might correctly estimate the depth in a visually transparent part of the image to be 10 meters or greater, while the ground truth is defined as the distance to the net itself.

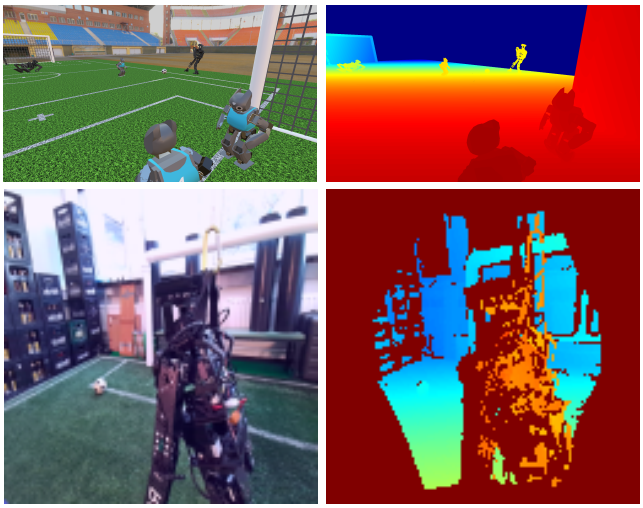


Fig. 2. Exemplary samples from the simulation and real-world dataset.

To test and train the pipeline in real-world conditions, we collected data with the Microsoft Azure Kinect. It was chosen because its performance was deemed sufficient based on the measurements of Tolgyessy et al. [26]. The real-world test environment features a single room set up for test games of the Hamburg Bit-Bots robots. It is equipped with a sub-scale RoboCup Kid Size Soccer field. During recording, objects typical in the RoboCup domain, such as goals, robots, and balls, are placed at varying positions on the field. A recording run consisted of a person moving the RGBD-camera around in the room. In this scenario, it is held at various heights and angles, mimicking viewpoints a robot would encounter. We also used an artificial light source to produce various lighting scenarios. For a small portion of the training set, we included objects not typical to the RoboCup domain to improve generalization without shifting the focus away from the domain. The test set data was recorded on a different day with new object positions and partially different variants of the soccer balls. An additional cluttered test set was also created. It includes material that is unlikely to be encountered during a RoboCup game, such as cardboard boxes, bags, pipes, or fire extinguishers. Due to the design of the Kinect sensor, sparse measurements are only provided in the center region of the image, as seen in Figure 2. This is in contrast to the simulation ground truth, which is dense and provides accurate values for all RGB-pixels.

C. Data Augmentation

Similar to the work of Tu et al. [21], we make heavy use of training data augmentation to increase the training data efficiency and achieve a better generalization. We performed geometric and color transformations as follows: A horizontal flip of the RGB-image, an RGB-shift with random shift values for each channel of the input RGB-image and a shift limit of 25 out of 255, and a random change in the brightness and contrast of the input image, both with a limit of 0.3.

TABLE I
OPTIMIZED HYPERPARAMETERS WITH THEIR COMPUTED VALUES AND SAMPLING CONFIGURATIONS FOR ALL OF OUR NETWORK ARCHITECTURES AND DIFFERENT DATA RUNS.

	FastDepth		U-Net		Optimization Configuration	
Parameter	Real World	Simulation	Real World	Simulation	Value Range	Sampling Strategy
learning rate	0.0063	0.0028	0.0001	0.0002	$1 \cdot 10^{-7}$ - $1 \cdot 10^{-2}$	log uniform
weight decay	$4.57 \cdot 10^{-12}$	$1.48 \cdot 10^{-12}$	$1.05 \cdot 10^{-8}$	$8.18 \cdot 10^{-9}$	$1 \cdot 10^{-12}$ - $1 \cdot 10^{-6}$	log uniform
batch size	4	8	4	16	$[4, 8, 16, 32]$	categorical
scheduling gamma	0.2	0.16	0.2	0.1	$0 - 0.2$	uniform (0.02-steps)
loss weighting	–	0.5	–	0.5	$0 - 1$	uniform (0.1-steps)

