Classifying Obstacles and Exploiting Knowledge about Classes for Efficient Humanoid Navigation

Peter Regier   Andres Milioto   Philipp Karkowski   Cyrill Stachniss   Maren Bennewitz

***Abstract*— In this paper, we propose a new approach to humanoid navigation through cluttered environments that exploits knowledge about different obstacle classes and selects appropriate robot actions. To classify objects from RGB images and decide whether an obstacle can be overcome by the robot with a corresponding action, e.g., by pushing or carrying it aside or stepping over or onto it, we train a convolutional neural network (CNN). Based on the associated action costs, we compute a cost grid of the environment on which a 2D path can be efficiently planned. This path encodes the necessary actions that need to be carried out to reach the goal. We implemented our framework in ROS and tested it in various scenarios with a Nao robot. As the experiments demonstrate, using the CNN the robot can robustly classify the observed obstacles into the different classes and exploit this information to efficiently compute solution paths. Our system finds paths also through regions where traditional planning methods are not able to calculate a solution or require substantially more time.**

I. Introduction

As humanoid robots are designed to work in human environments, one of the tasks that need to be solved consists of navigating through a cluttered environment where stepping over or onto obstacles or moving an object out of the way might be necessary. Finding suitable robot motions in these cases imposes a high level of complexity and is difficult to solve efficiently.

Common approaches to humanoid navigation in complex environments involve whole-body motion planning and multi-contact planning [1], [2]. These approaches usually take several seconds up to minutes to compute a solution. Other frameworks employ footstep planning [3], [4], [5]. However, if a blocking object needs to be moved aside to reach the goal location, they do not yield a solution.

In this paper, we present a novel approach to humanoid navigation that combines fast 2D path planning with 3D footstep planning and object manipulation actions in obstructed regions of the path. We consider an indoor environment from which the robot creates a 2D grid map with static obstacles in the absence of clutter using a standard mapping approach [6].

During navigation, the robot adds information about objects of different classes perceived with its camera to the map. We hereby use a convolutional neural network (CNN) to classify the objects and to decide whether an obstacle can be *stepped over* or *stepped onto*, or moved out of the way by *pushing* or *carrying* it aside. For the classification task, we use our recently developed real-time CNNs that are capable of segmenting RGB images to detect given object classes [7]. This framework provides reliable information about specific object types (e.g., books, boxes, toys etc.) that we map to appropriate action types, which allow the robot to navigate across the corresponding area. Our approach adds the associated stepping and manipulation costs to a 2D map that is used for path planning. In this way, we greatly simplify the full planning problem, as we split the whole plan into several parts, while each part is solved individually. Fig. 1 illustrates a motivating example, where the robot can only reach its goal by manipulating an object. According to the resulting cost map after classifying the obstacles, our planner chooses a path where the robot needs to move the stuffed toy aside.

All authors are with the University of Bonn, Germany.

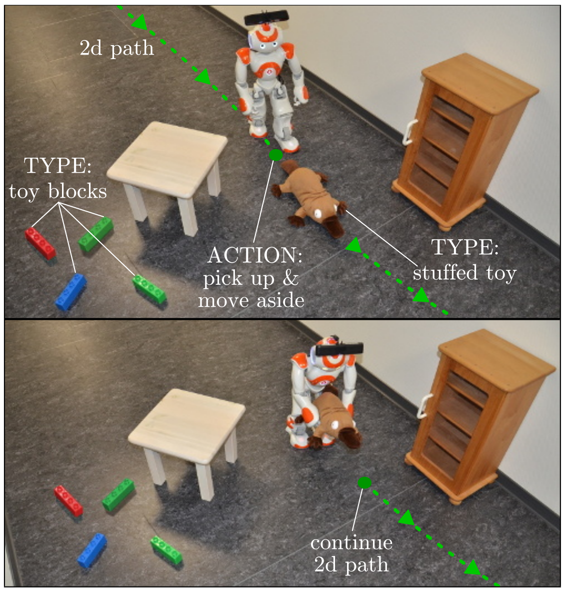

Fig. 1. Application example, in which the path of the robot is blocked. Based on classified non-static obstacles (in this case a stuffed toy and toy blocks) and their associated actions and costs, our system computes a cost grid on which a 2D path can be efficiently computed. The path also encodes the actions that need to be executed by the robot during navigation to reach the goal. As can be seen, the robot moves the stuffed toy aside to clear its path and can then continue walking along the 2D path.

We implemented our framework in ROS and tested it in various experiments with a Nao humanoid. As the experimental results demonstrate, the robot can robustly classify the observed obstacles into different classes and use this information to efficiently find solution paths through passages where objects are in the way.

