Predicting Obstacle Footprints from 2D Occupancy Maps by Learning from Physical Interactions

Marina Kollmitz[1] Daniel Büscher[1] Wolfram Burgard[1,2]

***Abstract*— Horizontally scanning 2D laser rangefinders are a popular approach for indoor robot localization because of the high accuracy of the sensors and the compactness of the required 2D maps. As the scanners in this configuration only provide information about one slice of the environment, the measurements typically do not capture the full extent of a large variety of obstacles, including chairs or tables. Accordingly, obstacle avoidance based on laser scanners mounted in such a fashion is likely to fail. In this paper, we propose a learning-based approach to predict collisions in 2D occupancy maps. Our approach is based on a convolutional neural network which is trained on a 2D occupancy map and collision events recorded with a bumper while the robot is navigating in its environment. As the network operates on local structures only, it can generalize to new environments. In addition, the robot can collect and integrate new collision examples after an initial training phase. Extensive experiments carried out in simulation and a realistic real-world environment confirm that our approach allows robots to learn from collision events to avoid collisions in the future.**

I. Introduction

Robots are increasingly employed in spaces designed for and populated by humans. These environments are typically cluttered and may change over time. At the same time, we demand rising levels of autonomy and decreasing costs from personal robots and robots employed in public spaces. This paper presents a novel approach to predicting collisions with parts of objects that cannot be perceived by a mobile robot due to the specific mounting of its sensors.

Throughout this paper, we focus on horizontally scanning 2D laser rangefinders, which are popular for autonomous indoor navigation. Horizontally scanning laser rangefinders only perceive one slice of the environment and therefore typically miss large parts of obstacles. Accordingly, the resulting 2D occupancy maps typically do not correctly resemble the free space in the environment. One example is depicted in Fig. 1. As large parts of the tables in the room are not visible to the laser rangefinder, many planned navigation paths based on the 2D occupancy map lead to collisions. One possible solution is to use additional sensors, such as RGB-D cameras or 3D Lidar sensors. However, additional sensors increase the computational cost and power consumption, and most cannot reliably perceive all obstacles either, like glass surfaces. Bumper sensors provide a reliable method for perceiving collisions outside the planar perceptive field of laser rangefinders. While they are cheap and have very low computational and power requirements, they cannot prevent collisions; the robot can only react appropriately to them. In addition, inferring and maintaining occupancy from bumpers is not straight forward if the environment is changing over time, since occupied space is never updated if the robot avoids it.

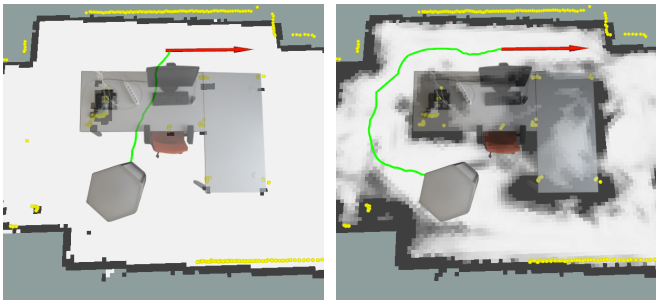

Fig. 1. Navigation based on 2D occupancy information can result in collisions for scenarios where only parts of the objects are visible (left). Our approach predicts the obstacle footprints from 2D occupancy maps and can, therefore, avoid collisions with parts of the objects that cannot be perceived by the sensor (right).

We propose a learning method to predict collisions in 2D occupancy maps. Our approach uses a convolutional neural network which is trained based on collision events recorded with bumpers. The network predicts map areas that will likely result in collision (see Fig. 1). With our approach, the robot can anticipate collision areas and plan paths to avoid them. We introduce a collision dataset recorded with a simulated robot in diverse simulated indoor scenes, which we used to train an initial model for collision prediction. We further demonstrate how the robot can combine simulated data with collision examples from real-world scenarios to train models for new environments. The contributions of our work are as follows:

- a novel approach to predict collisions in 2D occupancy maps learned from collision events
- a network structure that allows training on binary collision events as well as fast, efficient and high-resolution occupancy map processing
- a dataset for 2D map learning that is tailored for collision-free indoor mobile robot navigation with 2D laser rangefinders

Our approach presents the first work for collision prediction in the indoor mobile robotics context that allows the robot to collect and integrate new training examples after an initial training phase since the robot

*This work has been partially supported by the German Federal Ministry of Education and Research (BMBF), contract number 01IS15044B-NaRKo.

