2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

2021 IEEE International Conference on Robotics and Automation (ICRA) | 978-1-7281-9077-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICRA48506.2021.9561034

High-Speed Robot Navigation using Predicted Occupancy Maps

Kapil D. Katyal[1,2], Adam Polevoy[1], Joseph Moore[1], Craig Knuth[1], Katie M. Popek[1]

***Abstract*— Safe and high-speed navigation is a key enabling capability for real world deployment of robotic systems. A significant limitation of existing approaches is the computational bottleneck associated with explicit mapping and the limited field of view (FOV) of existing sensor technologies. In this paper, we study algorithmic approaches that allow the robot to predict spaces extending beyond the sensor horizon for robust planning at high speeds. We accomplish this using a generative neural network trained from real-world data without requiring human annotated labels. Further, we extend our existing control algorithms to support leveraging the predicted spaces to improve collision-free planning and navigation at high speeds. Our experiments are conducted on a physical robot based on the MIT race car using an RGBD sensor where were able to demonstrate improved performance at 4 m/s compared to a controller not operating on predicted regions of the map.**

I. Introduction

A key objective of mobile robotic systems is to execute safe, reliable motion while avoiding obstacles in the shortest amount of time possible. While mobile robots have demonstrated considerable success in recent years, they still fail to maneuver in environments at the speed and agility of humans. Traditional navigation algorithms require explicit perception, mapping, localization and control for collision free motion. Often, high-speed navigation using these traditional approaches is severely limited by the sensor's field of view (FOV) as well as the computational requirements needed for explicit mapping. This requires a robot to frequently reduce its speed to rescan the environment, construct a map and replan a new trajectory.

This is in contrast to human navigation where cognitive psychologists have hypothesized that humans (i) actively differentiate between occupied and free spaces based on observations, (ii) make predictions of occupied spaces beyond line of sight and (iii) use these predictions for navigation to help improve robustness and agility [1], [2], [3].

In this paper, we lay the foundation for developing algorithms that provide these capabilities to robotic systems. Our intuition is that as humans navigate, they leverage spatial cues within the environment to generate predictions of future spaces and use that as part of the planning process. Our objective is to mimic this predictive capability in robotic systems (see Fig. 1). Our approach similarly learns to predict spaces beyond the line of sight and further uses the predicted areas as part of the robot controller for planning.

[1]Johns Hopkins University Applied Physics Lab, Laurel, MD, USA. Kapil.Katyal@jhuapl.edu, Adam.Polevoy@jhuapl.edu
[2]Dept. of Comp. Sci., Johns Hopkins University, Baltimore, MD, USA.

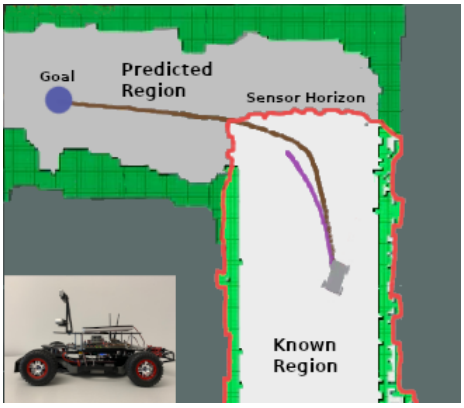

Fig. 1: A motivating example of our approach. We leverage occupancy map prediction to allow the robot to plan more robust trajectories at higher speeds compared to those limited by the sensor horizon.

Our specific contributions include:

- Novel perception algorithms that predict future occupancy maps using generative neural networks.
- A controller that leverages the predicted occupancy map during planning.
- Real-world hardware experiments using a robotic car to demonstrate higher speed navigation with improved reliability compared to a controller not operating on predicted regions of the map.

II. Related Work

Previous work has presented different strategies to predict unknown parts of the map, e.g. [4]. A variety of methods incorporate map predictions to speed exploration of an environment [5], [6], [7], [8], [9]. Predominantly, these methods are applied to exploration and therefore do not stress the capability of the map prediction module with respect to the control pipeline. In particular, this paper extends our prior work, [7], by using a balanced classification loss instead of a regression loss in addition to data augmentation and noise suppression techniques required to generate accurate predictions using real data.

An alternative to map prediction is presented in [10]; this method predicts the next best viewpoint for exploration by utilizing experience from previously mapped environments. Our method shows that map prediction effectively increases the amount of information available from the map and improves point-to-point high speed navigation.