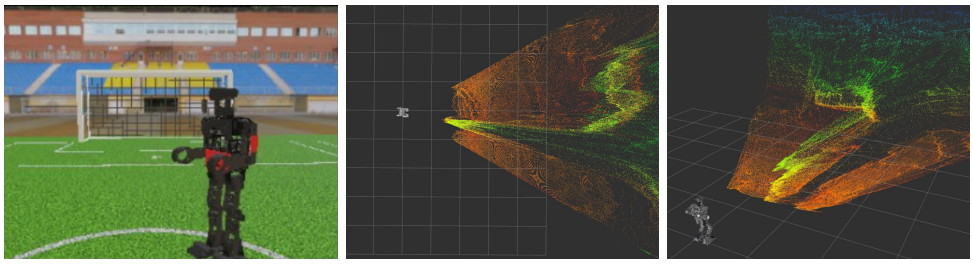


Fig. 3. Depth estimation output in the form of a point cloud in the simulation environment. The input image is shown on the left. The images on the right show two perspectives of the 3D point cloud generated based on the depth predictions and camera parameters visualized in RViz.

All transformations are applied with a 50% probability. We choose not to do a vertical and only a horizontal flip, because the data used was mostly collected from a robot's point of view. This means that our images are taken relatively close to the ground. There will also be no case in which we would have a vertically flipped image taken by the camera of the robot.

D. Hyperparameter Optimization

In the design, training, and construction of deep neural networks, a plethora of adjustable parameters need to be defined. While we kept most of the parameter definitions from the original architectures, we used the Optuna library [27] to optimize some of them for our specific use case. Most of them were related to the training process. Table I lists the optimized parameters and their value ranges. To make our findings reproducible, we also provide the best parameters we determined for our models. For the optimization, a sampler applying the Tree-structured Parzen Estimator algorithm [28] is used. All models presented in this paper were optimized on a validation set using a minimum of 1,000 trials, each with pruning activated. The validation data is a fixed subset of the training dataset.

E. Robot Deployment

In the Wolfgang robot platform utilized by the Hamburg Bit-Bots, most processing tasks are performed on an Intel NUC. The robot is also equipped with an Intel Neural Compute Stick 2 for neural network inference [29]. Because of positive experiences with the OpenVINO-toolkit [30] in the deployment of YOEO [31] to our robots, we follow a similar strategy for this approach. First, the model architecture is defined and trained in PyTorch [32]. This is especially convenient as PyTorch implementations of both architectures are available open-source, and thus, only minor adjustments are necessary. Second, the model, including the trained weights, is converted into the ONNX-format [33]. From there, the OpenVINO-toolkit model optimizer is used to convert it into an intermediate format, which is then deployed to the Intel Neural Compute Stick 2 (NCS2). This allows us to run the networks in near real-time while consuming a maximum of 2 Watts of power. It also frees up computational resources on the CPU that can be allocated to other tasks like the robot's self-localization.

Our approach is integrated into the ROS environment [34] used by the Hamburg Bit-Bots, publishing depth maps (Image message) as well as point clouds (PointCloud2 message). The latter of which can be seen visualized using RViz in Figure 3.

IV. EXPERIMENTS

To discuss the applicability of the method, provide a baseline for the RoboCup domain, and achieve comparability to other works on the topic, we evaluate the presented pipeline. The evaluation is performed with regard to the estimation accuracy as well as runtime performance on simulation and real-world data.

A. Estimation Accuracy

We evaluate the estimation accuracy of the pipeline with metrics commonly used by others [8], [21], [22]. Further, to get a better understanding of the significance of the errors, we use the mean absolute error. In both datasets, erroneous measurements are present. The simulation dataset includes cases where the camera is partially inside rigid objects, resulting in pixels with an infinite distance. They were replaced with zero-distance pixels. When collecting real-world data with the Kinect RGBD-camera, depth measurements for only a fraction of the pixels in the RGB-image are available. This is predominantly caused by areas where both sensors do not

TABLE II

QUANTITATIVE ANALYSIS OF THE DEVELOPED DEPTH ESTIMATION MODELS IN THE SIMULATION AND REAL-WORLD ENVIRONMENT. WE TESTED THE MODELS WITHOUT PRE-TRAINING (-) AND PRE-TRAINED ON SIMULATION (**S**) DATA. FASTDEPTH PROVIDES WEIGHTS FOR FINETUNING (**P**). THE BEST RESULTS ACHIEVED IN THE RESPECTIVE DATASETS ARE HIGHLIGHTED.