II. Related Work

Stilman and Kuffner were the first who considered navigation amongst movable objects where the robot can move objects aside if necessary to reach the goal location [8].

The idea of their NAMO approach is to decompose the environment representation into disjunct regions and search for obstacle motions that connect two disjoint regions allowing the robot to transition between them. Stilman *et al.* employed the framework on a real humanoid to navigate through an environment with movable chairs and tables [9] where the world state was observed by an external motion capture system. In contrast to this approach, we perform fast grid-based planning on a map which can be easily obtained by using standard mapping algorithms [6]. Furthermore, we classify objects perceived by the robot and encode different actions associated to the object classes in navigation costs. We then combine the planning on a 2D cost grid with local 3D footstep planning and object manipulation.

Levihn *et al.* approached a variant of that problem in which the robot can utilize objects to get to locations that are otherwise out of reach for the robot [10]. The authors introduced the concept of relaxed-constrained planning where the planner is allowed to violate certain constraints. The violation is then locally resolved by using suitable objects, e.g., to overcome a high step height.

Hornung *et al.* addressed the task of collecting objects and delivering them to designated places while clearing cluttered obstacles out of the robot's way [11]. The authors proposed to apply a high-level planner with integrated perception, world modeling, action planning, navigation, and mobile manipulation. Our navigation framework is orthogonal to that approach and could be integrated into such a higher level task planning framework.

Grey *et al.* recently presented an approach that uses so-called randomized possibility graphs to traverse environments with arbitrary obstacles, in which footstep as well as whole-body motion planning is required [1]. The authors distinguish between passages that are definitely passable by the robot and ones that are definitely impossible to be passed by using approximations of the constraint manifold. So far, their approach has only been tested in simulation with no perception involved. Lin and Berenson considered navigation in uneven terrain using contact planning of palm and foot locations and learned estimates about the traversability of regions [2]. The idea of this approach is to use the learned traversability estimates as a measure how quickly the planner can generate feasible contact sequences. This measure is used in the heuristic function of the contact space planner to guide the search to areas with more contactable regions. While these approaches also aim at speeding up the search for viable solutions paths, the authors do not take into account the possibility of actively modifying the environment.

The method proposed by Kaiser *et al.* extracts affordances of geometric primitives to support the planning of whole-body locomotion and manipulation actions [12]. Navigation through cluttered passages has not been considered in their scenario.

Although some of the above mentioned approaches provide a robust way to plan through environments containing cluttered regions, they neglect the semantics of the objects. Semantic information, however, is a key component of human navigation. Current advances in computer vision using deep CNNs to extract semantics of the environment [13], [14], [15] have made it possible to infer the semantic classes of objects in cluttered scenes with high accuracy. Such approaches allow us to map each object class to a different action type that we can use for planning in an efficient way. Running our classifier in a fast manner is necessary to avoid adding a large computational overhead to our approach. Thus, we build on top of recent work focusing on *real-time* CNNs [7], [16], [17], [18]. Since, training deep CNN models is a data intensive task, we alleviate this by generating a large-scale dataset of our interest clutter classes with minimal effort in a semi-autonomous way by crawling images from the Internet.

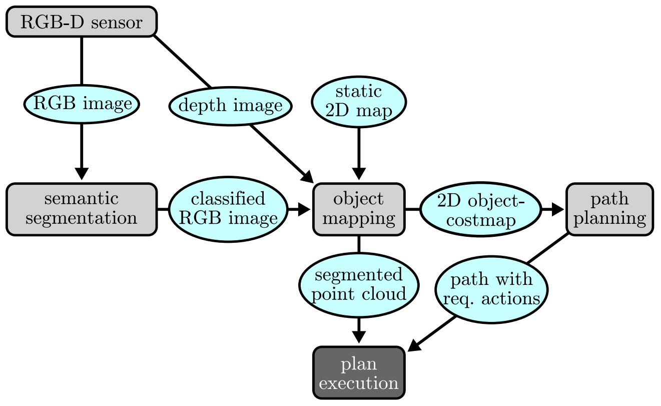

Fig. 2. Overview of our framework. The different components are summarized in Sec. III.

III. System Overview

Before we describe our approach in detail, we present an overview of the individual components, which are also illustrated in Fig. 2. The first important component is the *semantic segmentation* of the input RGB image. The semantically segmented image is then provided to the *object mapping*, which additionally uses 3D point cloud data from the input depth image aligned with the RGB data, and a 2D grid representation of static obstacles in the environment. This 2D static map is constructed using a standard mapping system [6] in the absence of cluttered objects. The output of our object mapping is a 2D cost grid, encoding static obstacles as well as the detected and classified objects, on which the *path planning* takes place. The execution of the computed path, which comprises actions corresponding to the detected objects, is done by the *plan execution*. The plan execution additionally uses the segmented point cloud when necessary, while invoking the required object actions.

