Submitted on February 18, 2018 to the NVIDIA Jetson Challenge

Bring me a beer from the fridge: Object segmentation and manipulation on a domestic service robot extended with a NVIDIA Jetson TX2

Raphael Memmesheimer Ivanna Mykhalchychyna
and Lukas Buchhold

Abstract In this document we describe our entry for the NVIDIA Jetson Challenge. We attached a NVIDIA Jetson TX2 to a PAL Robotics TIAGo robot. The Jetson development board extends the robot by serving as a vision module. We developed a custom approach for segmenting objects in image space inspired by current segmentation networks. Through the robots RGB-D camera we can project the resulting segmentation into a local robot coordinate system for manipulating objects. We integrated the semantic segmentation into a practical use case where we propose a state dependent segmentation method. The robot navigates to a fridge, searches for a handle to open it. Once the the fridge is open, the robot takes an other look for segmenting beer bottles and cans inside the fridge and grasps them. In the end the fridge is closed and the beer is delivered. The current focus of use for the Jetson TX2 is the execution of our custom semantic segmentation network. However other custom robot modules like navigation, mapping and speech recognition can be executed on the Jetson as well. The resulting video demonstration is available under `https://userpages.uni-koblenz.de/~robbie/homer/homer/team/nvidia_jetson_challenge/`.

1 Introduction

Many people are dreaming of domestic service robots fulfilling everyday household tasks like clearing the dishwasher, preparing a meal, cleaning up the flat. Along these common answers to the question of what a robot should do when you would own one, you also hear "A robot that bring beer" quite frequently. When we investigate some research into this topic, you also find some robots that are able to bring beer, either from a fridge or a table. However, most of these robots that have been built for this special case are programmed by predefined trajectories in the need of having the beer bottles placed at a very precise location as they don't recognize the interacting object. This will lead to unwanted side effects when a bottle is not in place.

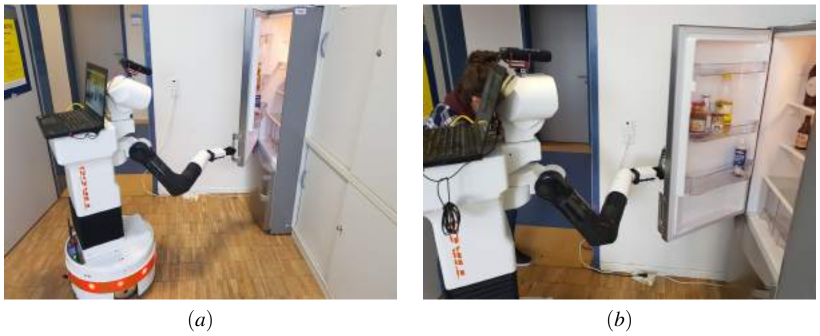

Figure 1 TIAGo first detected the handle of the fridge and opened it (a) After taking the beer it is closed again (b).

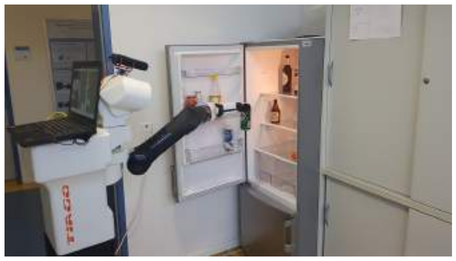

Figure 2 TIAGo getting the beer from the fridge.

We propose an approach, where we first find a handle of the fridge in order to open it. Then, in a second step we predict the precise location of the beer bottles in the fridge. In the end the fridge is closed and the beer is delivered. The models for predicting are dependent on the state of the robot. This approach has some major benefits over other beer serving robots. We do not have to mount the robot and place the beer at accurate positions, because the robot actually reasons about what it is seeing. Further there is no need to have a specialized robot trained for just the beer task. After our mobile robot executed the beer order task, it can go on to execute other tasks. Other proposed solutions for this task have shown solutions but in very slow execution speeds and / or constrained positions in the fridge. Figures of our robot executing the tasks are shown in Figure 1 and 2.