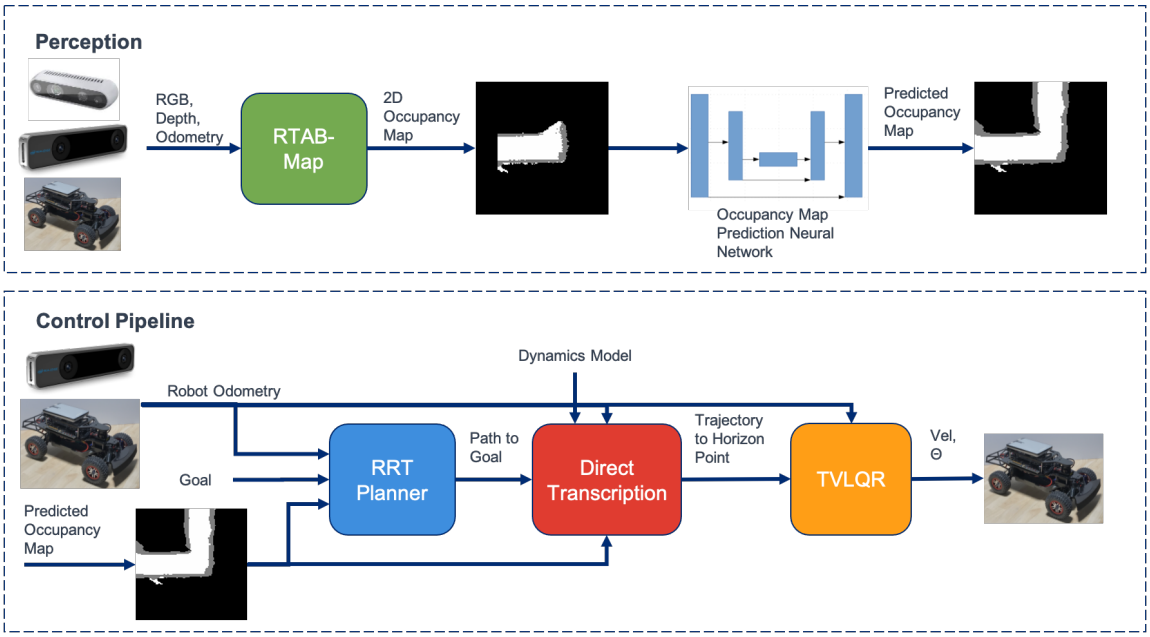

Fig. 2: This diagram describes the overall perception and control pipeline. The perception module receives on-board sensor data from the car and produces a predicted occupancy map using a U-Net style generative neural network. The control algorithm receives the robot state, predicted occupancy map and goal point and generates collision-free trajectories.

Adjacent to map prediction is the problem of planning the shortest path to a goal in an unknown environment. These methods perform some level of inference over the environment to inform motion planning. For example, in [11], they learn how to plan waypoints to a goal using a partial map of the environment. Unlike our method, this approach does not consider robot dynamics. Both [12] and [13] use prior experience to reduce the size of viable environment hypotheses. Specifically, [12] learns the probability of collision of motion primitives whereas [13] utilizes experience-based map predictions in a belief space planner. Similar to our approach, Elfhafsi *et al.* [14] incorporate map prediction with global path planning that respects the system dynamics, but all of their verification was done in simulation.

Existing work has also considered applying reinforcement learning (RL) in tandem with occupancy maps or depth information to navigate an unknown environment. In [15], the method uses RGB images and predicts the probability and variance of collision while navigating towards a goal, but is incapable of global planning. In [16], the approach models the world as a POMDP and take an end-to-end approach to navigate to a goal. Additionally the method presented in [17] uses a partial map as inputs to a deep RL policy to navigate in unknown environments. While these methods have shown success in application, the data requirements of deep RL policies are large and often impractical in real world scenarios. In addition, our method offers insight in the process by intermediately predicting a map.

A key challenge in high-speed navigation in unknown environments is balancing the speed of navigation against the information gained from its sensors. Our method alleviates this tension by learning to infer beyond the FOV of our sensor, but many other methods alter the planning and control pipelines to be reactive to changes in the environment. For instance [18], [19] focus on quickly planning trajectories in order to react to new obstacles. In [20], the method simultaneously plans two trajectories, a safe trajectory in known space and an aggressive trajectory into unknown space. For instance, [21] and [22] plan trajectories and verify safety using only point clouds, similar to [19]. As mentioned earlier, our strategy differs from these methods as they all focus on shortening the time between sensor reading and control generation, whereas our method infers beyond the sensor FOV to amplify the information available.

