2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Adaptive Dynamic Window Approach for Local Navigation

Matej Dobrevski[1] and Danijel Skočaj[1]

***Abstract*— Local navigation is an essential ability of any mobile robot working in a real-world environment. One of the most commonly used methods for local navigation is the Dynamic Window Approach (DWA), which heavily depends on the settings of the parameters in its cost function. Since the optimal choice of the parameters depends on the environment that may significantly vary and change at any time, the parameters should be chosen dynamically in a data-driven way. To cope with this problem, we propose a novel deep convolutional neural network, which dynamically predicts these parameters considering the sensor readings. The network is trained using a state-of-the art reinforcement learning algorithm. In this way, we combine the power of data-driven learning and the dynamic model of the robot, enabling adaptation to the current environment as well as guaranteeing collision-free movement and smooth trajectories of the mobile robot. The experimental results show that the proposed method outperforms the DWA method as well as its recent extension.**

I. INTRODUCTION

The ability to navigate unstructured environments is a crucial capability for mobile robots operating in the real world. Applications like service and inspection robots, delivery drones, unmanned surface vehicles, search and rescue robots and many more, are all limited by the capabilities of the navigation software in use. Rightfully, robot navigation has been an extensively researched topic [1].

The usual approach for designing navigation systems is to split the problem into global navigation and local navigation. Global navigation, or commonly path-planning [2], supposes that a map of the environment is available and produces a plan for reaching the goal position, while taking in consideration the geometric properties of the robot. Unfortunately, if the environment is not strictly managed, the map used for navigation quickly becomes outdated. Potentially adding the difficulty of moving obstacles, a need for local navigation quickly arises. Local navigation, in contrast to global navigation does not rely on a map of the environment, but only on the immediate sensor readings of the robot. It alleviates the problems of unforeseen changes in the map by considering the dynamic properties of the robot and making short-term decisions on avoiding obstacles while following the global plan as close as possible.

Fueled by the availability of computing power, data, and new techniques for their training, deep neural-networks have shown impressive results on a number of domains in the last decade. Consequently, many approaches that use a neural-network to navigate a robot have appeared [3], [4], [5], [6], mostly using reinforcement learning (RL) algorithms to train the network. However, in this navigation paradigm it is not easy to make targeted improvements to the robot behaviour. Additionally, they need mechanisms for ensuring collision free navigation. These shortcomings can be overcome by considering classical navigation approaches that specifically address these issues.

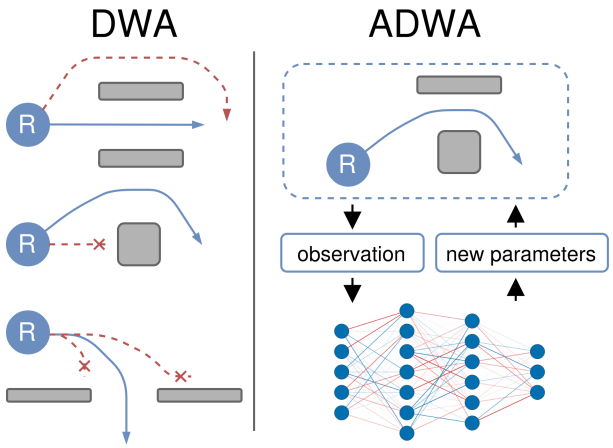

Fig. 1. On the left: some canonical navigation scenarios in which DWA can have problems. Parameters that are optimal for one situation can lead to a local minimum or a longer path in another situation. On the right: our proposal, to adapt the parameters at each step based on a pre-trained network.

One of the most commonly used classical methods for local-navigation today is the DWA [7]. In their extensive experiments for indoor navigation, Willow-Garage decided to include DWA as the local navigation method [8] and this is the default local navigation method used in the navigation stack in ROS [9]. DWA generates control velocities to achieve a goal position by using the dynamic model of the robot to optimize the distance to obstacles, its velocity, and the heading towards the goal. The reasons for its continued usage are numerous: it generates smooth trajectories, the trade-of between precision and computational power can be controlled via the sampling, it works in velocity space so there is no need for an additional controller, and it guarantees collision free trajectories. As such, it has been extended to numerous applications such as global navigation [10] and navigation in presence of known moving obstacles [11].

On the other hand, as pointed out in the original paper [7] and other research [12], the performance of DWA depends on the weights we choose for the obstacle clearance, speed and heading in the cost function. As illustrated in Figure 1 it is dependent on these weights whether a robot will pass through a narrow gap, whether it will circumvent a given

[1]Visual Cognitive Systems Laboratory, Faculty of Computer and Information Science, University of Ljubljana, Večna pot 113, Ljubljana, Slovenia
matej.dobrevski@fri.uni-lj.si

obstacle or come completely to a stop. Additionally, there is no general rule for setting these weights, but researchers and practitioners set them through trial-and-error.