Our main contribution for this challenge are as follows:

- We developed and integrated a deep neural network for object segmentation. This network is shown in two cases. First we detect the handle and also the beer inside the fridge. Model files, as well as packages for execution are made publicly available.
- We ported previously custom developed packages for the use on the NVIDIA Jetson development board.

This document structures as follows. First, we introduce the team in Section2. Then we give a brief hardware description in Section 3. An detailed description of

Raphael Memmesheimer *Ivanna Myckhalchychyna* *Lukas Buchhold*

Figure 3 The homer@UniKoblenz team for the NVIDIA Jetson Developer Challenge

our segmentation network is shown in Section 4. Further we give a short overview over how me manipulate the segmented image positions in order to open / close the fridge and grab the beer. In the end we give a short conclusion.

2 Team

Our team homer@UniKoblenz focuses on the participation of robotic competitions. Actively, we are participating successfully in domestic robot competitions like RoboCup@Home, where we won the World Championship in the @Home track twice (2017, in Nagoya, Japan and 2015 in Hefei, China). Further we have been successfully participated (wining four of five possible prices in 2016/2017) in the European Robotics League, which is refereed to as *Champions League* of European robots.

Our team forms of two master students and a supervisor who is pursuing his PhD in *Learning by visual observation*. The team is shown in Figure 3. All of them have previously participated in robotic competitions before and enjoy the spirit of robotic challenges.

- Raphael Memmesheimer (Team Leader)
- Lukas Buchhold (Semantic Segmentation)
- Ivanna Mykhalchychyna (Manipulation)

3 Hardware

For the NVIDIA Jetson Developer Challenge we use a PAL Robotics TIAGo service robot as basis. In the following we will show and describe the used hardware components, how we integrated the Jetson TX2 development platform into the robot. We have chosen the TIAGo robot over our custom robot because the capability of lifting the robots torso and therefore enabling us to grasp objects also at higher levels in the fridge. However, the approach is easily transferable to our custom developed service robot, because they share a very common hardware design and can be programmed by a custom interface that wraps functionalities for both robots. The hardware setup is shown schematically in Figure 4.

The TIAGo robot is equipped with a 5.6m laser range finder and wheel encoders, which we use for creating a gridmap representation of the environment by a custom

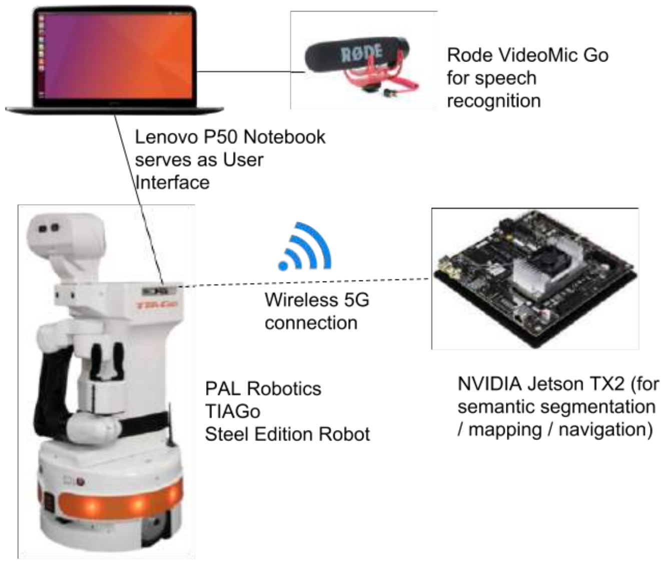
Rode VideoMic Go for speech recognition
Lenovo P50 Notebook serves as User Interface
Wireless 5G connection
PAL Robotics TIAGo Steel Edition Robot
NVIDIA Jetson TX2 (for semantic segmentation / mapping / navigation)

Figure 4 Hardware Architecture

SLAM implementation. On this map we can calculate a path based on a target position and the current position estimate. The mapping and navigation packages are publicly available[1]. In order to recognize objects, a RGB-D camera is mounted on a 2DOF pan-tilt head. The head is on top of a torso which is capable to lift the upper part of the robot up to 40cm. Further the robot is equipped with a 7DOF arm, capable of lifting objects of a weight up to 3.0 kilogram without an end-effector attached. A parallel gripper is used as an end-effector.