III. Preliminaries

A. Problem Formulation

Our objective is to enable an unmanned ground vehicle (UGV) to navigate, at a high speed to a goal position in an unknown environment using RGBD and tracking cameras for perception. Our approach makes use of RGBD mapping, map prediction, and a receding-horizon controller to achieve this objective.

For map prediction, the goal of our network architecture is to learn a function that maps an input occupancy map to an expanded occupancy map that extends beyond the FOV of the sensor. More formally, we are learning the function

$$f : M_{in} \mapsto M_{out}$$

where M_{in} represents the input occupancy map, and M_{out} represents the predicted, expanded output occupancy map. Components of the function f include an encoder $f_{enc}(M_{in}) \mapsto h \in \mathcal{H}$ which maps the input occupancy map to a hidden state and $f_{dec}(h) \mapsto (M_{out})$, which is a decoding

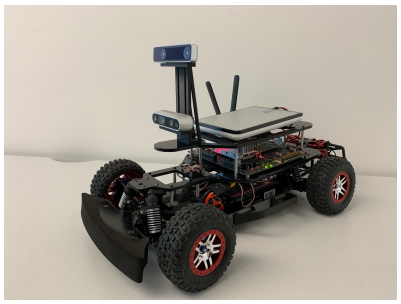

Fig. 3: MIT Racecar with Intel Realsense Cameras

function mapping the hidden state to an expanded, predicted occupancy map.

The controller accepts robot odometry state, $\mathbf{x_{robot}}$, the desired goal, $\mathbf{G_{robot}}$, a dynamics model of the robot, $\mathbf{VDM_{robot}}$, and the predicted occupancy map, M_{out} to produce the desired robot velocity, v and turn angle, θ.

B. Platform

The platform we use for our evaluation is the MIT Race car [23] built on the 1/10-scale Traxxas Rally Car platform, as shown in Fig. 3. This RC car has a reported maximum speed of 40 m/s and contains Intel Realsense D435 and T265 cameras. The onboard Nvidia® Jetson TX2 computer runs our perception, mapping and planning software integrated with the Robot Operating System (ROS) [24]. Our main interface to the RC car is the variable electronic speed controller (VESC) interface that provides vehicle state information and receives commands including desired velocity and turn angle.

IV. Approach

A. Perception

The objective of the perception algorithm is to observe co-registered RGB and depth data to produce an occupancy map for planning. As demonstrated in Fig. 2, RTAB-Map [25] is used to create 2D occupancy maps using the RGB and depth images from a Realsense D435 and visual inertial odometry from a Realsense T265.

To improve our mapping performance, we limit the D435's depth sensor range to 3 meters, and we apply gradient filtering on the raw depth images. The depth image gradients are calculated using a Sobel filter with a 5 $\times$ 5 kernel. All pixels with a gradient magnitude larger than twice the median are discarded. This removes "ghost noise" near sharp edges in the image. We use RTAB-Map to generate an updated map at approximately 3 Hz while running on the Nvidia® Jetson TX2 hardware on the car.

1) Neural Network Architecture: We use a U-Net style neural network architecture [26], [27] to receive the occupancy map provided by RTAB-Map and predict an expanded occupancy map. The U-Net neural network is a generative architecture used in several image completion algorithms including [27], [28], [29]. The U-Net network consists of skip connections allowing a direct connection between the layers i and $n-i$. These skip connections enable the option to bypass the bottleneck associated with the downsampling layers and significantly increases the accuracy of predicted occupancy regions [30]. Our implementation of the encoder network consists of 7 convolution, batch normalization and ReLU layers where each convolution consists of a 4×4 filter and stride length of 2. The number of filters for the 7 layers in the encoder network are: (64, 128, 256, 512, 512, 512, 512). Similarly, the decoder network consists of 7 upsampling layers with the following number of filters: (512, 1024, 1024, 1024, 512, 256, 128).

2) Loss Function: To train our network, we use a class-balanced cross-entropy loss function as described in [31]. The discrete classes used for labeling each pixel of the occupancy map include occupied, unoccupied and unknown spaces. Because there are significantly more pixels associated with unoccupied and unknown spaces versus obstacles, we apply class balancing techniques on the cross entropy loss with additional 5$\times$ weight added to the occupied space loss. This results in predictions where the edges representing obstacles are far more pronounced as seen in Fig. 4.

3) Post Processing: During our testing on the robotic car, we frequently observed small noise artifacts being generated by the neural network, a condition commonly found in generative neural networks [32]. While seemingly minor and transient, these artifacts caused significant instability issues during control as the trajectory planner would often abruptly change the planned path in response to these artificial obstacles or would fail to find a valid trajectory. To alleviate this, we apply a traditional morphological closing operation with a 5 $\times$ 5 kernel to suppress the noise generated by the neural network with results shown in Fig. 5. While this closing operation may eliminate pole like objects such as thin chair legs from the predicted map, overall it produced a more stable prediction. The observed map and the filtered predictive map are combined to create the planning map; any unknown space from the observed map is filled in with data from the predictive map.