Dataset		Model	Pre-training	MAE	$RMSE$	rel	SIE	δ_1	δ_2	δ_3
Sim		U-Net	–	**0.402**	**0.761**	**0.230**	**0.113**	0.515	0.745	0.940
		FastDepth	–	0.917	3.386	0.414	0.124	**0.580**	**0.815**	**0.941**
Real World		U-Net	–	**0.145**	**0.245**	**0.102**	**0.042**	0.924	**0.976**	**0.987**
			S	0.146	**0.245**	**0.102**	**0.042**	**0.928**	**0.976**	**0.987**
		FastDepth	–	0.188	0.307	0.129	0.053	0.872	0.965	0.983
			S	0.214	0.337	0.142	0.059	0.858	0.963	0.983
			P	0.185	0.301	0.127	0.052	0.872	0.965	0.984
	cluttered	U-Net	–	0.271	0.439	**0.139**	**0.066**	**0.798**	0.958	0.987
			S	**0.270**	**0.410**	0.142	0.067	0.791	**0.974**	**0.993**
		FastDepth	–	0.402	0.691	0.188	0.099	0.665	0.864	0.925
			S	0.381	0.637	0.179	0.089	0.654	0.896	0.966
			P	0.364	0.619	0.173	0.088	0.700	0.897	0.957

overlap and by very dark objects and reflective surfaces such as windows or glossy plastic. The incompleteness of the ground truth data needs to be accounted for. We, therefore, chose to ignore the output of the neural network for those undefined pixels and to exclude them when calculating the metrics. Thus, the depth maps used for the metric calculations can be considered sparse and do not contain a fixed amount of depth measurements. We denote the number of usable depth measurement pixels in a ground truth depth map as P. With the ground truth depth map defined as $\mathcal{D}$, a pixel i of which is denoted as d_i and the estimated depth map being $\hat{\mathcal{D}}$ with pixel i notated as $\hat{d}_i$, the used metrics are defined as follows: Mean absolute error $MAE = \frac{1}{P}\sum_{i=1}^{P}|d_i - \hat{d}_i|$, root mean square error $RMSE = \sqrt{\frac{1}{P}\sum_{i=1}^{P}\left(d_i - \hat{d}_i\right)^2}$, relative absolute error $rel = \frac{1}{P}\sum_{i=1}^{P}\frac{|d_i - \hat{d}_i|}{d_i}$, scale invariant error $SIE = \frac{1}{P}\sum_{i=1}^{P}|log_{10}(d_i) - log_{10}(\hat{d}_i)|$ and threshold accuracy δ_j which indicates for which portion of all P pixels the following applies: $max\left(\frac{d_i}{\hat{d}_i}, \frac{\hat{d}_i}{d_i}\right) < 1.25^j, j \in \{1, 2, 3\}$.

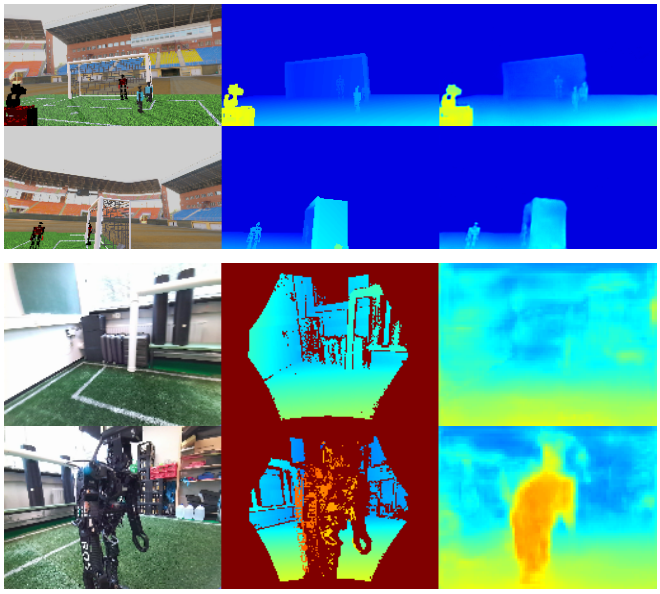


Fig. 4. Exemplary comparison between input, ground truth, and results. The upper samples show results of the U-Net model in the simulation environment as an upper bound. The lower ones present the FastDepth model on real-world scenes forming a lower bound. The FastDepth model was pre-trained on simulation data.

The system was designed to be directly applicable in simulation. Also, as explained in Section III-B, ground truth data for depth measurements are easily obtainable. We regard the results achieved in simulation as the most relevant because, as of now, this is the only environment where the system can be directly applied. To test the generalization capabilities in the real-world scenario, we created a second train and test dataset. Its creation is detailed in Section III-B. We evaluate the pipeline both on the standard and cluttered version of the real-world dataset. The results achieved in the experiments are listed in Table II. Figure 4 shows exemplary results of the trained pipelines.