[1]Department of Computer Science, University of Freiburg, Germany. {kollmitz, buescher, burgard}@informatik.uni-freiburg.de

[2]Toyota Research Institute, Los Altos, USA

can collect collision examples with the onboard bumper sensor in a self-supervised fashion. Thus, the robot can adapt to new environments by learning better models to represent them. Our simulated dataset and the code for our work are available at `https://github.com/marinaKollmitz/learn-collisions`.

II. Related Work

Robotic systems rely on adequate models of the environments they interact with to operate safely and efficiently. However, sensor data is noisy and typically incomplete. Therefore, a lot of research is concerned with anticipating and predicting missing sensor information, e.g., 3D semantic scene completion [1] and 3D object reconstruction [2]. Another example is the work of Ondrúška and Posner [3], who predicted occluded objects from laserscanner data.

In the context of mobile robot navigation, Burgard et al. [4] presented an approach for laser rangefinder-based obstacle avoidance in the presence of partially invisible obstacles. The robot was able to avoid collisions with parts of obstacles that the sensors could not perceive by using an adapted version of the dynamic window approach [5], μDWA [6]. However, a full model of the environment, including all obstacles, had to be provided. Axelrod et al. [7] considered robot navigation in the presence of obstacle uncertainty, but their approach is not suitable when large parts of the obstacles are invisible to the sensor.

Thrun [8] used a fully connected neural network to estimate the occupancy in maps using sonar sensor readings as inputs. They also used simulated data to train their model, but the approach was also not designed for cases where large parts of the obstacles are invisible. Guizilini and Ramos [9] learned to reconstruct 2D and 3D structures for occupancy mapping. They trained a convolutional variational autoencoder to recover data gaps, but they did not consider partially detectable objects.

Various works used additional sensors for obstacle avoidance, such as RGB-D sensors [10, 11]. Baltzakis et al. [12] fused laser and visual data to perceive parts of objects which were not visible in the laser data alone. The approaches rely on additional sensors with high computational and power demand, while we learn from collision events. Plagemann et al. [13] used Gaussian process classification and regression techniques to detect collisions with unseen obstacles. While their approach can perceive collisions when they occur, it cannot anticipate them beforehand.

The work by Lundell et al. [14] is closest related to ours. The authors used a fully convolutional autoencoder to predict laser rangefinder readings with true obstacle distances from 2D laser scans. Their later work [15] integrated the processed laser scans into occupancy maps with uncertainty estimation. The true obstacle distances for training were collected with a 3D camera. The authors showed that their approach can avoid collisions with obstacles in realistic navigation scenarios. However, it relies on an additional sensor for generating the training examples, which is removed after the training phase. Our approach, however, allows to integrate further training examples after the initial training phase and thus to adapt to the environment over time.

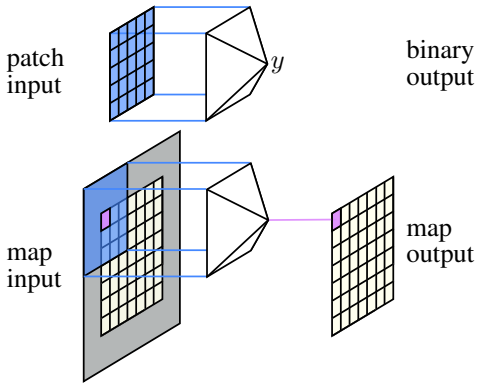

Fig. 2. Our neural network structure can be used for binary classification on image patches (top). During inference on map inputs, the network performs image segmentation and outputs full resolution maps (bottom).

Our approach relies on image segmentation techniques to process 2D occupancy maps efficiently. Image segmentation, especially semantic segmentation, is a very active research field in which convolutional neural networks have caused large gains in performance over the last years [16, 17, 18, 19, 20, 21, 22]. We predict collisions based on 2D occupancy maps, which differ from usual image data because of their small size and resolution. Because of the already small spatial resolution, we want to avoid further downsampling of the occupancy maps. Various approaches aim to maintain a large output resolution [20, 21]. Dilated convolutions [22] have been used in semantic segmentation to reduce the amount of downsampling in segmentation networks [22, 19], like in our work. We base our approach on the Fully Convolutional Network (FCN) model [16] and use dilated convolutions to eliminate output downsampling. The FCN model uses convolutionalized versions of CNNs designed originally for object detection. The FCN paradigm is well suited for our approach because we can train our model as a classification network on binary collision events and still apply it efficiently for segmentation, as described in the next section.