4) Training Details: We generate the datasets in an unsupervised manner. As the robot navigates a new environment, the robot collects data that consists of a submap that corresponds to the current occupancy map based on the sensor's horizon as well as the expanded ground truth map after the environment has been explored. We explored various sizes of the submap and found a map corresponding to 6 m $\times$ 6 m provided by best geometric size of the submap given the characteristics of the sensor. We used a map resolution of 0.05 meters per pixel so the input occupancy map image resolution was 120 $\times$ 120 pixels. Further, we experimented with various predicted region sizes and found predicting a region of 7.5 m $\times$ 7.5 m corresponding to an image size of 150 $\times$ 150 provided the optimal accuracy and performance characteristics for the controller. Further, due to limited amounts of real world data available, we perform data augmentation techniques to apply random rotations to the occupancy map training data. This allows us to be more robust to various hallway configurations as shown in Fig. 6.

Fig. 4: Predicted occupancy map generated without class balancing weight (Left) and with class balancing weight (Right) where white represents unoccupied space, grey is occupied and black is unknown. The class balancing weight produces stronger edges for obstacles in the predicted occupancy map.

Fig. 5: (Left) Generated occupancy map with noise artifacts. (Right) Predicted occupancy map after morphological close operation (white is unoccupied space, grey is occupied and black is unknown).

B. Control Algorithm

The objective of the control algorithm as described in Fig. 2 is to compute a collision free path to the goal, generate a series of feasible trajectories to waypoints, and send control commands to the mobile robot to follow the computed trajectory. We adapt the controller proposed in our prior work, [33]. While initially developed for fixed-wing flight, we believe it is particularly well suited for high-speed navigation. The receding horizon allows for rapid replanning while using a dynamically built map, and the trajectory generation and tracking allows for a high-rate, dynamically feasible control output.

1) Dynamics Model: We use a simple bicycle acceleration model to describe the robot's dynamics. The equations of motion are as follows:

$$\begin{aligned} \dot{x} &= v * cos(\theta) \\ \dot{y} &= v * sin(\theta) \\ \dot{v} &= u_0 \\ \dot{\theta} &= v * tan(\delta)/L \\ \dot{\delta} &= u_1 \end{aligned} \tag{1}$$

The state is written as $\mathbf{x} = \begin{bmatrix} x, y, v, \theta, \delta \end{bmatrix}$ where x and y are 2D position, v is forward velocity, θ is orientation, and δ is turn angle. The input, $\mathbf{u} = \begin{bmatrix} u_0, u_1 \end{bmatrix}$, represents acceleration and turn angle velocity, respectively, and L is the wheel base length.

2) Control Strategy: Here, we review the receding horizon controller proposed in [33], which can be decomposed into three main stages.

In the first stage, a path to goal is generated using a standard rapidly-exploring random tree (RRT) [34]. The resulting path is pruned and then smoothed using G2 Continuous Cubic Bézier Spiral Path Smoothing (G2CBS) [35]. The curvature along the smoothed path, $\begin{bmatrix} x(s), y(s) \end{bmatrix}$, is calculated as:

$$\kappa(s) = \frac{(y''(s)x'(s) - x''(s)y'(s))}{(x'(s)^2 + y'(s)^2)^{\frac{3}{2}}}$$

This curvature is then mapped to velocity based upon v_{max} and v_{min}, the maximum and minimum velocity.

$$v(s) = \frac{d\mathbf{x}}{dt}(s) = v_{max} - \kappa(s) * \frac{v_{max} - v_{min}}{2}$$

The path's velocity parameterization is used to reparametrize the path by time.

$$t = \int_0^s \frac{1}{v(s)} ds$$

In our implementation, we modified the RRT to improve performance in a dynamically built map by initializing the RRT tree with the raw RRT path to goal from the previous control iteration. Before initialization, we check the path for collisions and truncate it if a collision is detected. This initialization results in faster RRT computation and increased path consistency between iterations.

