Optical 3D Object Recognition for Automated Driving

Raphael Schwarz[1], Marin Marinov[2] and Stefan Hensel[1]

[1]Offenburg University of Applied Sciences, [2]Technical University of Sofia
rschwar1@stud.hs-offenburg.de, mbm@tu-sofia.bg, stefan.hensel@hs-offenburg.de

Abstract. In this contribution, we propose an system setup for the detection and classification of objects in autonomous driving applications. The recognition algorithm is based upon deep neural networks, operating in the 2D image domain. The results are combined with data of a stereo camera system to finally incorporate the 3D object information into our mapping framework. The detection system is locally running upon the onboard CPU of the vehicle. Several network architectures are implemented and evaluated with respect to accuracy and run-time demands for the given camera and hardware setup.

Keywords: Deep neural networks, autonomous driving, optical recognition, system setup

1 Introduction

Team Schluckspecht from Offenburg University of Applied Sciences is a very successful participant of the *Shell Eco Marathon* [1]. In this contest, student groups are to design and build their own vehicles with the aim of low energy consumption. Since 2018 the event features the additional *autonomous driving* contest.

In this area, the vehicle has to fulfill several tasks, like driving a parcour, stopping within a defined parking space or circumvent obstacles, autonomously.

For the upcoming season, the *Schluckspecht V* car of the so called *urban concept class* has to be augmented with the according hardware and software to reliably recognize (i. e. detect and classify) possible obstacles and incorporate them into the software framework for further planning.

In this contribution we describe the additional components in hard- and software that are necessary to allow an opitcal 3D object detection. Main criteria are accuracy, cost effectiveness, computational complexity for relative real time performance and ease of use with regard to incorporation in the existing software framework and possible extensibility.

This paper consists of the following sections. At first, the Schluckspecht V system is described in terms of hard- and software components for autonomous driving and the additional parts for the visual object recognition. The second part scrutinizes the object recognition pipeline. Therefore, software frameworks, neural network architecture and final data fusion in a global map is depicted in detail. The contribution closes with an evaluation of the object recognition results and conclusions.

2 System Setup

2.1 Schluckspecht Car

The Schluckspecht V is a self designed and self build vehicle according to the requirements of the Eco Marathon rules. The vehicle is depicted in Figure 1.

Fig. 1. *Schluckspecht V* energy efficient vehicle in urban concept class.

The main features are the relatively large size, including driver cabin, motor area and a large trunk, a fully equipped lighting system and two doors that can be opened separately.

For the autonomous driving challenges, the vehicle is additionally equipped with several essential parts, that are divided into hardware, consisting of actuators, sensors, computational hardware and communication controllers. The software is based on a middle ware, CAN-Open communication layers, localization, mapping and path planning algorithms that are embedded into a high level state machine.

2.2 Autonomous Driving Hardware

Actuators The car is equipped with two actors, one for steering and one for braking. Each actor is paired with sensors for measuring steering angle and braking pressure.

Environmental Sensors Several sensors are needed for localization and mapping. Backbone is a multilayer 3D laser scanning system (LiDAR), which is combined with an inertial navigation system that consists of accelerometers, gyroscopes and magnetic field sensors all realized as triads. Odometry information is provided from a global navigation satellite system (GNSS) and two wheel encoders.

Communication Controller The communication is based on two separate CAN-Bus-Systems, one for basic operations and an additional one for the autonomous functions. The hardware CAN nodes are designed and build from the team coupling USB-, I2C-, SPI- and CAN-Open-Interfaces. Messages are send from the central processing unit or the driver depending on drive mode.

Central Computing Unit The trunk of the car is equipped with an industrial grade high performance CPU and an additional graphics processing unit (GPU). CAN communication is ensured with an internal card, remote access is possible via generic wireless components.

2.3 Software

Software Structure The Schluckspecht uses a modular software system consisting of several basic modules that are activated and combined within a high level state ma-

chine as needed. An overview of the main modules and possible sensors and actuators is depicted in Figure 2