III. Method

Our goal is to segment 2D occupancy maps into collision space and free space. At the same time, we want to learn from binary collision events recorded by the robot in a self-supervised fashion. Our approach is based on a neural network that is suited both for binary classification and image segmentation, as depicted in Fig. 2. When applied to image patches, the network output is binary. We can hence train the network on binary collision events, using occupancy map patches as input. When applied to full occupancy maps, the network efficiently slides over the input and outputs segmented collision maps.

To achieve a network that can perform both classification and segmentation, we follow the fully convolutional network (FCN) paradigm presented by Long et al. [16]. They

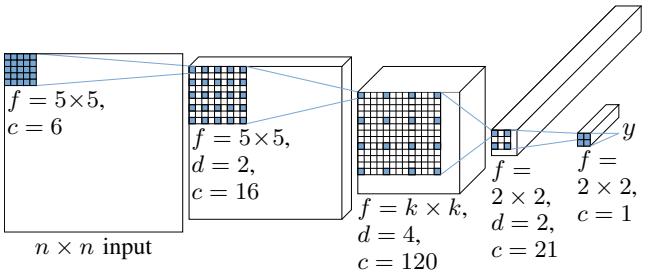

Fig. 3. Dilated convolution layers replace convolution and pooling in our adapted network structure. The dilated layers enlarge the receptive field and reduce the number of parameters. The kernel size f, dilation factor d and the number of filters c are specified for each layer.

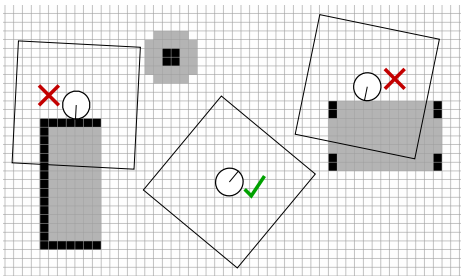

Fig. 4. Our collision classifier takes patches from the 2D occupancy map to classify map poses into collision (red cross) and free (green checkmark). The gray parts of the obstacle footprints are not visible in the 2D occupancy map and are to be predicted by our approach.

proposed to convolutionalize the fully connected layers of CNNs originally designed for classification. To this end, the fully connected layers are replaced by equivalent convolution layers with kernels that span over all input neurons of the layer. As fully convolutional structures, FCNs can efficiently convolve arbitrary-sized inputs and thus segment them into class heat maps, as visualized in Fig. 2 (bottom). For input patches that exactly match the network input shapes, FCNs output class predictions like the original classification CNNs (Fig. 2 top). Image segmentation with FCNs is equivalent to patch-wise processing of the input image, but it is much more efficient because computations are shared over overlapping patches.

FCNs typically downsample the input image due to striding in the network, e.g. by pooling layers. However, such a decrease in resolution is problematic for our use-case, because we already operate on low-resolution occupancy maps. In the original FCN paper, Long et al. [16] proposed to add deconvolution layers with skip connections for upsampling the class heatmaps. However, the resulting structure cannot be trained as a binary classifier any more. Instead, we replace the pooling layers by dilated convolutions [22] to maintain full map resolution during inference. Dilated convolutions have spaces between the kernel neurons, where the dilation factor determines the number of skipped neurons. They span larger input regions than conventional convolution filters with the same number of neurons. Thus, they increase the receptive field in the same way as pooling layers, but they do not downsample the input.

Our network architecture is visualized in Fig. 3. We base our architecture on the LeNet model [23] because it is fast and showed superior performance on the MNIST handwriting dataset [23], which is similar to our type of data. The architecture is a fully convolutional version of LeNet with dilated convolutions instead of pooling. Furthermore, we use a sparse connectivity throughout the network to keep the number of parameters similar to the original LeNet.