During navigation, our system continuously updates the representation of the environment with information about newly sensed obstacles that might be blocking the way of the robot and replans the robot's path toward the goal if necessary.

IV. Semantic Segmentation

Our approach aims at inferring possible robot actions from the objects in the environment in real-time. This requires

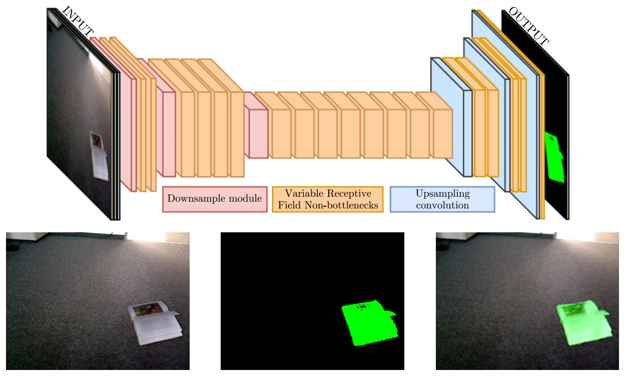

Fig. 3. Top: encoder-decoder semantic segmentation CNN based on the non-bottleneck concept behind ERFNet [17] inferring an image from our dataset. Bottom, left to right: original RGB image, prediction from CNN, and alpha blend for visual qualitative performance assessment. Best viewed in color.

a visual classifier that can recognize individual objects accurately from a dictionary of possible classes, while still running fast on a power- and payload-constrained machine, such as a humanoid robot. The state of the art in object detection and semantic segmentation using CNNs makes the accuracy of such algorithms acceptable for this approach to be feasible, but most CNN pipelines are computationally intensive and require large amounts of training data. We rely on a lightweight architecture to achieve a good runtime vs. accuracy trade-off. To approach the amount of training data needed, we use pre-trained models from Bonnet's library [7], which already provides useful features in the convolutional layers and we create a large dataset by mixing images recorded by ourselves and a huge amount of scavenged data from the Internet for refining the pre-trained models.

We opt in favor of a semantic segmentation pipeline which maps each pixel of the robot's camera images into one of the eight classes: *"balls", "books", "boxes", "cars", "dolls", "stuffed toys", "toy blocks", and "background"*, where each class has at least one navigation action assigned to it. Note that our approach is not limited to these classes and could easily be extended.

A. Bonnet

Fig. 3 shows a diagram of the encoder-decoder CNN architecture used in our approach. The chosen architecture is based on ERFNet [17], which proposes to change each computational bottleneck introduced in ResNet [14] and ENet [16] with a "separable non-bottleneck" of a variable receptive field. This module uses a set of separable filters of sizes $[1 \times 3]$ and $[3 \times 1]$ and different dilation rates, which makes each layer effectively wider without increasing computational cost, allowing the network to be more descriptive without affecting runtime. The choice of using different dilation rates allows the network to have a bigger equivalent receptive field in the image space, capturing long-range dependencies, which is key for big objects. By using a model with these properties, and adjusting the number and width of the layers to fit our data, we can achieve a

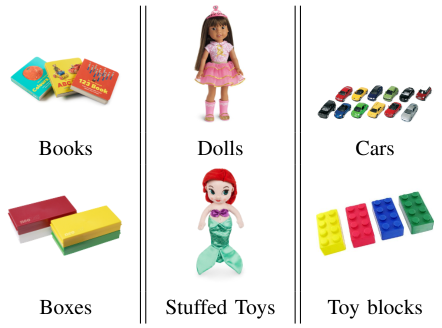

Fig. 4. Examples of pairs of classes with low inter-class distance, challenging for the CNN, but important for our approach due to different associated object interactions.

model that is descriptive enough to provide us with accurate semantic labels while running in real time.

We start from a pre-trained encoder from a model which was trained with the COCO dataset [19] and therefore provides rich features even before the training. We attach a small decoder that is trained from scratch using random weights and fine-tune the model training end-to-end using back-propagation and a pixel-wise cross-entropy loss with a dataset of 5,000 images containing roughly 20,000 toys instances with their respective masks, which we explain in the following section.