This problem has been addressed in [13], [14] and [15]. In [13] and [14] the authors propose a fuzzy system for adapting these weights. However the rules for adapting these weights are still set manually, and only 8 configurations of obstacles are considered. In [15] the authors propose the usage of an Artificial Neuro Fuzzy Inference System (ANFIS) to learn to adapt these weights from experience. However, they do not consider the configuration of obstacles in the learning system.

In this work we present a new method for local navigation, which combines the adaptability of learned methods while guaranteeing collision free trajectories and generating smooth trajectories. Our method is based on the DWA, but the cost function that DWA optimizes is changed at each step by a deep neural network. We train the neural network in a reinforcement learning setting with the usage of the state of the art Proximal Policy Optimization (PPO) [16] method. Our method takes as an observation a laser-scan of the environment and the position of the next goal and output the new weights for the cost function. We evaluate our method extensively on the Gibson dataset [17] consisting of 3D scans of real indoor spaces and show that we outperform both DWA and its recent extension [15].

The paper is organised as follows. In the following section we first present the related work. This is followed by the presentation of the proposed method Adaptive Dynamic Window Approach (ADWA). Evaluation of the proposed method and comparison to related work is presented in Sec. IV. The paper closes with the concluding remarks in Sec. V.

II. RELATED WORK

Local navigation, or navigation based on the robot sensor readings, has been studied since the development of the first mobile robots. From these various approaches the most used ones have been methods operating in velocity space, as they consider the kinematic and dynamic constraints of the robot, and in contrast to geometrical approaches, can be easily translated to motor commands. Velocity space methods include several curvature based methods [18], [19], [7], [20], from which the most popular has been the DWA, and is included as the default local navigation method in ROS.

The DWA addresses the problem of navigation by searching in the collision free part of the reachable velocity space, for speeds which maximize a weighted cost function in order to maximize the clearance from obstacles, heading towards the goal, and movement speed of the robot. Brock and Khatib [10] expanded the algorithm so that it is applicable to global navigation scenarios. Their approach requires a costly calculation of a global navigation function, and is not suitable for scenarios where the environment is expected to change. Ogren et al. [21] inspired by the work done by Rimon and Koditschek [22] developed a version of DWA that is provably convergent by viewing the method as Model Predictive Control. Recently, Missura and Bannewitz [11] presented a variant of DWA which also addresses navigation scenarios with moving obstacles, which are modeled as moving polygons.

To address the problem of needing different weights for different scenarios Hong et al. [13] proposed the usage of a fuzzy system to adapt the weights of the algorithm based on the distribution of obstacles around the robot. However, the obstacle distribution is one of eight types, and the fuzzy rules are set by hand. Betoño et al. [15] expand this idea and introduce an ANFIS for predicting the weights, and introduce optimization of the cost function for two steps ahead. In the inputs to their neuro-fuzzy predictor they only consider the mean of the obstacle distances, the heading and the current speed, and not the detailed distribution of obstacles. In contrast to these approaches we consider the distribution of obstacles in 128 measurement points around the robot and we use a deep convolutional network to predict the optimal parameters for the cost function. Additionally, our network is trained on experience using the state-of-the art PPO method [16].

In the last few years numerous approaches that use a neural network to model the complete navigation policy have appeared [3], [4], [5], [6], [23], using reinforcement learning to train the network. Xie et al. [3] have developed a monocular image obstacle avoidance method based on the dueling and double-Q mechanisms. This method however does not navigate to a goal location. Zhu et al. [4] developed a method for navigating to a target given as an image in a previously known environment. Leiva et al. [24] learn a collision avoidance policy which as input combines the data from the RGB, depth image and lasers on the robot. Lei at al. [23] train a navigation policy for the Turtlebot using a Deep RL algorithm. One downside of purely using a neural network is that targeted improvements in the navigation behavior are impossible. Other downsides are the lack of guarantees for collision free trajectories, and the generated smoothness of the robot movement. In our work we combine the strength of both, learning based and classical approaches, and offer improvement over classical approaches while also guaranteeing collision free navigation and smooth trajectories.

Our method is based on deep reinforcement learning. Since the incredible results achieved by Mnih et al. [25] and their Deep Q-Networks (DQN) there has been a resurgence of research in reinforcement learning, such as methods that improve the DQN like Dueling DQN [26] and Double DQN [27]. Techniques for improving convergence by breaking the correlation between successive samples, like Experience Replay [28] and Prioritized Experience Replay [29] have also been used. Another direction has been research into methods applied to continuous outputs like Deep Deterministic Policy Gradients [30] and other methods directly optimizing the policy like Trust Region Policy Optimization (TRPO) [31] which addresses some of the problems with the stability in reinforcement learning methods, by making sure that the policy does not change drastically between

successive updates. It does this by limiting the Kullback-Leibler divergence between updates, with the size of the updates to the network parameters. PPO [16] is a kind of successor method to TRPO, however the costly Killback-Leibler divergence calculation has been replaced with a much computationally simple limitation on the maximal permitted gradient update. We use PPO because of its elegant and efficient implementation and demonstrated convergence properties.