The first convolutional layer is equal to the original LeNet; the second uses a dilation of 2. Without the pooling layers, the feature map size after the convolutional part is larger compared to the original LeNet. Therefore, we use a convolution filter with a dilation of 4 after the second layer. The filter size of the third layer $k \times k$ depends on the size of the network input $n \times n$ and is calculated by $k = {}^{n}/_{4} - 3$. Note that the feature map of the original LeNet after the convolutional part is also $k \times k$. Thus, the dilated layer has the same number of parameters. The last two layers of our network are again independent of the output shape. Note that we add dilations and reduce the number of convolution channels compared to the original LeNet to keep the number of parameters similar.

Our architecture can be used for classification as well as segmentation of images at the full input resolution. In the following, we will explain in more detail how it can be trained on collision events and how it segments full occupancy maps.

A. Learning from Collision Events

As visualized in Fig. 4, we train our network to classify patches of the occupancy map. Each input patch is centered around a map pose $\mathbf{S} = (x, y, \theta)^{\mathrm{T}}$, and the network outputs the probability that the map pose $\mathbf{S}$ is in collision as $m_{\mathbf{S}} = p(\mathbf{S}) \in [0, 1]$.

The parameters ϕ of the network are optimized using stochastic gradient descent according to

$$\phi^* = \underset{\phi}{\operatorname{argmin}} \sum_{i=1}^{N} \mathcal{L}\left(\hat{m}_{\mathbf{S}}^i, m_{\mathbf{S}}^i\right). \tag{1}$$

The number of training examples is N, and $\hat{m}_{\mathbf{S}}$ denotes the collision label that specifies whether the robot perceived a collision at pose $\mathbf{S}$ from which the input patch was generated. The loss function $\mathcal{L}$ is a weighted cross-entropy loss,

$$\mathcal{L}\left(\hat{m}_{\mathbf{S}}^i, m_{\mathbf{S}}^i\right) = \frac{1}{N} \sum_{i=1}^{N} w \cdot \hat{m}_{\mathbf{S}}^i \log m_{\mathbf{S}}^i + (1 - \hat{m}_{\mathbf{S}}^i) \log(1 - m_{\mathbf{S}}^i). \tag{2}$$

The factor w increases the contribution of the positive training examples, which are less frequent in the dataset.

B. Segmenting 2D Occupancy Maps

Due to the fully convolutional structure, the trained collision classification network can segment full 2D occupancy maps without further modifications. When applied for segmentation, we first need to pad the input image as visualized by the gray border in Fig. 2 to account for the shrinkage at the image borders caused by the network size itself. Since the

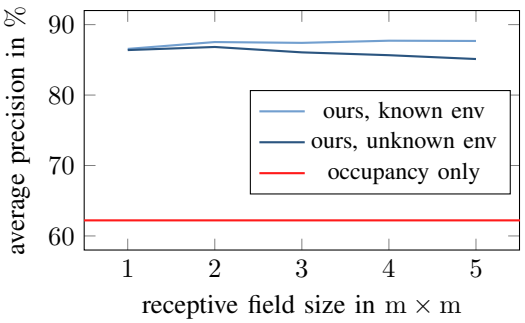

Fig. 5. Average precision for varying receptive field sizes for our approach, compared to using the occupancy map information alone.

collision poses for training included orientation information, we process 8 equidistant orientations for each occupancy map. To this end, we rotate the map by $\theta_k = k \cdot 45°, k = \{0, ..., 7\}$, for predicting the collision map $m_{\theta,k}$. We then take the maximum predicted collision probability for each cell as the final map value $m = \max m_{\theta,k}$.

IV. Experiments

We devised a set of experiments to evaluate the ability of our network to predict collisions in 2D occupancy maps. Sec. IV-A introduces our simulated collision dataset, and Sec. IV-B describes our training setup. The classification and segmentation capabilities are evaluated in Sec. IV-C and Sec. IV-D, respectively. Finally, Sec. IV-E shows a real-world scenario in which a robot combines simulated data with new collision events to train updated models for the environment.

A. Simulated Collision Dataset

We use the SceneNet [24] synthetic indoor scenes dataset to train and test our approach. The SceneNet dataset consists of 3D models of 59 indoor scenes from 5 room categories with different furniture items and room layouts. We removed a total of 7 rooms from our set because of missing furniture parts or different model scales and divided the remaining into 9 rooms for testing, 9 for validation, and 34 for training.