In the second stage, a dynamically feasible trajectory from the current state to a horizon point is generated. The horizon point is selected as a time horizon selected along the parameterized RRT path. We utilize the same direct transcription feasibility problem as formulated in [33] and refer the reader to their formulation. This approach discretizes the trajectory into N knot points using a variable time interval dt. Let $\mathbf{x}_0(t_k)$ be the position of the robot and $\mathbf{u}_0(t_k)$, $k < N$, be the input at the k^{th} knot point where $t_{k+1} = t_k + dt$. We modify this feasibility problem by introducing a cost function to penalize large dt (therefore encouraging high speeds) as well as slightly penalizing the input to encourage smoother trajectories. The objective function is shown below.

$$J(\mathbf{x}_0(t_k), \mathbf{u}_0(t_k), dt) = \sum_{k=0}^{N-1} \mathbf{u}_0(t_k)^T \ \mathbf{R_c} \ \mathbf{u}_0(t_k) + dt$$

In order to track the dynamically feasible trajectory, time-varying LQR (TVLQR) is performed in the third stage. The control signal is generated as:

$$\mathbf{u}(t_k, \mathbf{x}) = \mathbf{K}(t_k)(\mathbf{x} - \mathbf{x}_0(t_k)) + \mathbf{u}_0(t_k).$$

where $\mathbf{K}(t_k)$ is the optimal gain matrix.

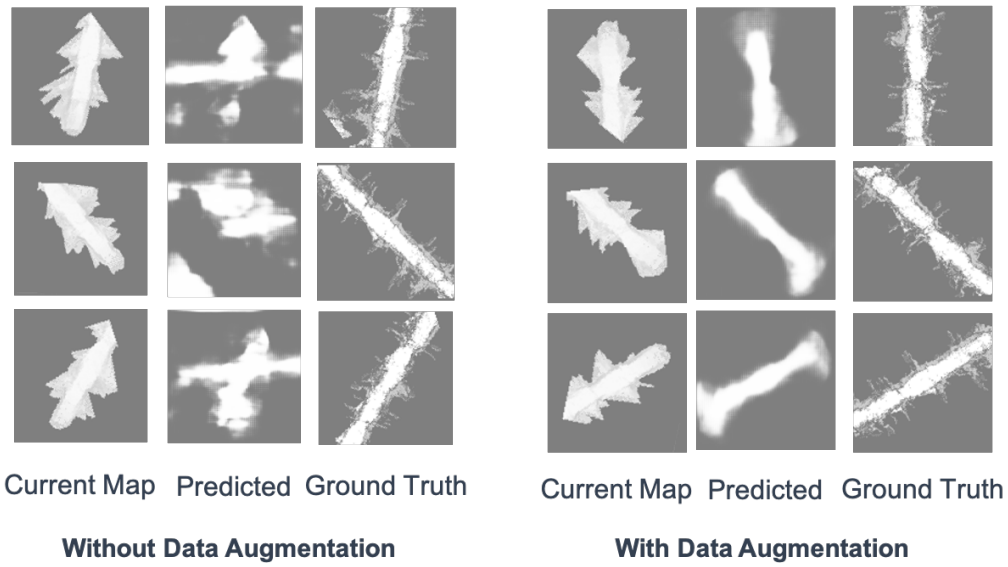

Fig. 6: Three examples from our training set of occupancy maps and their resulting expanded predictive map along with the ground truth (white is unoccupied space, light grey is occupied and dark grey unknown). Augmenting our training data with random rotations, allows the network prediction to be more robust to different environment configurations encountered by the robot.

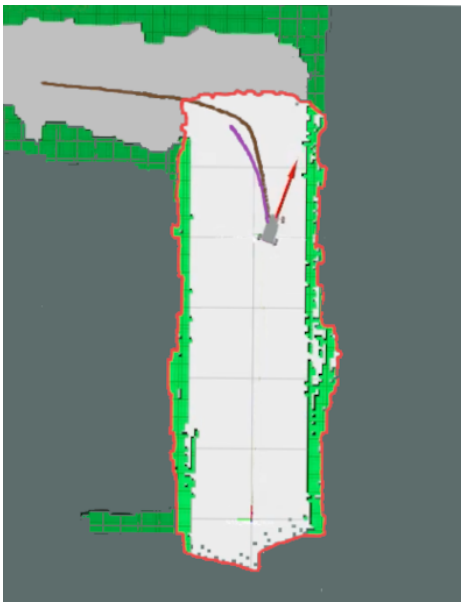

Fig. 7: Visualization of system during hardware experiment. Known map is enclosed by the red boundary. The brown path is the smoothed RRT path to goal, and the purple path is the local optimized direct transcription trajectory.