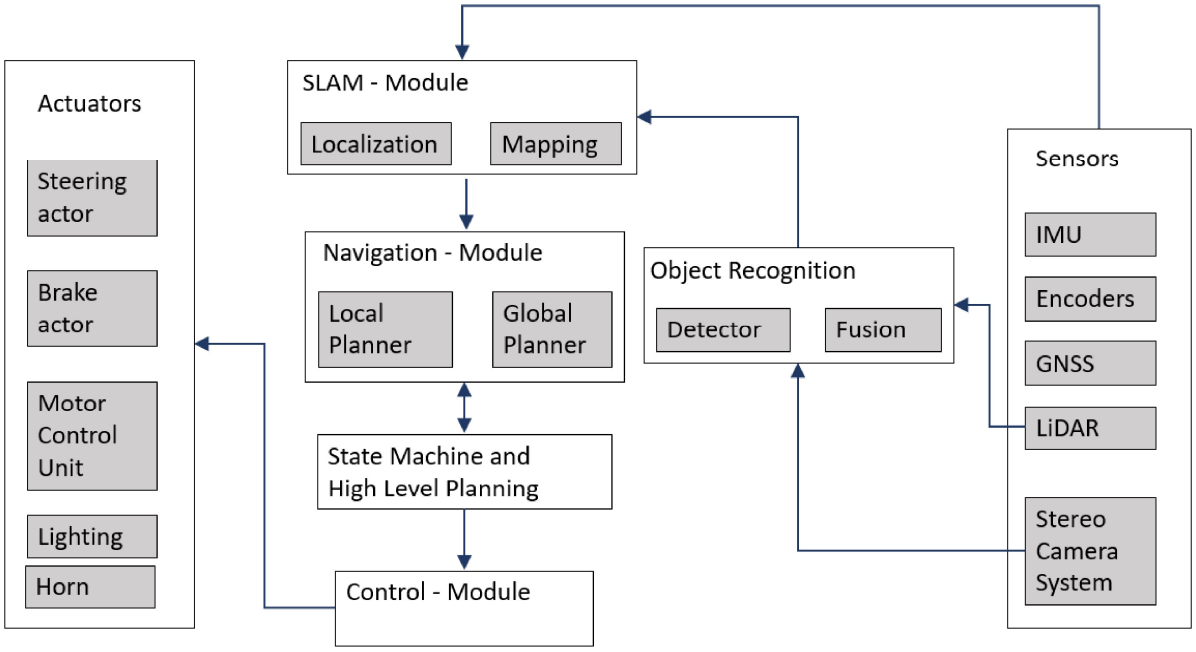

Fig. 2. Software modules and component structure overview for autonomous driving.

Localization and Mapping The Schluckspecht V is running a simultaneous localization and mapping (SLAM) framework for navigation, mission planning and environment representation. In its current version we use a graph based SLAM approach based upon the *cartographer* framework developed by Google [2]. We calculate a dynamic occupancy grid map that can be used for further planning. Sensor data is provided by the LiDAR, inertial navigation and odometry systems. An example of a drivable map is shown in Figure 3. This kind of map is also used as base for the localization and placement of the later detected obstacles.

The maps are accurate to roughly 20 centimeters, providing relative localization towards obstacles or homing regions.

Path Planning To make use of the SLAM created maps, an additional module calculates the motion commands from start to target pose of the car. The Schluckspecht is a classical car like mobile system which means that the path planning must take into account the non holonomic kind of permitted movement. Parking maneuvers, close by driving on obstacles or planning a trajectory between given points is realized as a combination of local control commands based upon modeled vehicle dynamics, the so called *local planner*, and optimization algorithms that find the globally most cost efficient path given a cost function, the so called *global planner*. We employ a kinodynamic strategy, the elastic band method presented in [3], for the local planning. Global planning is realized with a variant of the A* algorithm as described in [4].

Middleware and Communication All submodules, namely, localization, mapping, path planning and high-level state machines for each competition are implemented within

Fig. 3. Exemplary occupancy grid map of the Offenburg test track, based on LiDAR, inertial and odometry sensors.

the *robot operating system (ROS)* middleware [5]. ROS provides a messaging system based upon the subscriber/publisher principle. The single modules are capsuled in a process, called *node*, capable to asynchronously exchange messages as needed. Due to its open source character and an abundance on drivers and helper functions, ROS provides additional features like hardware abstraction, device drivers, visualization and data storage. Data structures for mobile robotic systems, e. g. static and dynamic maps or velocity control messages, allow for rapid development.

2.4 3D Object Recognition

The LiDAR sensor system has four rays, enabling only the incorporation of walls and track delimiters within a map. Therefore, a stereo camera system is additionally implemented to allow for object detection of persons, other cars, traffic signs or visual parking space delimiters and simultaneously measure the distance of any environmental objects.

Camera Hardware A ZED-stereo-camera system is installed upon the car and incorporated into the ROS framework. The system provides a color image streams for each camera and a depth map from stereo vision. The camera images are calibrated to each other and towards the depth information. The algorithms for disparity estimation are running around 50 frames per second making use of the provided GPU.