We used a simulated version of our robot Canny to explore the simulated environments and collect collision examples. Like the real version, the simulated Canny can detect collisions with a force-sensitive shell [25] that perceives the magnitude, impact point, and direction of collision forces. To collect collision data, we teleported the robot to random poses in free areas of the simulated environment and drove it forward until it encountered a collision. We then saved the pose of each perceived collision as a positive collision example. During collision-free motion, we saved non-collision examples at regular time intervals of 1 s. Here, we selected the front corner of the robot and sampled one configuration inside the footprint. We collected a total of 206.532 examples from the train and validation rooms, which corresponds to almost 33 hours of exploration.

We generated ground truth collision maps by marking whether the cells intersect with the simulated environment models. Intersections were checked with the simulation physics engine. Note that collisions only occur at the object borders. Therefore, we performed a wavefront exploration from the free space inside each environment to find and mark the object borders as collision. Areas inside objects or outside the environments do not have a valid label and are ignored for evaluation. We further generated 2D occupancy maps for each environment. To generate realistic occupancy maps, we simulated noisy laser rangefinder beam arrays and integrated all measurements using the counting model [26] to produce maximum-likelihood grid maps. Example simulated environments, together with the generated occupancy maps and ground truth collision maps, are depicted in Fig. 6.

TABLE I

Performance in terms of precision, recall, average precision and map processing time.

	prec.	recall	AP	GPU time [s]
occupancy only	87.90	53.27	49.31	-
LeNet + patch classification	34.47	82.22	51.70	181.07
FCN LeNet + bilinear upsample	31.06	81.40	35.50	0.0825
dilated FCN LeNet (ours)	33.32	82.33	55.66	0.3118

B. Training Setup

We implemented the collision networks in PyTorch. To increase the impact of the less frequent positive collision examples, we used a weight of $w = 2$ (Eq. (2)). We used stochastic gradient descent with an initial learning rate of 0.025, reduced by half after each epoch, and trained for 10 epochs on mini-batches with 32 examples.

C. Evaluation of Collision Classification

The first experiment evaluates the performance of our binary classifier on our simulated collision dataset. We performed 3-fold cross-validation on the training set. The leave-out set is not part of the training, but it is composed of collisions samples collected in the same environments used for training. We will refer to the leave-out set as *known env* set. The final model is trained on the entire training set and tested on the validation set. The validation set examples stem from unseen environments, referred to as *unknown env*.

To analyze the influence of the network receptive field on the classification performance, we vary the network input sizes during training and testing between 20×20 and 100×100 pixels, corresponding to 1×1 m^2 and 5×5 m^2 at 0.05 m map resolution. Note that the receptive field size does not influence the prediction resolution, since the network always estimates only the center pixel (Fig. 2). We compare our network performance to the naive "occupancy only" baseline where we classify a pose as collision if the corresponding cell in the occupancy map is occupied and as free otherwise. Fig. 5 shows the performance of our approach and the occupancy only baseline in terms of average precision for varying network input sizes. Even for small receptive fields, our approach outperforms the baseline by a large margin. This confirms that our approach can learn to predict collisions in 2D occupancy maps from