3) Control Parameters: The control pipeline executes at a rate of 5 Hz, or a control interval of $T = 0.2$ seconds. The control signal is calculated from the odometry and TVLQR gains at a rate of 50 Hz. The max time and max iterations for the RRT search were set to 0.05 seconds and 20000 iterations. The maximum velocity for RRT path parameterization was set to the maximum allowed velocity specified in each hardware trial. For RRT sampling, we use an an obstacle avoidance radius of 0.4 m.

For direct transcription, we used $N = 10$ knot points and a time horizon of $H = 2$ seconds. We set $\delta_f = \big[0.1, 0.1, 0.1, 0.25, 100.0\big]$, $\mathbf{R_c} = diag(0.1, 0.1)$ and use an obstacle radius of 0.35 m.

The costs for TVLQR are as follows:

$$Q = diag(10, 10, 10, 10, 10)$$
$$Q_f = diag(1, 1, 5, 1, 1)$$
$$R = diag(1, 1)$$

The acceleration and turn angle control bounds were set to [-2.5, 2.5] m/s and [-1.5, 1.5] rad/s respectively. The velocity minimum bound was set to 0.5 m/s and turn angle state bounds were set to [-0.3, 0.3] rad.

V. Experimental Evaluation

We conduct preliminary hardware experiments to validate our approach using a robotic car based on MIT's open-source race car [23] as described in III-B. Our map prediction network was trained on indoor scenes consisting primarily of straight and turning corridors. Similar to [36], we conduct both zero-shot and continual learning scenarios. In our zero-shot experiments, we evaluate the performance on new environments (Fig. 8) not seen by the robot. In the continual learning evaluation, we allow the robot to collect data in a semi-supervised manner from the new environment, fine tune the network offline and reevaluate performance. We assessed the maximum speed allowed by the robot and the number of successful trials, which we defined as reaching the target goal without collision. In these experiments, unobserved space is interpreted as unoccupied by the controller. A visualization

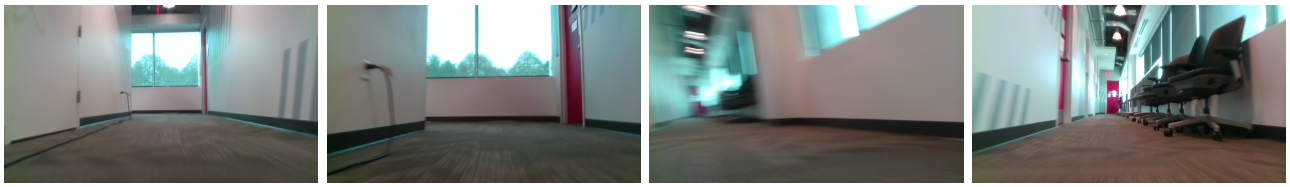

Fig. 8: This sequence of images represents a trajectory taken by the robot during our experimental evaluation.

Algorithm	Max Speed	Success Rate
Without Map Prediction	3 m/s	5/5
Without Map Prediction	4 m/s	1/5
With Map Prediction (Zero-shot)	4 m/s	3/5
With Map Prediction (Fine Tuned)	4 m/s	4/5

TABLE I: This table captures the results of our preliminary hardware experiments on our modified MIT race car.

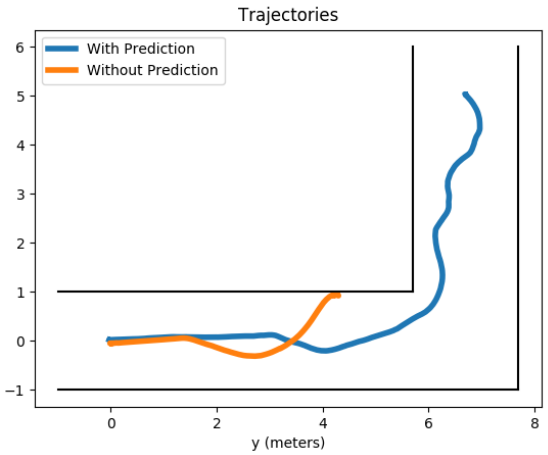

Fig. 9: Example trajectories of car with max velocity of 4 m/s with and without map prediction.

of our system during the hardware experiment is shown in Fig. 7. The results are summarized in Table I. Without map prediction, the robot's maximum speed, v_{max}, was 3 m/s without collision. When evaluating without map prediction with the maximum speed of 4 m/s, the robot was only able to successfully reach the goal 1/5 attempts. With map prediction, we were able to achieve success 3/5 trials. After allowing the network to fine tune on the new environment, we were able to increase the ratio to 4/5 successful trials showing the ability to continually learn as the robot explores new environments. For comparison, the trajectories with and without prediction for one of the trials where $v_{max} = 4\,\text{m/s}$ is shown in Fig. 9. Without map prediction, the robot, limited by the sensor's field of view, optimistically plans a waypoint in unknown space. At the highest tested velocities, it is not able to react in time once the map has been updated to reflect the true occupied space. In contrast, with map prediction, we are able to plan with longer horizons resulting in smoother trajectories and more time to react to obstacles which allows the robot to reach the desired goal.