Software Framework The object recognition relies on deep neural networks. To seamlessly work with the other software parts and for easy integration, the networks are evaluated with tensorflow [6] and pyTorch [7] frameworks. Both are connected to ROS via the openCV image formats providing ROS-nodes and -topics for visualization and further processing.

3 Optical Object Recognition

The object recognition pipeline relies on a combination of mono camera images and calibrated depth information to determine object and position. Core algorithm is a deep learning approach with convolutional neural networks.

3.1 Deep Convolutional Networks

Main contribution of this paper is the incorporation of a deep neural network object detector into our framework. Object detection with deep neural networks can be subdivided into two approaches, one being a two step approach, where regions of interest are identified in a first step and classified in a second one. The second are so called single shot detectors (like[8]), that extract and classify the objects in one network run. Therefore, two network architectures are evaluated, namely YOLOv3 [9] as a single shot approach and Faster R-CNN [10] as two step model. All are trained on public data sets and fine tuned to our setting by incorporating training images from the Schluckspecht V in the ZED image format.

The models are pre-selected due to their real time capability in combination with the expected classification performance. This excludes the current best instance segmentation network Mask R-CNN [11] due to computational burdens and fast but inaccurate networks based on the mobileNet backbone [12]. The class count is adapted for the contest, in the given case eight classes, including the relevant pedestrian, car, van, tram and cyclist.

3.2 Architectures and Training

For this paper, the two chosen network architectures were trained in their respective framework, i. e. *darknet* for the YOLOv3 detector and *tensorflow* for the Faster R-CNN detector. YOLOv3 is used in its standard form with the *Darknet 53* backbone, Faster R-CNN is designed with the *ResNet 101* [13] backbone.

The models were trained on local hardware with the KITTI [14] data set. Alternatively, an open source data set from the teaching company *udacity*, with only three classes (truck, car, pedestrian) was tested.

To deal with the problem of domain adaptation, the training images for YOLOv3 were pre-processed to fit the aspect ratio of the ZED camera. The Faster R-CNN net can cope with ratio variations as it uses a two stage approach for detection based on regions of interest pooling.

Both networks were trained and stored. Afterward, their are incorporated into the system via a ROS node making use of standard python libraries.

3.3 Information Fusion within Dynamic Map

The detector output is represented by several labeled bounding boxes within the 2D image. Three dimensional information is extracted from the associated depth map by calculating the center of gravity of each box to get a x and y coordinate within the image. Interpolating the depth map pixels accordingly one gets the distance coordinate z from the depth map to determine the object position $p(x, y, z)$ in the stereo camera coordinate system.

The ease of projection between dieeferent coordinate systems is one reason to use the ROS middleware. The complete vehicle is modeled in a so calle *tranform tree (tf-tree)*, that allows the direct interpolation between different coordinate systems in all six spatial degrees of freedom.

The dynamic map, created in the SLAM subsystem, is now augmented with the current obstacles in the car coordinate system. The local path planner can take these into account and plan a trajectory including kinodynamic constraints to prevent collision or initiate a breaking maneuver.

4 Evaluation and Results

4.1 Training set evaluation

Both newly trained networks were first evaluated upon the training data. Exemplary results for the KITTI data set are shown in Figure 4.

Fig. 4. Exemplary results for the object detectors on two KITTI images. YOLOv3 is in the upper half, Faster R-CNN in the lower half (The image was cut to enlarge results).

The results clearly indicate an advantage for the YOLOv3 system, both in speed and accuracy. The Figure depicts good results for occlusions (e. g. the car on the upper right) or high object count (see the black car on the lower left as example). The evaluation on a desktop system showed 50 fps for YOLOv3 and approximately 10 fps for Faster R-CNN.

4.2 ZED camera system evaluation

After validating the performance upon the training data, both networks were started as a ROS node and tested upon real data of the Schluckspecht vehicle.

Table 1. Qualitative comparison of detection architectures.

Architecture	Frame rate [fps]	Detection quality		Quality w.r.t. size	
		Vehicles	Pedestrians	large objects	small objects
YOLOv3	9 – 10	++	–	+	+
Faster R-CNN	4 – 5	++	+	++	–

As the training data differs from the ZED-camera images in format and resolution, several adaptions were necessary for the YOLOv3 detector. The images are cropped in real time before presented to the neural net to emulate the format of the training images. The R-CNN like two stage networks are directly connected to the ZED node.

The test data is not labeled as ground truth. It is therefore not possible to give quantitative results for the recognition task. Table 1 gives a quantitative overview of

the object detection and classification, the subsequent Figures give some expression of exemplary results.