B. Runtime Performance

Similar to the embedded devices mentioned in [22], [21], and [16], robots participating in RoboCup competitions have limited processing capabilities while requiring a high frame rate and low processing delays. As mentioned in Section III-E, the network architecture is converted using the OpenVINO-toolkit and runs on an Intel NCS2.

We measure the size of the network architectures by their number of parameters and multiply-and-accumulate operations (MACs). The number of frames processed per second (FPS) on the robot hardware is also measured. During

TABLE III

COMPARISON OF RUNTIME PERFORMANCE.

Architecture	Parameters	MACs	FPS
U-Net	17.267.393	10.039.083.008	8.7
FastDepth	3.960.929	245.374.976	40.9

the measurements, the networks were configured for an input image size of 128×128 pixels. The architecture was run alone on the device to allow comparability between approaches and platforms. Table III showcases the performance measurements of the architectures on our robot hardware.

V. Discussion

Our experiment results in Table II indicate a generally usable estimation accuracy. The architectures used in this work were not designed specifically for the domain. However, they were selected to showcase the possibility of adapting and integrating well-known approaches into the RoboCup Soccer Humanoid domain instead of starting the discussion on monocular depth estimation in the domain with an unproven pipeline. The results indicate a sufficiently high precision for tasks such as path planning and collision avoidance. Regarding the runtime performance, the FastDepth architecture outperforms the U-Net by a factor greater than four. As this was achieved while maintaining a usable estimation accuracy (especially in the real-world scenario), it can be considered to be a valid compromise between runtime performance and precision. It should also be noted that due to the simulator pre-training and generalization, the FastDepth model can provide estimations in areas which are not captured by the Kinect camera and are therefore not included in the real-world training data. However, this advantage is not reflected in the results shown in Table II because the estimations for pixels without ground truth data are not taken into consideration in the metrics. This would also lead to better depth estimation utilizing the FastDepth model. The real-time depth estimation capability is critical to maintain a sufficient reaction time during interaction with the dynamic and competitive environment of the robot. However, in a less dynamic scenario or with fewer limitations on the hardware, the improvements in estimation accuracy of larger and less runtime-efficient approaches could be leveraged.

The learning based monocular depth estimation approaches presented in this paper depend less on kinematic measurements than IPM, but naturally, it is also hard for them to estimate the distance of faraway objects. They could reduce large errors by implicitly learning field dimensions as well as the relation of objects relative to the field. It also needs to be pointed out that the depth estimation approaches predict pixel-wise dense predictions while the IPM is only realistically applied to the foot points of known objects, which produces a sparse depth estimation, relies on robust object detection, and is also not able to generalize on arbitrary object shapes. As already mentioned, these requirements are often not met, meaning that in common cases, IPM is not applicable while our depth estimation pipeline is. Furthermore, we do not require a bounding box detection of an obstacle to be located due to the pixel-precise approach.

Both during training and testing, the quality and the peculiarities of the datasets utilized have a massive impact on the results. Our real-world dataset is mainly designed to show how the achieved performance translates to a changed and more challenging environment. To train a usable depth estimation pipeline for real-world competitions, a more diverse dataset collected, for example, during test games is necessary. Further, the real-world depth data currently used only covers a portion of the RGB-image yielding suboptimal results. By pre-training the networks on simulation data, this effect could be reduced. Also, pre-training helped with generalization, as is especially evident in the results of the cluttered dataset.

Despite the generally promising results, the presented pipeline still has some limitations. While we do not need an accurate depth estimation for the areas outside of the field, the network still needs to be supplied with training data for these areas as it can affect the estimations in areas relevant to our domain. In the simulation, this can be partially addressed by using a large number of various backgrounds for the environment in the form of HDRis. But still, there is no depth information available for them as they are a spherical 2D backdrop. To achieve similar generalization capabilities for real-world approaches, however, a large amount of training data recorded in different scenarios is required. Using a neural network capable of detecting the field boundary [31], [35] or conventional approaches [36], [37] to remove all depth estimations above it can address the issue. However, this can negatively impact detections above or close to the field boundary, such as robots or goalposts.