VI. Discussion and Conclusion

In this paper, we describe an approach that enables high speed navigation based on predicted occupancy maps. Specifically, we present a generative neural network architecture approach to collect self-supervised training data and training methodology that allows the robot to predict occupied spaces beyond the line of sight of the camera. Further, we present several engineering solutions in the perception algorithm that were required for prediction to work on real data on the physical hardware including data augmentation, using a class-balanced loss function and traditional computer vision methods for noise suppression.

In addition, we also present a real-time controller that leverages the predicted occupancy map as part of the planning algorithm. At a max velocity of 3 m/s, 3 Hz mapping rate, 3 m sensor range, and 1 sec time horizon, planned trajectories will almost always be within the known region of the map. It isn't surprising, thus, that the system is successful without map prediction with these parameters. When max velocity is increased to 4 m/s, planned trajectories will often be within unknown space of the map (outside of the sensor range). This can cause trajectories to plan through unseen walls, causing failure. The predicted occupancy map is able to help address this limitation by providing a longer horizon for planning which accounts for the improved performance at 4 m/s in our preliminary hardware evaluation.

While the results are promising, there are many opportunities for future work. One area is to extend our preliminary hardware experiments to more thoroughly train and test in various indoor and outdoor environments to assess the impact of map prediction for high-speed navigation. No prediction algorithm will be perfect; another area for future exploration is to incorporate the uncertainty associated with the occupancy map's prediction into our controller and trajectory planner. Possibilities include adjusting the robot's velocity based on the the confidence of the prediction, or rewarding risk-adverse trajectories. Another area to explore is to improve continual learning techniques as the robot enters new environments. It is unrealistic to expect the training data to capture the full distribution of the environments that the robot will expect to see. We believe further research is needed to improve the data efficiency of continual learning techniques so that the robot can improve performance in real time as it explores new environments.

In spite of these limitations, we have shown the promise of predictive capabilities that improve navigation performance of mobile robots and are continually developing techniques to further extend these capabilities as described above.

REFERENCES

[1] H. Voicu and N. Schmajuk, "Exploration, navigation and cognitive mapping," *Adaptive Behavior*, vol. 8, no. 3-4, pp. 207–223, 2000. [Online]. Available: https://doi.org/10.1177/105971230000800301

[2] R. L. Buckner, "The role of the hippocampus in prediction and imagination," *Annual review of psychology*, vol. 61, pp. 27–48, 2010.

[3] P. Schwartenbeck, J. Passecker, T. Hauser, T. H. B. FitzGerald, M. Kronbichler, and K. J. Friston, "Computational mechanisms of curiosity and goal-directed exploration," *bioRxiv*, 2018. [Online]. Available: https://www.biorxiv.org/content/early/2018/09/07/411272

[4] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1746–1754.

[5] H. J. Chang, C. G. Lee, Y.-H. Lu, and Y. C. Hu, "P-slam: Simultaneous localization and mapping with environmental-structure prediction," *IEEE Transactions on Robotics*, vol. 23, no. 2, pp. 281–293, 2007.

[6] R. Shrestha, F.-P. Tian, W. Feng, P. Tan, and R. Vaughan, "Learned map prediction for enhanced mobile robot exploration," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 1197–1204.

[7] K. Katyal, K. Popek, C. Paxton, P. Burlina, and G. D. Hager, "Uncertainty-aware occupancy map prediction using generative networks for robot navigation," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May. Institute of Electrical and Electronics Engineers Inc., may 2019, pp. 5453–5459.

[8] K. D. Katyal, K. M. Popek, C. Paxton, J. L. Moore, K. C. Wolfe, P. Burlina, and G. D. Hager, "Occupancy map prediction using generative and fully convolutional networks for vehicle navigation," *CoRR*, vol. abs/1803.02007, 2018. [Online]. Available: http://arxiv.org/abs/1803.02007

[9] S. K. Ramakrishnan, Z. Al-Halah, and K. Grauman, "Occupancy anticipation for efficient exploration and navigation," 2020.

[10] D. P. Ström, F. Nenci, and C. Stachniss, "Predictive exploration considering previously mapped environments," in *2015 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2015, pp. 2761–2766.