B. Data Collection

As previously stated, training deep CNNs requires a large amount of labeled training data to obtain the accuracy required to run other approaches on top of the obtained semantics. This effect is particularly magnified when using a semantic segmentation pipeline, because a holistic knowledge of what each image contains is not enough, and labels are required at a pixel level, increasing the effort required to label each image considerably. Even though a using pre-trained model helps to reduce the amount of required labeled data, a particular case which makes the training more data-hungry is having low inter-class distances, like in our case (see Fig. 4). To circumvent this problem, we collected a dataset with 1,000 objects focusing on the hard examples of inter-class distance, i.e., focusing mostly on labeling objects whose appearance is similar, but which have a different semantic label. This is done to train CNN features that are sensitive enough to allow the classifier to fit the classification hyper plane effectively. To generate a dataset for semantic segmentation, we need pixel-wise masks for each individual object. Since such a labeling is expensive, we collected the data with an RGB-D camera and segmented the objects in the depth channel to obtain the ground-truth mask, before feeding them to the CNN as a 3-channel RGB-only image.

In order to scale up the dataset and make it an order of magnitude bigger, we wrote a script to automatically download images from the Internet with properly formatted queries returning images fulfilling the following conditions: (i) the image contains only one of the desired classes in the dictionary, and (ii) the image contains either an alpha channel making the background transparent or has a blank

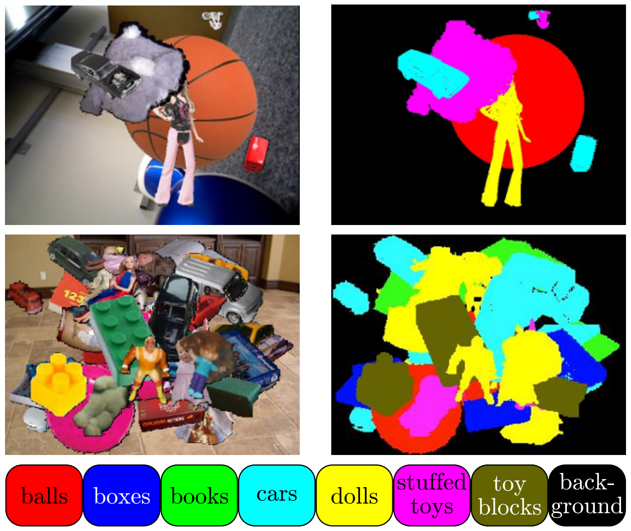

Fig. 5. Examples of generated clutter images with added background. Left: RGB image, right: ground truth. Best viewed in color.

TABLE I

EXAMPLE ASSOCIATION OF ACTIONS TO OBJECT CLASSES. THE ACTIONS ARE CHOSEN BY AN EXPERT USER ACCORDING TO THE ROBOTIC HARDWARE, IN THIS CASE A NAO ROBOT.

Object class	**Action type**
balls	push
toy blocks	step over
boxes, books	step onto
stuffed toys, dolls, cars	pick up

background. Under these restrictions, the script returned roughly 25,000 images using Google. We further reduce it to roughly 20,000 images after six hours of supervised cleaning by a human. The supervision consists of navigating quickly through the crawled images and identifying objects that either do not belong to the specific class we are interested in, or whose alpha channel does not fit the object boundary. For our CNN to be usable in realistic environments, the last step in this dataset generation method is to generate 5,000 clutter images from our raw data, containing one of 300 different backgrounds and any number of random objects from the database from 0 to 20 objects per image. This can be considered as "synthetic" data, but it is a step closer to the real world, because the images are of real-world objects (Fig. 5).

V. Path Planning Utilizing Obstacle Information

In this section, we describe our method to exploit the semantic information about segmented objects during path planning.

A. *Object Mapping*

We use the semantic segmentation from Sec. IV and combine it with the depth information of the RGB-D image to get a segmented point cloud of the corresponding objects (see Fig. 6). Afterwards, we project the segmented point cloud onto a 2D grid map representation of the environment containing inflated static obstacles as illustrated in Fig. 7a.

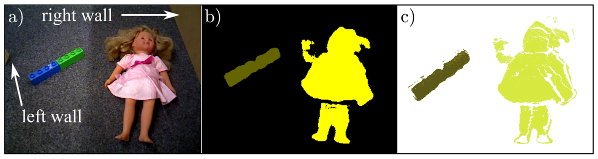

Fig. 6. Data processing for the RGBD data. a) The original RGB image containing a doll and toy blocks between two walls. b) Semantic segmentation results using the Bonnet framework [7]. c) Segmented point cloud of the corresponding objects, using the depth image to get the spatial information of each marked pixel.