The real-world data is collected using a single type of camera. While we expect similar results with a different depth camera model, given a similar quality of the depth measurements, we do not know how well the system works. Testing with other RGB-cameras (no depth measurements) showed promising predictions, but due to the missing ground truth data, no quantitative analysis was performed. Thus, we cannot quantitatively evaluate the transferability to other RGB-cameras. This is not an issue in the simulated environment, as we used images taken with various randomly selected camera intrinsics. The variance between the cameras could explain the generally weaker but still usable accuracy achieved and is not reproducible in a real-world scenario.

Even though further improvements are possible, we consider this work a successful proof of concept in the domain. General applicability of the method was demonstrated and can be leveraged for practical setups.

VI. Conclusion

We presented and evaluated a pipeline allowing the application of monocular depth estimation for tasks such as object localization and collision avoidance in the RoboCup domain. Generally, we are able to demonstrate the practical applicability of deep learning based depth estimation for the RoboCup Soccer Humanoid League simulation environment. We make use of pre-training with simulation data to generate dense depth estimations on real-world test samples despite relying only on sparse ground-truth data for real-world training. Our real-world ablation study indicates that it is also possible to use the methodology in real-world competitions.

In future work, domain-specific adaptions of the architectures and newer approaches to the problem like guided

upsampling blocks [22] need to be evaluated. Plus, we are very interested in integrating depth estimation into shared backbone networks such as YOEO [31]. Also, new real-world datasets have to be recorded, possibly during test games, to improve the applicability in real-world games. To allow for a precise real-world application of the method on the robot hardware, simultaneous data collection and fusion of a depth camera with the robot camera can be considered. An implementation of the system presented, the datasets used, and more information is publicly available at https://github.com/bit-bots/depth-estimation.

References

[1] Xin Feng, Youni Jiang, Xuejiao Yang, Ming Du, and Xin Li. Computer Vision Algorithms and Hardware Implementations: A Survey. *Integration*, 69:309–320, 2019.

[2] Huihui Sun, Weijie Zhang, Runxiang Yu, and Yujie Zhang. Motion Planning for Mobile Robots – Focusing on Deep Reinforcement Learning: A Systematic Review. *IEEE Access*, 9:69061–69081, 2021.

[3] RoboCup-Federation. RoboCup Soccer Humanoid League Laws of the Game 2021/2022. http://humanoid.robocup.org/wp-content/uploads/RC-HL-2022-Rules.pdf, 2022. Last accessed: 2022/07/01.

[4] Richard D Walk. Monocular Compared to Binocular Depth Perception in Human Infants. *Science*, 162(3852):473–475, 1968.

[5] Antoine Mauri, Redouane Khemmar, Benoit Decoux, Tahar Benmoumen, Madjid Haddad, and Rémi Boutteau. A Comparative Study of Deep Learning-based Depth Estimation Approaches: Application to Smart Mobility. In *2021 8th International Conference on Smart Computing and Communications (ICSCC)*, pages 80–84, 2021.

[6] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *European conference on computer vision*, pages 746–760. Springer, 2012.

[7] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision Meets Robotics: The KITTI Dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[8] David Eigen, Christian Puhrsch, and Rob Fergus. Depth Map Prediction from a Single Image using a Multi-Scale Deep Network. *Advances in neural information processing systems*, 27, 2014.

[9] David Eigen and Rob Fergus. Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-Scale Convolutional Architecture. In *International Conference on Computer Vision*, pages 2650–2658. IEEE, 2015.

[10] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper Depth Prediction with Fully Convolutional Residual Networks. In *Fourth International Conference on 3D Vision (3DV)*, pages 239–248. IEEE, 2016.

[11] Ke Xian, Chunhua Shen, Zhiguo Cao, Hao Lu, Yang Xiao, Ruibo Li, and Zhenbo Luo. Monocular Relative Depth Perception with Web Stereo Data Supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 311–320, 2018.

[12] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. The KITTI Vision Benchmark Suite. http://www.cvlibs.net/datasets/kitti/eval_depth.php?benchmark=depth_prediction, 2022. Last accessed: 2022/07/01.

[13] Zhenyu Li, Zehui Chen, Xianming Liu, and Junjun Jiang. DepthFormer: Exploiting Long-Range Correlation and Local Information for Accurate Monocular Depth Estimation. *arXiv preprint arXiv:2203.14211*, 2022.

[14] Chang Shu, Ziming Chen, Lei Chen, Kuan Ma, Minghui Wang, and Haibing Ren. SideRT: A Real-time Pure Transformer Architecture for Single Image Depth Estimation. *arXiv preprint arXiv:2204.13892*, 2022.