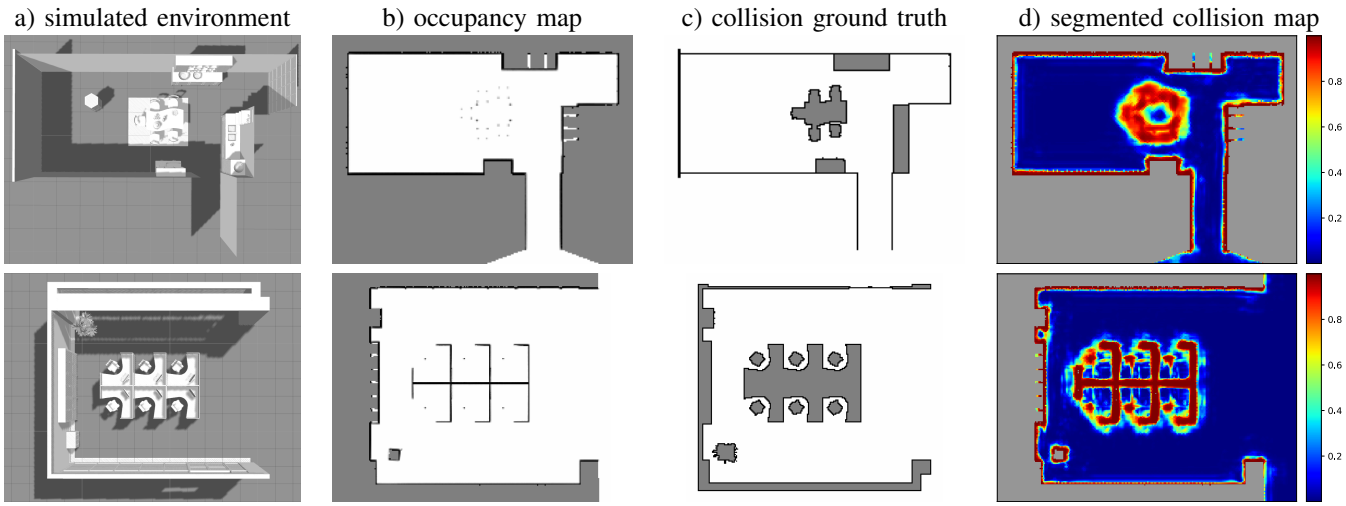

Fig. 6. Example simulation environments used for data generation with occupancy and ground truth collision maps and segmentation outputs of our approach.

collision events. For the *unknown env* set, the performance slightly drops for receptive fields larger than 2×2 m^2. The drop could indicate that collision examples that rely on local features, captured by small receptive fields, are similar between the known and unknown environments. However, more complex examples, captured only by larger receptive fields, differ between the sets. Therefore, a larger receptive field can cause a loss in performance by overfitting to specific environments. The gain in performance with larger receptive fields on the *known env* set, where the network can exploit more information from known arrangements of objects, also supports this hypothesis.

The receptive field choice depends on the environment complexity and is always a trade-off between performance and inference speed. In the following, we will show results for the 3×3 m^2 receptive field network only.

D. Evaluation of Occupancy Map Segmentation

The second experiment evaluates the ability of our approach to segment occupancy maps. We test on the occupancy maps in the *test* split, which was not part of the training set. To evaluate the collision map reconstruction performance, we calculate pixel-wise precision and recall and regard a cell as in collision if the predicted collision probability is larger than 0.5. We further compare the average precision score and the runtime on an Nvidia Titan Black GPU, normalized for a 100×100 pixel input. The ground truth collision map was generated from simulation, as described in Sec. IV-A. As before, we compare our approach with the occupancy only baseline where the collision probability of each cell corresponds to its occupancy value. We also evaluate the map processing performance of other network variants to test the impact of our network adaptations from the original LeNet. We first compare our approach to the original LeNet with patch processing. Instead of convolving the network over the input image, all cells are processed individually by sampling image patches. The second variant is a fully convolutional version of the original LeNet, including the pooling layers. The pooling layers downsample the input image by a factor of 4. Therefore, we perform bilinear upsampling to yield the full map output size. Note that we do not consider larger networks like AlexNet [27], VGG [28] or ResNet [29] because their large input dimensions make them unsuitable for our low-resolution map data. Furthermore, they strongly downsample the input due to the many pooling layers.

Tab. I shows the map segmentation performance. All architecture variants show a drop in precision but a notable improvement in recall compared to the occupancy only baseline. The occupancy only baseline represents the pixels that are definitely in collision, resulting in a high precision score. However, the low recall shows that it misses large parts of the actual collision space. The networks tend to overestimate the collision space, hence the drop in precision, but identify many previously missed collision areas, as shown by the higher recall. The LeNet variant with patch classification performs comparably to our approach, but the runtime is orders of magnitude higher. In contrast, the fully convolutional LeNet yields very fast processing speeds. However, the down- and upsampling causes a notable segmentation performance drop. Our approach combines both fast processing speeds with high segmentation scores and even outperforms patch classification in terms of average precision.