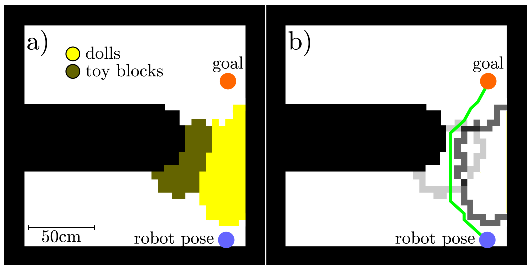

Fig. 7. a) Visualization of the projection of the objects onto a 2D grid. Inflated static obstacles are represented as black cells and cells with detected objects from the segmented point cloud (see Fig. 6c) are color coded. Yellow cells correspond to the doll, brown cells to the toy blocks. b) Resulting 2D cost map for planning with the costs of the object actions encoded in the border cells of the objects. The border cells of the toy blocks (light gray), which can be stepped over, have lower cost than the cells corresponding to the doll (dark gray), which needs to be moved away. The overlapping border cells of the inflated objects are the sum of their action costs.

Inflating all objects with the robot radius is a general concept to prevent the robot from colliding with obstacles in case of localization errors. We maintain an object database that contains the information about objects in form of object ID, object class, and the set of corresponding 2D grid cells. Note that obstacles that cannot be classified by the CNN and, thus, are considered as background are mapped as static and are not stored in the database of objects that can be manipulated.

We assume that an expert user assigns for the individual object classes possible actions defining how the robot can overcome such obstacles during planning. The possible actions for the object types inherently depend on the specific robot. As an example, Tab. I shows the actions associated with the object classes for a Nao. The actual execution of the actions is planned during plan execution (see Sec. V-C).

Our approach uses a 2D cost grid map for path planning that encodes, in addition to the static obstacles, the costs of the actions according to their estimated complexity and corresponding estimated execution time in relation to just walking straight (which differs for the deployed robot types). In Fig. 7b, the cost value is represented by the gray level of the object border cells. Hereby, the costs of the overlapping border areas between the inflated segmented objects are the sum of the corresponding action costs of the objects, which means that in these regions several object actions will be necessary.

B. *Path Planning*

Since the cost map contains the information about all obstacles, i.e, static and not static, and encodes the potential

TABLE II

Classification metrics on validation set for different input image resolutions.

Resolution	mAP	Per-class AP							mIoU
		Ball	Books	Boxes	Cars	Dolls	Stuffed toys	Toy blocks	
320×240	0.792	0.908	0.747	0.733	0.768	0.801	0.828	0.759	0.585
640×480	0.875	0.946	0.878	0.876	0.847	0.874	0.874	0.831	0.715

TABLE III

Runtime for segmentation at different image resolutions.

Resolution	Runtime	
	GTX1080Ti	Jetson TX2
320×240	10 ms (100 FPS)	89 ms (11 FPS)
640×480	33 ms (30 FPS)	245 ms (4 FPS)

object actions and their associated costs, we can efficiently use A* search with the Euclidean distance heuristic function to find a path for the robot on the cost grid. If the path leads through any object area, the corresponding class and, thus, the action can be derived from the object database. In the example shown in Fig. 6 and Fig. 7, our approach computes the green path with the action to step over the toy blocks.

C. Plan Execution

The execution of the computed 2D path then depends on the necessary actions involved.

No objects: If the path does not contain any object crossings, the robot's walking controller follows the 2D grid path.

Push: If the path crosses an object that needs to be pushed away, the robot follows the 2D path to the last free grid cell before the crossing and starts the pushing action via the plan execution. Thus, the robot senses the object locally using the segmented point cloud, moves to a target position relative to the object, and pushes the object out of its way in forward direction.

Step over and step onto: If the path traverses an area with objects that need to be stepped over or onto, the robot applies footstep planning in the corresponding region on a height map computed from the point cloud using our previous work [20].

Pick up: If the path contains an object that needs to be picked up and moved out of the way, the robot again follows the 2D path to the last free cell before it crosses the object and then identifies the object in the segmented point cloud, finds the target position relative to the object, grabs it, rotates 180° and puts the object onto the ground.

After the object action has been executed, the robot updates the cost grid and replans its path.

VI. Experimental Evaluation

In this section, we present experiments to evaluate the performance of the CNN-based classification framework (Sec. VI-A), to demonstrate the performance of our planner in a larger simulated environment (Sec. VI-B), and to show the real-world applicability with Nao humanoid (Sec. VI-C). Throughout this section, the illustrated grid maps are similar to Fig. 7a, i.e., they show the inflated object classes (rather than the actual costs) for better visualization. The resolution of the grid maps was set to 5 cm for all experiments.