[11] G. J. Stein, C. Bradley, and N. Roy, "Learning over subgoals for efficient navigation of structured, unknown environments," in *Conference on Robot Learning*, 2018, pp. 213–222.

[12] C. Richter, J. Ware, and N. Roy, "High-speed autonomous navigation of unknown environments using learned probabilities of collision," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 6114–6121.

[13] O. Asraf and V. Indelman, "Experience-based prediction of unknown environments for enhanced belief space planning," 10 2020.

[14] A. Elhafsi, B. Ivanovic, L. Janson, and M. Pavone, "Map-Predictive Motion Planning in Unknown Environments," in *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., may 2020, pp. 8552–8558.

[15] G. Kahn, A. Villaflor, V. Pong, P. Abbeel, and S. Levine, "Uncertainty-aware reinforcement learning for collision avoidance," *CoRR*, vol. abs/1702.01182, 2017. [Online]. Available: http://arxiv.org/abs/1702.01182

[16] P. Karkus, D. Hsu, and W. S. Lee, "Qmdp-net: Deep learning for planning under partial observability," in *Advances in Neural Information Processing Systems*, 2017, pp. 4694–4704.

[17] H. Li, Q. Zhang, and D. Zhao, "Deep reinforcement learning-based automatic exploration for navigation in unknown environment," *IEEE transactions on neural networks and learning systems*, 2019.

[18] S. Kousik, S. Vaskov, F. Bu, M. Johnson-Roberson, and R. Vasudevan, "Bridging the gap between safety and real-time performance in receding-horizon trajectory design for mobile robots," *arXiv preprint arXiv:1809.06746*, 2018.

[19] B. T. Lopez and J. P. How, "Aggressive 3-d collision avoidance for high-speed navigation." in *ICRA*, 2017, pp. 5759–5765.

[20] J. Tordesillas, B. T. Lopez, M. Everett, and J. P. How, "Faster: Fast and safe trajectory planner for flights in unknown environments," 2020.

[21] F. Gao, W. Wu, W. Gao, and S. Shen, "Flying on point clouds: Online trajectory generation and autonomous navigation for quadrotors in cluttered environments," *Journal of Field Robotics*, vol. 36, no. 4, pp. 710–733, 2019.

[22] P. Florence, J. Carter, and R. Tedrake, "Integrated Perception and Control at High Speed: Evaluating Collision Avoidance Maneuvers Without Maps." Springer, Cham, 2020, pp. 304–319. [Online]. Available: https://doi.org/10.1007/978-3-030-43089-4{_}20

[23] B. Cain, "High speed autonomous vehicles using the mit racecar platform," 2017.

[24] Stanford Artificial Intelligence Laboratory et al., "Robotic operating system." [Online]. Available: https://www.ros.org

[25] M. Labbé and F. Michaud, "Rtab-map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation," *Journal of Field Robotics*, vol. 36, no. 2, pp. 416–446, 2019.

[26] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *CoRR*, vol. abs/1505.04597, 2015. [Online]. Available: http://arxiv.org/abs/1505.04597

[27] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[28] C. Xie, S. Liu, C. Li, M. Cheng, W. Zuo, X. Liu, S. Wen, and E. Ding, "Image inpainting with learnable bidirectional attention maps," in *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 8857–8866.

[29] Z. Yan, X. Li, M. Li, W. Zuo, and S. Shan, "Shift-net: Image inpainting via deep feature rearrangement," in *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XIV*, ser. Lecture Notes in Computer Science, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds., vol. 11218. Springer, 2018, pp. 3–19.

[30] K. D. Katyal, K. M. Popek, C. Paxton, J. L. Moore, K. C. Wolfe, P. Burlina, and G. D. Hager, "Occupancy map prediction using generative and fully convolutional networks for vehicle navigation," *CoRR*, vol. abs/1803.02007, 2018. [Online]. Available: http://arxiv.org/abs/1803.02007

[31] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *CVPR*, 2019.

[32] T. Kaneko and T. Harada, "Noise robust generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[33] M. Basescu and J. Moore, "Direct nmpc for post-stall motion planning with fixed-wing uavs," *arXiv preprint arXiv:2001.11478*, 2020.

[34] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[35] K. Yang and S. Sukkarieh, "An analytical continuous-curvature path-smoothing algorithm," *IEEE Transactions on Robotics*, vol. 26, no. 3, pp. 561–568, 2010.

[36] G. Kahn, P. Abbeel, and S. Levine, "BADGR: an autonomous self-supervised learning-based navigation system," *CoRR*, vol. abs/2002.05700, 2020. [Online]. Available: https://arxiv.org/abs/2002.05700