We integrated the NVIDIA Jetson wireless by connecting to the robots internal network using a 5G connection. The training was executed on a Personal Computer, equipped with a NVIDIA Quadro M6000 and 12GB GPU memory.

4 Software

We now present and describe the core software components of our contribution. We use the Robot Operation System [7] as general architecture. Custom packages serving functionalities such as mapping, navigation, grasping and object recognition have been ported to the Jetson development board. The core contribution is our semantic segmentation package. A ready to use ROS package is available on [2]. We also provide pre-trained weights for the beers in the fridge as well as the handle of the fridge.

4.1 Software Architecture The software architecture is shown in Figure 5. Green boxes are running on the NVIDIA Jetson TX2 development platform. Hardware nodes are executed on the onboard computer of the robot. We use ROS and have ported several packages to run on ARM. Further we developed a new segmentation

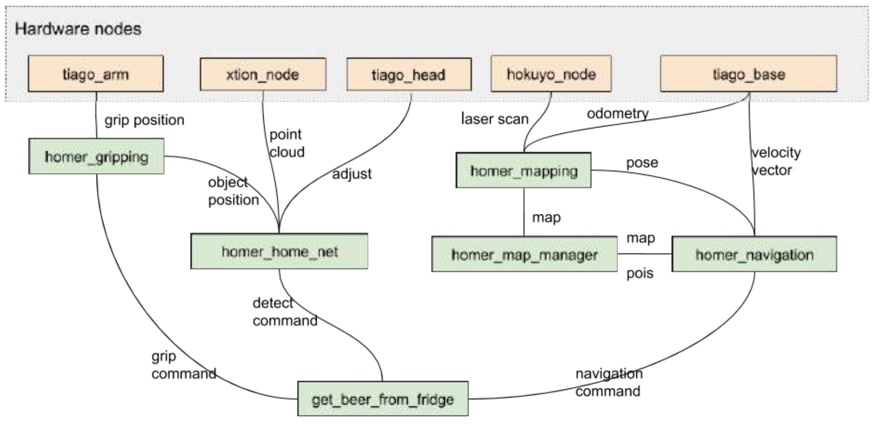

Figure 5 Software Architecture

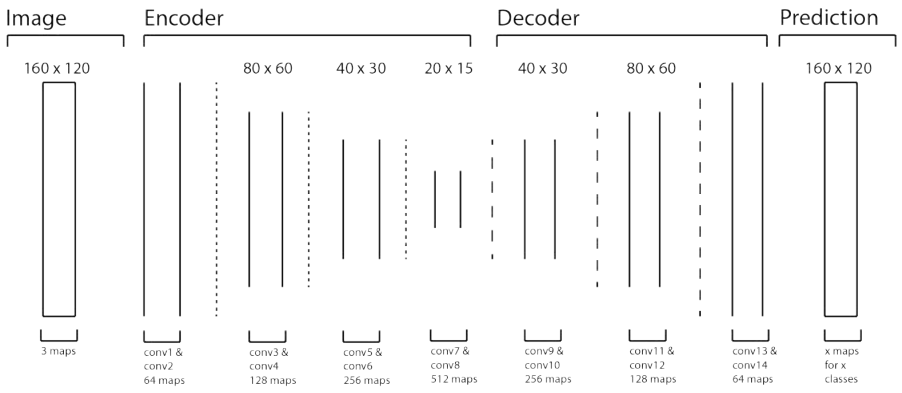

Figure 6 Architecture of our used network.

approach for the challenge which is described later. All core packages are also running on the Jetson, meaning that our mapping, navigation, speech recognition, arm control and our new object segmentation packages are able to be executed on the Jetson development board.

4.2 Semantic Segmentation

4.2.1 Architecture Like every segmentation network homeNet is divided into an encoder to generate robust feature maps and a decoder to upsample the pooled feature maps back to the original content size.
In detail, our network consists of 14 convolutional and 3 pooling layers with corresponding upsampling layers, which are illustrated in Figure 6. The encoder uses 8 convolutional layers to generate robust feature maps. The decoder upsamples the

feature maps and uses additional convolutions to refine the upsampling result.