The evaluation on the Schluckspecht videos showed an advantage for the YOLOv3 network. Main reason is the faster computation, which results in a frame rate nearly twice as high compared to two stage detectors. In addition, the recognition of objects in the distance, i. e. smaller objects is a strong point of YOLO. The closer the camera gets, the bigger is the balance shift towards Faster R-CNN, that outperforms YOLO on all categories for larger objects.

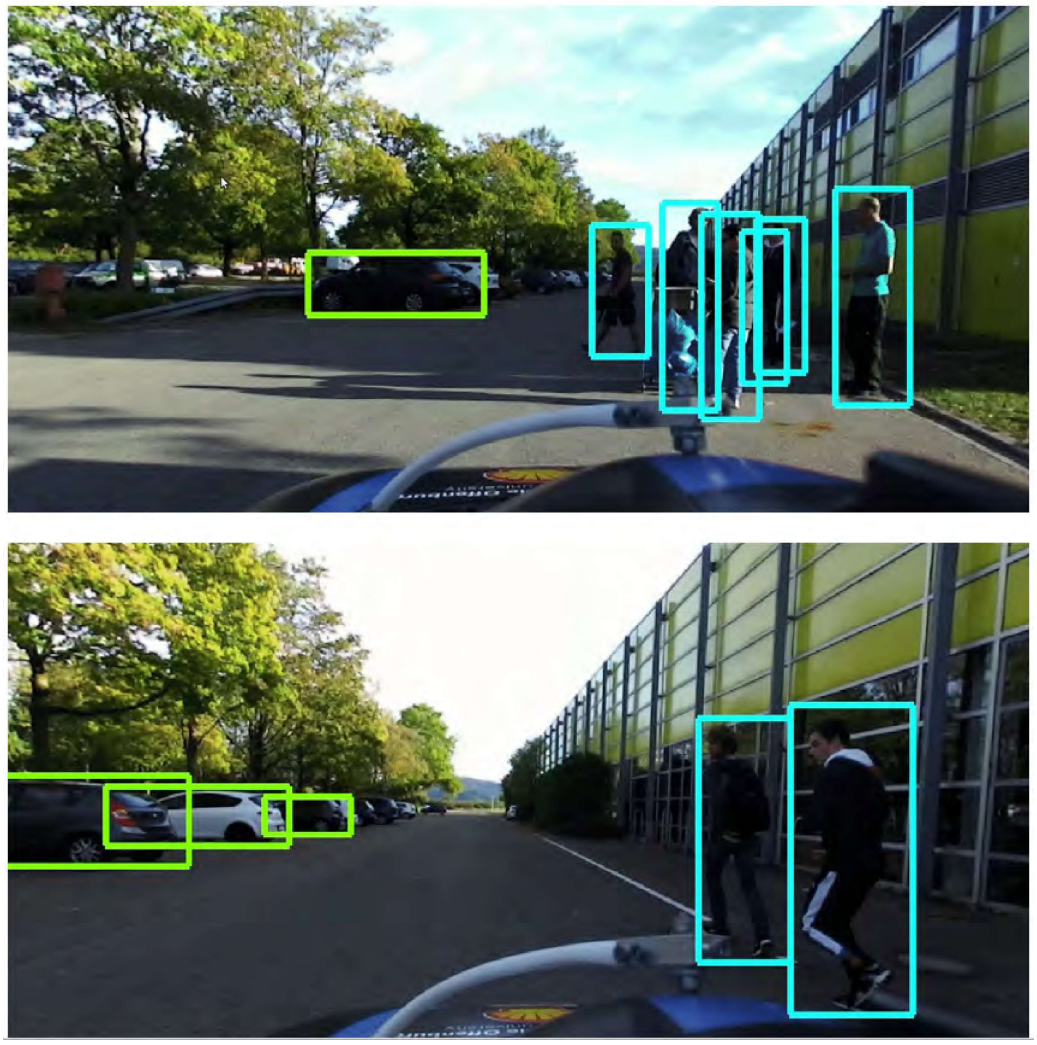

Fig. 5. Exemplary results for the object detectors on real Schluckspecht images (Persons colored cyan, cars green).

Figure 5 shows results from the YOLO-detector for a mixture of persons and cars. What becomes apparent is a maximum detection distance of approximately 30 meters, from which on cars become to small in size. Figure 6 shows an additional result demonstrating the detection power for partially obstructed objects.