A. Classification Results

As detailed in Sec. IV-B, we train a semantic segmentation CNN with 5,000 images, generated from a database of 20,000 different object instances, and over 300 backgrounds. The network is then evaluated on a test set containing 1,000 real-world images collected in our lab and pixel-wise annotated. Semantic segmentation approaches which assign a label to each pixel in an image are typically evaluated using the mean Jaccard index, also called mean intersection over union (mIoU), defined as

$$mIoU = \frac{1}{C} \sum_{i=1}^{C} \frac{tp_i}{tp_i + fp_i + fn_i},$$

where C is the number of classes, and tp, fp, and fn are the pixel-wise number of true positives, false positives, and false negatives per-class, respectively. For approaches that in the end work on objects, a commonly used measure is the mean average precision (mAP):

$$mAP = \frac{1}{C} \sum_{i=1}^{C} \frac{1}{11} \sum_{r \in \{0, 0.1, \ldots, 1\}} p_i(r),$$

where r corresponds to a value of recall in the precision-recall curve for each class, and $p_i(r)$ is the value of precision corresponding to recall r for class i. Predicted instances are defined as a positive detection when they have more than 50% IoU overlap with the ground truth mask.

In our experiments, we found that the quality of the classification depends mostly on the size of the input images, and therefore report in Tab. II the results for two resolutions of the used sensor (ASUS Xtion PRO). For a resolution of 320×240, we achieve a mAP over all classes of 0.79 and for 640×480 the mAP increases to 0.88. These results are encouraging since they show that starting from pre-trained weights, a network can be trained solely on images crawled from the Internet and few hours of human supervision to clean the dataset from improper objects present in the query results and wrong masks in the alpha channel. In future work, we will explore using a mixture of computationally intensive models to do the dataset cleaning automatically and train our lightweight CNNs. The limitations of off-line batch training in terms of labeled data collection and pre-definition of the classes are hard to circumvent and currently an open research area in computer vision.

As with the accuracy of the model, the runtime of the CNN is also highly dependent on the input resolution. In Tab. III

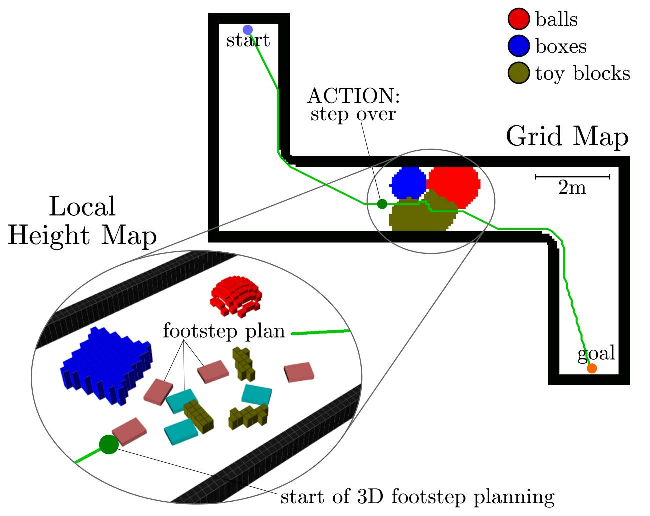

Fig. 8. Path planning through a region with several obstacles of three different classes. Our framework computes a path on a cost map that contains information about the segmented objects, which are projected onto the grid with an inflation radius (cf. grid map) and performs 3D footstep planning where necessary (cf. local height map).

we show the runtime of the model in different hardware and using different resolutions. The results show that the approach is usable also in resource-constrained hardware, such as the NVIDIA Jetson TX2, where we achieve a framerate of 4Hz-11Hz depending on the image resolution.

B. Simulation Experiment

In order to demonstrate the combination of our newly developed navigation framework with a state-of-the-art footstep planner [20], we performed an experiment in a simulated environment with a dimension of $9m \times 8m$, see Fig. 8. Here, we simulate a humanoid with a maximum step size of 30 cm forward and 20 cm sideways.

Our combined approach found the solution shown in Fig. 8 on the grid map, taking into account the object actions encoded in the cost map. The solution provided by our system involved stepping over the toy blocks and it took only 1.35 ms to compute the path using the 2D grid. For actually stepping over the toy blocks, the footstep planner computed the path in 2.83 ms. Combined, this greatly reduced the planning time to only 4.18 ms, compared to finding a full footstep plan from start to goal, which took 491 ms. All experiments in this section were performed on an *Intel Core i7-4710MQ*. We did not include the time for the semantic segmentation of the obstacles (Tab. III) since the classification runs in the parallel thread of the perception system and, thus, does not influence the planning time.