Each convolutional layer performs a convolution with a 3×3 filter kernel and adds up a bias. We map the three color channels in the beginning to 64 feature maps and keep the number of features throughout the network. Therefore, we double the number of feature maps after a pooling layer halves the total number of features. The results are batch normalized and we apply ReLU ($max(0,x)$) as the activation function.
Particular important are the pooling and upsampling layers. These are introduced by Badrinarayanan et al. [1] as an extension to normal pooling layers. The goal is to save important local object locations before downsampling the image and losing that information. Since we want to preserve object boundaries, we need to provide the upsampling layers with some sort of saved local features. One could save the whole feature map before downsampling, but this would take too much GPU memory. The solution is to save only the max pooling indices. These are the feature map indices of the highest feature in each window.
We downsample with a 2×2 window and stride of 2 so that the windows are not overlapping. In each pooling layer, we sample the new image from the highest feature values of the 4 pixels (the max pooling index) in one window. Hence, we cut the size of the image in half.
We pass the saved location of the indices directly to the upsampling layers. An upsampling layer takes a small feature map and produces a larger spare map with the features at the positions of the saved indices. We do not need to interpolate in the upsampling process. We use convolutional layers to let the network learn to fill the resulting spares feature maps after the upsampling layer.
The last convolutional layer maps from the 64 feature maps to as much feature maps as classes to classify. We apply the softmax function to these feature maps to transform them in a range between zero and one indicating the probability of one pixel to be in a specific class. The final segmented image is then labeled using the classes with the highest probability of each pixel.
We need to use one-hot encoded labels to train this output representation of the network. One-hot encoding saves the ground truth data like the output in as many channels as classes, where each channel stands for a certain class. The correct label has a probability of one, all other channels at the same index are labeled zero.

4.2.2 Training We train the network using a cross-entropy loss function and update the variables using the gradient decent Adam optimizer [5]. We compute several metrics like the pixel and class accuracy or the mean interaction over union (mIU) to measure the network's performance and validate it every two hundred steps to detect overfitting.
We originally used an encoder that is almost identical to the VGG16 network and initialized it with the trained weights of VGG16 on ImageNet to make use of the general acquired knowledge. While this is useful on very complex segmentation tasks like the SUN [9, 4, 10] or Pascal [3] datasets we observed that more restricted problems on specific classes did not benefit with the use of VGG16 [8] weights. Instead using VGG16 as encoder, we reduced the network complexity by a few layers compared to VGG16 and pre trained it on the SUN dataset to provide it with general segmentation knowledge. The SUN RGB-D dataset contains 10000 RGB-D images

with segmented ground truth data. The NYU dataset [6] is completely integrated. It is captured by a variety of sensors with different resolutions. We downsample and reshape all pictures to a resolution of 160 by 120 and a ratio of $4 : 3$. Therefore the whole network is first trained on the SUN RGB segmentation dataset. After the initial training on the SUN set, we fine-tune our network with a smaller dataset of specific objects.
We use about 60 labeled training images and 20 unlabeled random background images in our own datasets for fine tuning. Several objects are labeled in the one image. It takes not more than 2 hours to create a specialized dataset because of the small needed dataset size.
Custom images are labeled with our own simple labeling tool. It loads image by image from a specified folder location and lets you define polygon segments around the desired objects. It assigns labels that are suitable to be passed into our softmax cross entropy loss.
The finetuning depends on the dataset size and takes about 5 to 8 hours on a Nvidia Quadro M6000.

4.2.3 ROS Integration The ROS package is public available at Gitlab. The package provides a subscriber for a constant image stream processing and a service for single time use. Any variables can be configured in the config file or in real time through the use of the ROS parameter server. It is possible to load and run multiple models to provide a broad object recognition knowledge. In our example case we simultaneously executed a model trained for handle detection and another one for beer detection.
Incoming images get pre-processed to fit the network input requirements. We cut images to a four by three format and then downsample them to 160×120 pixels.
Post-processing is used to provide smooth labels, centroids for gripping and to sort out false detections. At first contours are calculated for each coherent segment. Too small segments are considered as noise and get sorted out. To get a smoother prediction we calculate the convex hull of the remaining segment and use it as our label. Based on the convex hull we also estimate the objects centroids and determine its class using the most frequent pixel label. We project the object labels back on the original images and publish all informations.