This experiment confirms that our approach can process occupancy maps fast and that it can successfully identify areas in the map where collisions may occur. However, we can see that our approach is often too conservative and classifies areas as in collision that are actually in free space. For navigation, we argue that overcautious obstacle avoidance is preferable to risking collisions. However, overestimating occupancy can lead to incomplete path planning in tight spaces. In future work, we plan to investigate exploration/exploitation strategies so the robot can test and

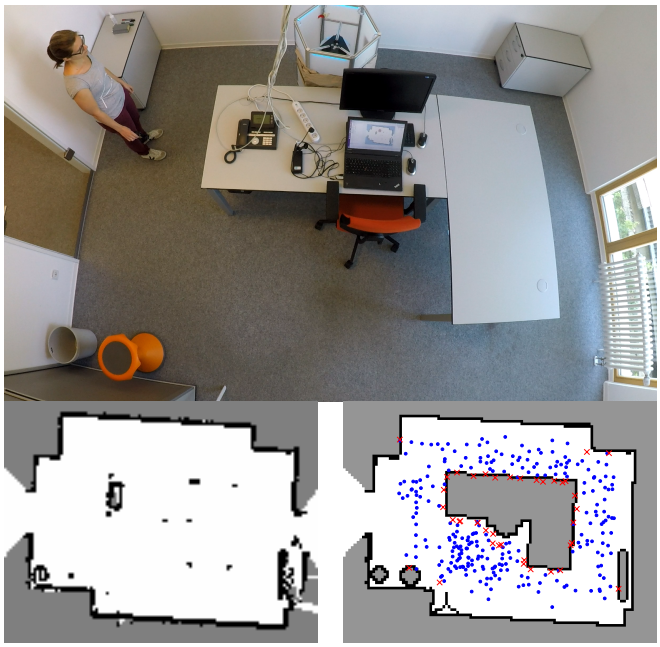

Fig. 7. Top: Office environment for the real-robot experiment. Our force sensitive robot Canny is collecting new collision examples for improving the environment model. Bottom: 2D Occupancy map (left) and collision map with recorded collision poses (right).

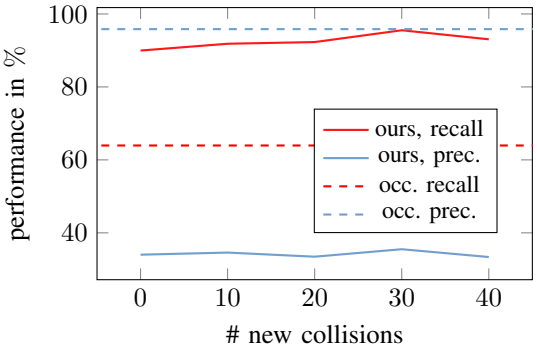

Fig. 8. Map segmentation performance for the real office environment vs. number of integrated new collision examples.

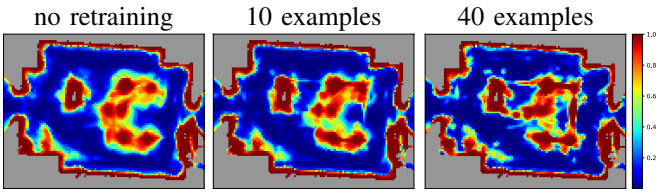

Fig. 9. Processed collision maps before retraining and after integrating 10 and 40 collision examples from the new environment.

confirm if map areas are actually occupied. Fig. 6 shows example occupancy and ground truth collision maps as well as collision maps generated by our approach for two simulated environments.

E. Real Robot Experiment with Canny

Finally, we show the performance of our network for a real-robot scenario. Our force-sensitive robot Canny operated in the previously unseen office environment depicted in Fig. 7. Also shown are the occupancy and the manually annotated ground truth collision maps. The environment is challenging because a large portion of the objects is not visible in the 2D occupancy map: e.g., only the table legs are visible. This experiment evaluates both the ability of our approach to generalize from simulation to the real robot as well as the retraining performance on new collision examples. For retraining, we collected 40 new collision examples by colliding the robot with the room furniture. We first trained our approach on the training set from simulation and iteratively retrain with the new collision examples. We add the new collision examples to the training set and retrain the model from scratch, as described in Sec. IV-B. To give them more weight, we added 100 copies of the new collision examples to the training set.

Fig. 8 shows the performance of our approach after retraining with varying numbers of collision examples, compared to the occupancy only baseline. The segmented collision maps of our approach after integrating 0, 10 and 40 collision examples are visualized in Fig. 9. As in the previous experiment, we can notice a drop in precision for our approach compared to the baseline, but the recall improved by a large margin. Without the new collision examples, we already see a greatly improved recall, which means that the robot can already anticipate a large fraction of the collisions in the new environment with the model trained in simulation. The recall further improves with a rising number of integrated collision examples, which means that the approach can improve its model over time. Fig. 9 shows qualitatively that the robot learns an improved model: The contours of the previously unseen objects are more and more visible with an increasing number of new collision examples. Fig. 1 shows two navigation paths planned between two poses in the environment: one on the original 2D occupancy map and one with our approach. While the original path results in a collision, our approach can prevent the collision and produces a safe navigation path.