Note that in more complex scenes, more sophisticated planners (e.g., whole-body, multi-contact planners) might be necessary, which makes the planning more time consuming. In such settings, our approach is even more advantageous as it performs fast 2D grid-based planning on a cost map of the classified obstacles to generate paths containing necessary object manipulation actions, which are planned separately. In this way, a humanoid using our path planning system can instantly react to changes in the environment

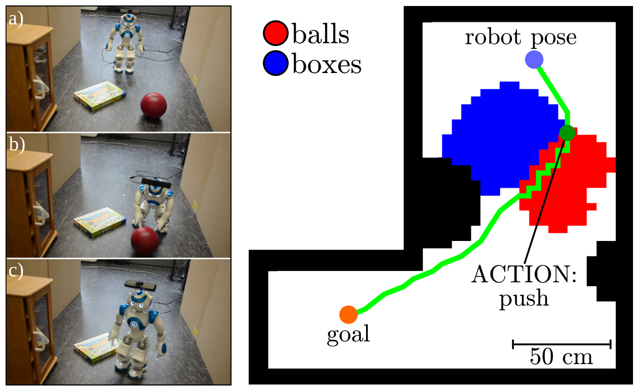

Fig. 9. Real-world experiment with a Nao robot (left) and segmented objects with inflation radius projected onto the grid map (right). *Left*: a) The robot detects the box and the ball in its way toward the goal and decides to push the ball in order to clear the path since this is the cheapest path on the corresponding cost map. b) After the robot has followed the path close to the object, it performs the pushing action. c) The robot continues walking along the path, which does not contain any further objects.

C. Real-World Experiments

We performed two real-world experiments with the Nao robot equipped with a ASUS Xtion PRO to show the full capability of our navigation framework. For 6D localization we apply Monte Carlo localization as developed by Hornung *et al.* [21] and extended by Maier *et al.* [22] for depth camera data.

In the situation depicted in Fig. 9a, the robot detected a box and a ball that obstructed the way. To reach the goal location, the robot could either push the ball aside or step onto the box before continuing walking. Based on the computed cost map our planner found a path across the ball. The robot followed the path to the vicinity of the ball and then performed a push action (see Fig. 9b). After pushing the ball aside, the path to the goal was free and the robot continued walking (see Fig. 9c.)

The experiment in Fig. 10 shows a scenario with two different routes to the goal. Since the balls were too large for the Nao to step over them (Fig. 10a), the robot could either push the balls out of the way or take a detour around the large central obstacle. Our planner found a path that included the detour, since pushing multiple balls out of the way would have required more effort and, thus, had higher associated costs. In Fig. 10b another obstructing object, in this case a stuffed toy, was detected along the detour, where our planner decided to perform a pick up action to clear the path.

These experiments demonstrate the advantages of our planning system as opposed to a existing footstep or whole-body-motion planners, as none of these approaches would be capable of finding a solution to the goal.

VII. Remarks

Instead of mapping only one action to each object class, several actions may be used. The different possible action types per class may be determined more accurately by using detailed point cloud data. This would be advantageous, e.g., in the scenario in Fig. 10a, where stepping over the balls might be a possibility if the balls were small enough.

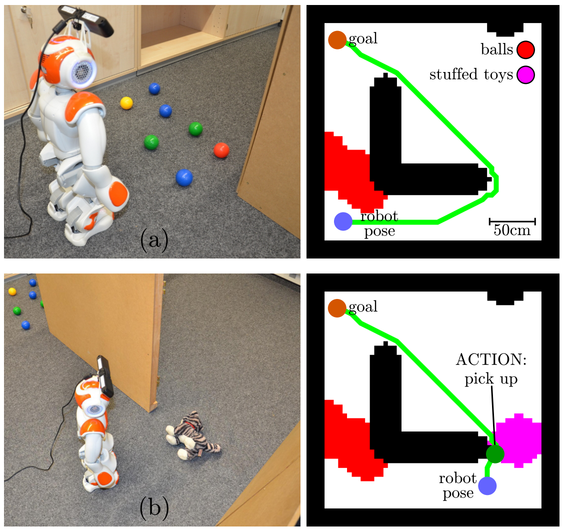

Fig. 10. Real-world experiment with a Nao humanoid (left) and segmented objects with inflation radius projected onto the grid map (right). a) The robot perceives seven objects classified as balls in its way and decides to take the detour around the L-shaped static obstacle since this appears to be the cheapest solution according to the cost map. b) On its way to the goal, the robot detects a further object (in this case a stuffed toy). Our framework now decides to pick up the object to reach the goal location.

Note that the path planner generally provides only suggestions regarding the action types. Should the execution module not find a solution with any of the proposed actions, e.g., due to object-environment configuration or classification errors, the object would be marked as impassible such that our planner would seek a different solution.