4.2.4 Results The segmentation and classification runs with 8Hz on the Jetson TX2. On slightly faster graphic cards like an Nvidia GeForce GTX 950M we are able to archive fast real time segmentation and classification with 30Hz. Nevertheless, since our robot requests only single predictions for object recognition and grasping tasks a running speed of 8Hz is sufficient. We appreciate the 8GiB memory of the Jetson TX2 which is even more important than the prediction speed. It enables us to load and provide several models simultaneously and to run additional networks for example for pose detection or speech recognition. A single model takes about 100Mb memory.
Figure 7 and 8 demonstrate our networks prediction quality on the trained beer and handle models. The beer model is able to correctly segment and recognize 5 different beer brands while the handle model is trained to detect our fridge's handle. As we can see the network is also able to make correct predictions while objects are partly covert.
The prediction is very robust but currently lacks of two mentionable problems. First

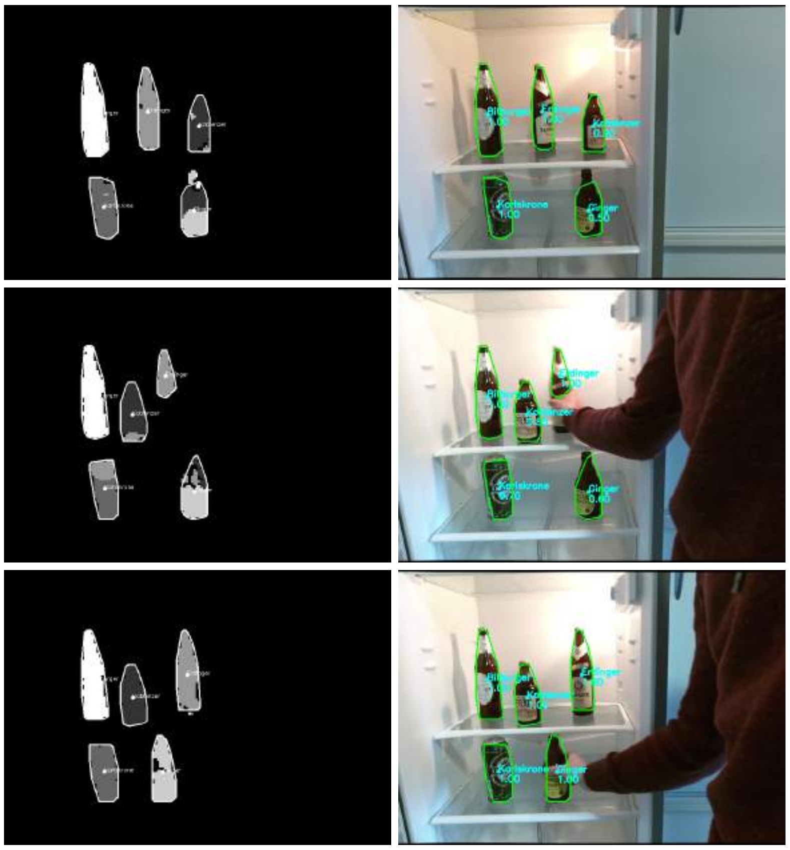

Figure 7 Example labels using the trained model on 5 beer brands. The labels are 'Koblenzer', 'Erdinger', 'Bitburger', 'Ginger' and 'Karlskrone'. On the left you see the networks output and on the right the post processed result.

it sometimes tends to cut objects into half by predicting two not connected segments of the same objects and secondly it can happen that an object is segmented correctly but classified wrong. We would like to address these kind of problems in the future by modifying our network to a recurrent convolutional network and the use of previous detection results.
Our network reliable segments and classifies various objects. Since object detection is one of the basic tasks a robot needs to acquire we are in need of robust solutions and are encouraged to further develop and enhance our network. Today it already enables our robots to precisely grasp and to manipulate its environment.