Another interesting finding was the capability of the networks to generalize. Faster R-CNN copes much better with new object instances than YOLOv3. Persons with so far unknown cloth color or darker areas with vehicles remain a problem for YOLO, but

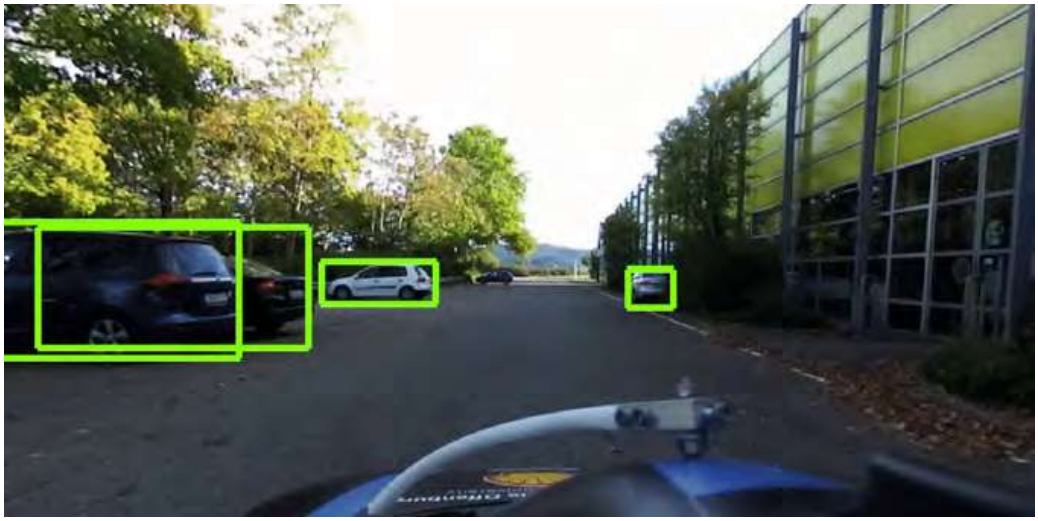

Fig. 6. Exemplary results for the object detectors on real Schluckspecht images.

commonly not for the R-CNN. The domain transfer from training data in Berkeley and KITTI to real ZED vehicle images proved problematic.

5 Conclusions

This contribution describes an optical object recognition system in hard- and software for the application in autonomous driving under restricted conditions, within the Shell Eco Marathon competition. An overall overview of the system and the incorporation of the detector within the framework is given.

Main focus was the evaluation and implementation of several neural network detectors, namely YOLOv3 as one shot detector and Faster R-CNN as a two step detector, and their combination with distance information to gain a three dimensional information for detected objects. For the given application, the advantage clearly lies with YOLOv3. Especially the achievable frame rate of minimum 10 Hz allows a seamless integration into the localization and mapping framework. Given the velocities and map update rate, the object recognition and integration via sensor fusion for path planning and navigation works in quasi real-time.

For future applications we plan to further increase the detection quality by incorporating new classes and modern object detector frameworks like M2Det [15]. This will additionally increase frame rate and bounding box quality. For more complex tasks, the data of the 3D-LiDAR system shall be directly incorporated into the fusion framework to enhance the perception of object boundaries and object velocities.

References

1. Shell: The shell eco marathon. `https://www.shell.de/ueber-uns/shell-eco-marathon.html` (2019)
2. Hess, W., Kohler, D., Rapp, H., Andor, D.: Real-time loop closure in 2d lidar slam. In: IEEE International Conference on Robotics and Automation (ICRA). (2016) 1271 – 1278
3. Rösmann, C., Hoffmann, F., Bertram, T.: Kinodynamic trajectory optimization and control for car-like robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2017)

4. Doran, J., Michie, D.: Experiments with the graph traverser program. Proceedings of the Royal Society of London. **294** (1966) 235 – 259
5. Stanford Artificial Intelligence Laboratory et al.: Robot operating system. `https://www.ros.org` (2015)
6. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015) Software available from tensorflow.org.
7. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. In: Conference on Neural Information Processing Systems (NIPS). (2017)
8. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., Berg, A.C.: SSD: single shot multibox detector. CoRR **abs/1512.02325** (2016)
9. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. CoRR **abs/1804.02767** (2018)
10. Girshick, R.B., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. CoRR **abs/1311.2524** (2014)
11. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask R-CNN. CoRR **abs/1703.06870** (2017)
12. Howard, A., Zhu, M., Chen, B., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. In: Conference on Computer Vision and Pattern Recognition (CVPR). (2016)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. CoRR **abs/1512.03385** (2016)
14. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: Conference on Computer Vision and Pattern Recognition (CVPR). (2012)
15. Zhao, Q.: M2det: A single-shot object detector based on multi-level feature pyramid network. In: Association for the Advancement of ArtificialIntelligence (AAAI). (2019)