[15] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. New CRFs: Neural Window Fully-connected CRFs for Monocular Depth Estimation. *arXiv preprint arXiv:2203.01502*, 2022.

[16] Wofk, Diana and Ma, Fangchang and Yang, Tien-Ju and Karaman, Sertac and Sze, Vivienne. FastDepth: Fast Monocular Depth Estimation on Embedded Systems. In *International Conference on Robotics and Automation (ICRA)*. IEEE, 2019.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet Classification with Deep Convolutional Neural Networks. *Advances in Neural Information Processing Systems*, 25, 2012.

[18] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition*, pages 770–778. IEEE, 2016.

[20] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*, 2017.

[21] Xiaohan Tu, Cheng Xu, Siping Liu, Renfa Li, Guoqi Xie, Jing Huang, and Laurence Tianruo Yang. Efficient Monocular Depth Estimation for Edge Devices in Internet of Things. *IEEE Transactions on Industrial Informatics*, 17(4):2821–2832, 2020.

[22] Michael Rudolph, Youssef Dawoud, Ronja Güldenring, Lazaros Nalpantidis, and Vasileios Belagiannis. Lightweight monocular depth estimation through guided decoding. In *International Conference on Robotics and Automation (ICRA)*, pages 2344–2350, 2022.

[23] 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.

[24] Fabian Schnekenburger, Manuel Scharffenberg, Michael Wülker, Ulrich Hochberg, and Klaus Dorer. Detection and Localization of Features on a Soccer Field with Feedforward Fully Convolutional Neural Networks (FCNN) for the Adult-Size Humanoid Robot Sweaty. In *Proceedings of the 12th Workshop on Humanoid Soccer Robots, IEEE-RAS International Conference on Humanoid Robots*, 2017.

[25] Marc Bestmann, Timon Engelke, Niklas Fiedler, Jasper Güldenstein, Jan Gutsche, Jonas Hagge, and Florian Vahl. TORSO-21 Dataset: Typical Objects in RoboCup Soccer 2021. *RoboCup XXIV*, pages 339–346, 2021.

[26] Michal Tölgyessy, Martin Dekan, Ľuboš Chovanec, and Peter Hubinský. Evaluation of the Azure Kinect and its Comparison to Kinect V1 and Kinect V2. *Sensors*, 21(2):413, 2021.

[27] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[28] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for Hyper-Parameter Optimization. *Advances in Neural Information Processing Systems*, 24, 2011.

[29] Marc Bestmann, Jasper Güldenstein, Florian Vahl, and Jianwei Zhang. Wolfgang-OP: A Robust Humanoid Robot Platform for Research and Competitions. In *20th International Conference on Humanoid Robots (Humanoids)*, pages 90–97. IEEE, 2021.

[30] Intel-Corporation. OpenVINO Toolkit. https://github.com/openvinotoolkit/openvino, 2018. Last accessed: 2022/07/01.

[31] Florian Vahl, Jan Gutsche, Marc Bestmann, and Jianwei Zhang. YOEO–You Only Encode Once: A CNN for Embedded Object Detection and Semantic Segmentation. In *International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2021.

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, et al. Pytorch: An Imperative Style, High-Performance Deep Learning Library. *Advances in Neural Information Processing Systems*, 32, 2019.

[33] Junjie Bai, Fang Lu, Ke Zhang, et al. ONNX: Open Neural Network Exchange. https://github.com/onnx/onnx, 2019. Last accessed: 2022/07/01.

[34] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, et al. ROS: An Open-Source Robot Operating System. In *ICRA Workshop on Open Source software*, volume 3. IEEE, 2009.

[35] Arne Hasselbring and Andreas Baude. Soccer Field Boundary Detection using Convolutional Neural Networks. In *Robot World Cup*, pages 202–213. Springer, 2021.

[36] Hafez Farazi, Philipp Allgeuer, and Sven Behnke. A Monocular Vision System for Playing Soccer in Low Color Information Environments. *arXiv preprint arXiv:1809.11078*, 2018.

[37] Niklas Fiedler, Hendrik Brandt, Jan Gutsche, Florian Vahl, Jonas Hagge, and Marc Bestmann. An Open Source Vision Pipeline Approach for Robocup Humanoid Soccer. In *Robot World Cup: XXIII*, pages 376–386. Springer, 2019.