4.3 Manipulation For manipulation tasks we use motion planning framework - MoveIt! [2]. MoveIt! communicates with the robot through ROS topics, services and actions in order to get current states of the arm joints, robot pose and different sensor data, for instance Point Cloud. Once having this information and 3D-position

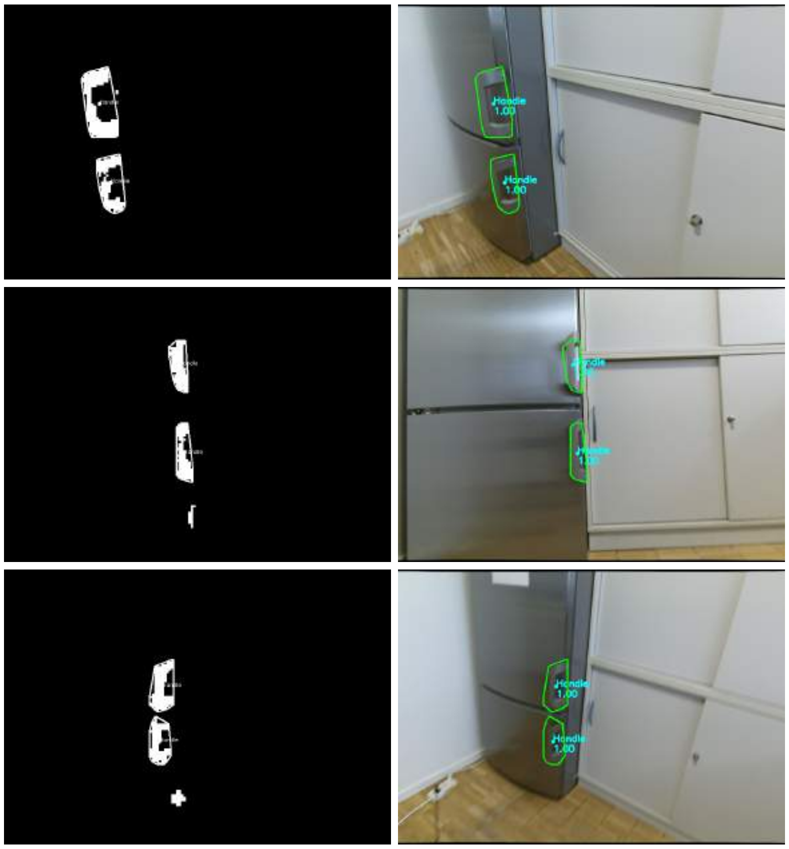

Figure 8 Example labels using the trained model on our fridge's handle. On the left you see the networks output and on the right the post processed result.

of the detected object in robot coordinate system, a motion planner is used to generate and execute the arm trajectory for moving the end-effector into a new pose.
Figure 9 depicts the initial state of the end-effector and the planned path to red marker visualizing 3D-position of the fridge handle. For grasping the handle as well as fetching the beer from the fridge the robot arm uses Cartesian motion planning, which shows larger time complexity, but produces highly accurate joint movements. In order to speed up the manipulation of the objects in the fridge robot moves its arm to predefined pose before grasping and in the next step the end-effector is moved towards the goal using Cartesian movement. After grasping action robot lifts the beer can and drives backward some small distance from fridge and changes the arm pose, in which the object can be safely transported to the desired position.
As robot has only one arm, it can not close the fridge door with object in the gripper. To solve this task the robot moves the arm to the position behind the open fridge door and pushes it by turning the base towards the fridge.