V. Conclusion and Future Work

We presented a novel approach for predicting collisions in 2D occupancy maps build with horizontally scanning 2D laser rangefinders. Our approach uses a convolutional neural network trained on collision events recorded with a bumper. Our experiments confirm that the model trained on a simulated collision dataset can reliably predict collisions in 2D occupancy maps. We also show that the performance of our approach can be further improved by integrating new collision examples collected during real-world operation. In future work, we plan to incorporate exploration/exploitation techniques to confirm that the space predicted as occupied actually leads to a collision and actively guide the robot to such areas.

References

[1] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[2] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[3] P. Ondrúška and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in *National Conference on Artificial Intelligence (AAAI)*, 2016.

[4] W. Burgard, A. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, and S. Thrun, "The interactive museum tour-guide robot," in *National Conference on Artificial Intelligence (AAAI)*, 1998.

[5] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[6] D. Fox, W. Burgard, S. Thrun, and A. B. Cremers, "A hybrid collision avoidance method for mobile robots," in *International Conference on Robotics and Automation (ICRA)*, 1998, pp. 1238–1243.

[7] B. Axelrod, L. P. Kaelbling, and T. Lozano-Pérez, "Provably safe robot navigation with obstacle uncertainty," *International Journal of Robotics Research (IJRR)*, vol. 37, no. 13-14, 2018.

[8] S. Thrun, "Learning metric-topological maps for indoor mobile robot navigation," *Artificial Intelligence*, vol. 99, no. 1, 1998.

[9] V. Guizilini and F. Ramos, "Learning to reconstruct 3D structures for occupancy mapping," in *Robotics: Science and Systems (RSS)*, 2017.

[10] A. Oliver, S. Kang, B. C. Wünsche, and B. MacDonald, "Using the kinect as a navigation sensor for mobile robotics," in *Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2012.

[11] D. Maier, A. Hornung, and M. Bennewitz, "Real-time navigation in 3D environments based on depth camera data," in *International Conference on Humanoid Robots (Humanoids)*, 2012.

[12] H. Baltzakis, A. Argyros, and P. Trahanias, "Fusion of laser and visual data for robot motion planning and collision avoidance," *Machine Vision and Applications*, vol. 15, 2003.

[13] C. Plagemann, D. Fox, and W. Burgard, "Efficient failure detection on mobile robots using particle filterswith gaussian process proposals," *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2007.

[14] J. Lundell, F. Verdoja, and V. Kyrki, "Hallucinating robots: Inferring obstacle distances from partial laser measurements," in *International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[15] J. Lundell, F. Verdoja, and V. Kyrki, "Deep network uncertainty maps for indoor navigation," *arXiv preprint arXiv:1809.04891*, 2018.

[16] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[17] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 12, 2017.

[18] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[19] L. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *Computing Research Repository (CoRR)*, vol. abs/1706.05587, 2017.

[20] A. Paszke, A. Chaurasia, S. Kim, and E. Culurciello, "Enet: A deep neural network architecture for real-time semantic segmentation," *Computing Research Repository (CoRR)*, vol. abs/1606.02147, 2016.

[21] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe, "Full-resolution residual networks for semantic segmentation in street scenes," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *International Conference on Learning Representations (ICLR)*, 2016.

[23] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, 1998.

[24] A. Handa, V. Pătrăucean, S. Stent, and R. Cipolla, "Scenenet: An annotated model generator for indoor scene understanding," in *International Conference on Robotics and Automation (ICRA)*, 2016.

[25] M. Kollmitz, D. Büscher, T. Schubert, and W. Burgard, "Whole-body sensory concept for compliant mobile robots," in *Intermaional Conference on Robotics and Automation (ICRA)*, 2018.

[26] D. Hähnel, R. Triebel, W. Burgard, and S. Thrun, "Map building with mobile robots in dynamic environments," in *International Conference on Robotics and Automation (ICRA)*, 2003.

[27] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Neural Information Processing Systems (NIPS)*, 2012.

[28] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

[29] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.