VIII. Conclusion

In this paper, we proposed a novel framework that exploits the knowledge about obstacle classes during path planning. We trained a convolutional neural network to distinguish different object classes and use this information to construct a cost grid during navigation. The cost grid represents the static obstacles in the environment as well as the costs of actions that need to be carried out by the robot to cross regions containing observed obstacles, i.e., step over or onto objects, push an object, or pick it up and put it aside. During navigation, the robot then uses the cost grid to efficiently (re)plan its path to the goal location, which implicitly contains all necessary actions.

As we showed in various experiments, the trained neural network is able to robustly distinguish between the different obstacle classes. Furthermore, we demonstrated that a Nao robot can exploit the knowledge about classified obstacles during navigation and efficiently find appropriate actions to deal with the objects. Finally, we illustrated that more complex planning frameworks can be integrated with our system to efficiently find paths through challenging regions.

References

[1] M. Grey, A. Ames, and C. Liu, "Footstep and motion planning in semi-unstructured environments using randomized possibility graphs," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2017.

[2] Y. Lin and D. Berenson, "Humanoid navigation in uneven terrain using learned estimates of traversability," in *Proc. of the IEEE Intl. Conf. on Humanoid Robots*, 2017.

[3] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *Proc. of the IEEE Intl. Conf. on Humanoid Robots*, 2014.

[4] M. Fallon, P. Marion, R. Deits, T. Whelan, M. Antone, J. McDonald, and R. Tedrake, "Continuous humanoid locomotion over uneven terrain using stereo fusion," in *Proc. of the IEEE Intl. Conf. on Humanoid Robots*, 2015.

[5] A. Hildebrandt, M. Klischat, D. Wahrmann, R. Wittmann, F. Sygulla, P. Seiwald, D. Rixen, and T. Buschmann, "Real-time path planning in unknown environments for bipedal robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, Oct 2017.

[6] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with rao-blackwellized particle filters," *IEEE transactions on Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[7] A. Milioto and C. Stachniss, "Bonnet: An open-source training and deployment framework for semantic segmentation in robotics using cnns," *Workshop on Perception, Inference, and Learning for Joint Semantic, Geometric, and Physical Understanding, IEEE Int. Conf. on Robotics & Automation (ICRA)*, May 2018.

[8] M. Stilman and J. Kuffner, "Navigation among movable obstacles: Real-time reasoning in complex environments," *Intl. Journal of Robotics Research (IJRR)*, vol. 2, no. 4, 2005.

[9] M. Stilman, K. Nishiwaki, S. Kagami, and J. Kuffner, "Planning and executing navigation among movable obstacles," in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2006.

[10] M. Levihn, K. Nishiwaki, S. Kagami, and M. Stilman, "Autonomous environment manipulation to assist humanoid locomotion," in *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2014.

[11] A. Hornung, S. Boettcher, C. Dornhege, A. Hertle, J. Schlagenhauf, and M. Bennewitz, "Mobile manipulation in cluttered environments with humanoids: Integrated perception, task planning, and action execution," in *Proc. of the IEEE Intl. Conf. on Humanoid Robots*, 2014.

[12] P. Kaiser, D. Gonzalez-Aguirre, F. Schueltje, J. Borras, N. Vahrenkamp, and T. Asfour, "Extracting whole-body affordances from multimodal exploration," in *Proc. of the IEEE Intl. Conf. on Humanoid Robots*, 2014.

[13] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *arXiv preprint*, vol. abs/1606.00915, 2016.

[14] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," *arXiv preprint*, vol. abs/1612.01105, 2016.

[16] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "ENet: Deep neural network architecture for real-time semantic segmentation," *arXiv preprint*, vol. 1606.02147, 2016.

[17] E. Romera, J. Alvarez, L. Bergasa, and R. Arroyo, "ERFNet: Efficient residual factorized convnet for real-time semantic segmentation," *IEEE Trans. on Intelligent Transportation Systems (ITS)*, vol. 19, no. 1, pp. 263–272, 2018.

[18] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation," *arXiv preprint*, 2018.

[19] T. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. Zitnick, "Microsoft COCO: Common objects in context," *CoRR*, vol. abs/1405.0312, 2014.

[20] P. Karkowski, S. Oßwald, and M. Bennewitz, "Real-time footstep planning in 3d environments," in *Proc. of the IEEE Intl. Conf. on Humanoid Robots*, 2016.

[21] A. Hornung, K. M. Wurm, and M. Bennewitz, "Humanoid robot localization in complex indoor environments." in *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2010.

[22] D. Maier, A. Hornung, and M. Bennewitz, "Real-time navigation in 3d environments based on depth camera data," in *Proc. of the IEEE Intl. Conf. on Humanoid Robots*, 2012.