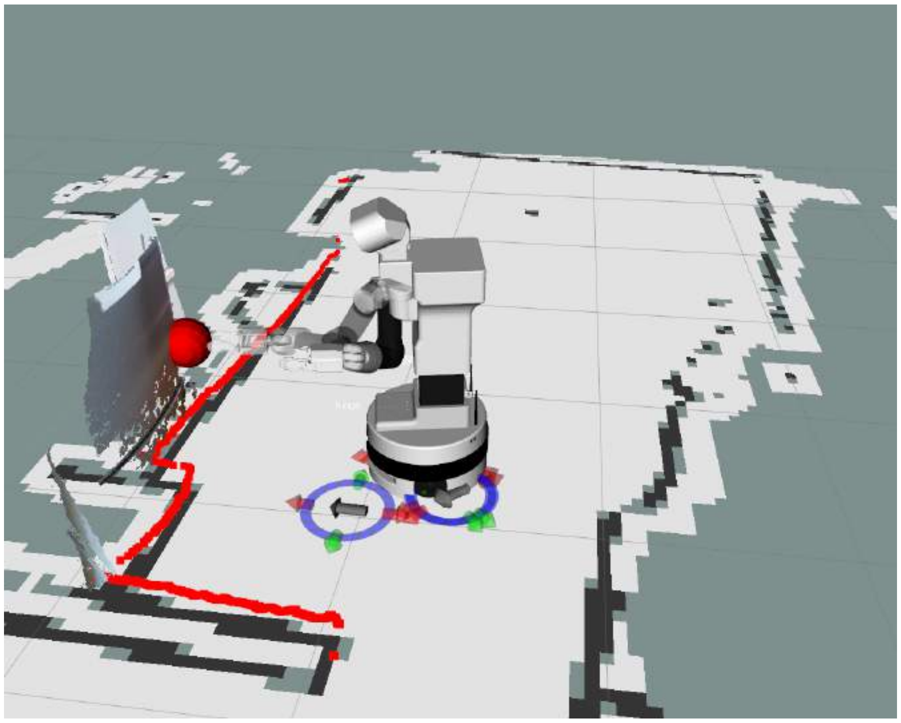

Figure 9 Motion planning to grasp the fridge handle.

5 Conclusion

We developed, trained and integrated a deep convolutional neural network on a service robot. Further we have shown a practical use case by opening a fridge, taking out a beer bottle, closing the fridge and deliver the beer. An demonstration is shown in a video. The used approaches for segmenting objects, which was our focus of this challenge are described in detail in this document. Models as well as code for executing are presented on our project website. We found that the Jetson TX2 development board brings a major benefit for executing neural networks in combination with a service robot. Mobile notebooks usually tend to have limited GPU memory, limiting the amount of neural network models. However the Jetson development board offers 8GB memory for storing networks.

Acknowledgments

We highly appreciate the support of PAL Robotics by sponsoring us a TIAGo robot platform for one year as part of the European Robotics League.

Notes

1. `https://gitlab.uni-koblenz.de/robbie/homer_mapnav`

2. https://gitlab.uni-koblenz.de/robbie/homer_home_net

References

[1] Badrinarayanan, V., A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *arXiv preprint arXiv:1511.00561*, (2015). 6

[2] Chitta, S., I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19 (2012), pp. 18–19. 8

[3] Everingham, M., L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 6

[4] Janoch, A., S. Karayev, Y. Jia, J. T. Barron, M. Fritz, K. Saenko, and T. Darrell, "A category-level 3-d object dataset: Putting the kinect to work," in *IEEE International Conference on Computer Vision Workshops, ICCV 2011 Workshops, Barcelona, Spain, November 6-13, 2011*, pp. 1168–1174, 2011, URL https://doi.org/10.1109/ICCVW.2011.6130382. 6

[5] Kingma, D., and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, (2014). 6

[6] Nathan Silberman, P. K., Derek Hoiem, and R. Fergus, "Indoor segmentation and support inference from rgbd images," in *ECCV*, 2012. 7

[7] Quigley, M., K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, volume 3, page 5. Kobe, 2009. 4

[8] Simonyan, K., and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556 (2014). URL http://arxiv.org/abs/1409.1556. 6

[9] Song, S., S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pp. 567–576, 2015, URL https://doi.org/10.1109/CVPR.2015.7298655. 6

[10] Xiao, J., A. Owens, and A. Torralba, "SUN3D: A database of big spaces reconstructed using sfm and object labels," in *IEEE International Conference on Computer Vision, ICCV 2013, Sydney, Australia, December 1-8, 2013*, pp. 1625–1632, 2013, URL https://doi.org/10.1109/ICCV.2013.458. 6

Active Vision Group
Institute of Computervisualistics
University of Koblenz
Universitätsstr. 1
56070 Koblenz
Germany
raphael@uni-koblenz.de
http://homer.uni-koblenz.de