III. Adaptive DWA

A. DWA method

In the DWA we assume that the control intervals are sufficiently small so that in each control interval $[t_i, t_{i+1}]$ we can assume the velocity of the robot is constant. In our experiments this interval is $\Delta t_i = t_{i+1} - t_i = 0.2s$. If we assume a synchro-drive robot and if the translational velocity of the robot in the $[t_i, t_{i+1}]$ interval is ν_i, its rotational velocity is ω_i and its orientation at time t_i is $\theta(t_i)$, then we can derive that during the given interval the robot will be moving along a circular arc with a center in:

$$(-\frac{\nu_i}{\omega_i} sin\theta(t_i), \frac{\nu_i}{\omega_i} cos\theta(t_i)) \tag{1}$$

and a radius of $r = \frac{\nu_i}{\omega_i}$ assuming $\omega_i \neq 0$ [7]. If $\omega_i = 0$ then the robot will be moving along a straight line in the direction $\theta(t_i)$. Knowing this, at each control interval the two-dimensional search space (ν, ω) for the command is reduced to the intersection of the admissible velocities and the dynamic window. Admissible velocities are all velocities that will not cause a collision, in other words, velocities that, if applied, the robot will be able to stop before reaching an obstacle on the current curvature, given its deceleration limitations. The dynamic window are all velocities that can be reached within the next control cycle, given the limited acceleration of the robot.

According to the DWA from this search space we choose the velocities that maximize the function:

$$G(\nu, \omega) = \sigma(\alpha \cdot heading(\nu, \omega) + \beta \cdot dist(\nu, \omega) + \gamma \cdot vel(\nu, \omega)) \tag{2}$$

where $heading(\cdot)$ is the angle between the forward direction of the robot and the goal, a measure of how much we are moving towards the goal; $dist(\cdot)$ is a measure of clearance, or the distance to the closest obstacle on the curvature; $vel(\cdot)$ is simply the translatonal velocity of the robot, so as to maximize its speed; $\sigma(\cdot)$ is a smoothing operator that normalizes $heading(\nu, \omega)$, $dist(\nu, \omega)$ and $vel(\nu, \omega)$ to the range $[0, 1]$. The behaviour of the robot will change with different values of α, β and γ. Our experiments showing this dependence can be seen in the next section. The optimal setting of these weights depends on the configuration of obstacles around the robot, and in this work we design a neural network that does exactly that.

B. Deep reinforcement learning approach

We model the problem of setting these weights as a Markov Decision Process (MDP) and solve it in a reinforcement learning setting. The state at time t is defined as a vector $s_t = [l_1, l_2, ..., l_{128}, \theta_g, d_g, \nu_r, \omega_r]$ where l_1 to l_{128} are laser range readings around the robot, measuring distance $[0, 4]m$ in $\pm 120^\circ$ about the forward direction of the robot; θ_g is the angle to the goal; d_g is the distance to the goal; ν_r is the translational speed of the robot, and ω_r is the rotational speed of the robot. After each observation of the state, the robot produces an action $a_t = [\Delta\alpha, \Delta\beta, \Delta\gamma]$. During the training we consider the discount factor to be $\gamma_{discount} = 0.99$. The reward function $R(s_t)$ does not contain any intermediate rewards and is defined as:

$$R = \begin{cases} 100 & , d_g < 0.3m \\ -30 & , \text{robot stuck} \\ 0 & , \text{otherwise} \end{cases} \tag{3}$$

The transition function of the MDP is implicitly defined by the dynamics of the robot and the environment in which it is learning.

The laser-scan data in the physical world is highly correlated. The network should learn to extract from this low-level data information like locations of sharp edges, small obstacles, and traversable areas, and combine this information with the movement parameters of the robot and the location towards it is headed and these ideas are reflected in the design of the network.

As input, the network takes a 128 laser-range measurements in $\pm 120^\circ$ around the robot, passes this information through four 1D convolutional layers and concatenates it with the relative distance and orientation to the goal, as well as the current translational and rotational velocity of the robot, as shown in Figure 2. The combined vector is processed by four more fully connected layers, before the output layer. The outputs of the network have the $tanh$ activation function so they are normalized in the range $[-1, 1]$. The laser-scan, as well as the four additional parameters to the network, are normalized before they are fed in the network. In PPO, during training we use an additional network for estimating the value of the states, which is in turn used for calculating the advantage function. As is usual, the value estimation network which is used for calculating the advantage function during learning, shares the same structure but has a single output with a linear activation function.

The network produces $\Delta\alpha$, $\Delta\beta$ and $\Delta\gamma$ as outputs which are scaled to the ranges $[-0.5, 0.5]$, $[-0.3, 0.3]$ and $[-0.3, 0.3]$ respectively, then summed with the default values of $\alpha = 0.8$, $\beta = 0.1$ and $\gamma = 0.1$, and the underlying DWA operates with these adapted parameters.

C. Training

The network is trained in simulation in a reinforcement learning setting using an artificially constructed obstacle course depicted in Figure 3. The obstacle course is significantly different to most maps of real world environments

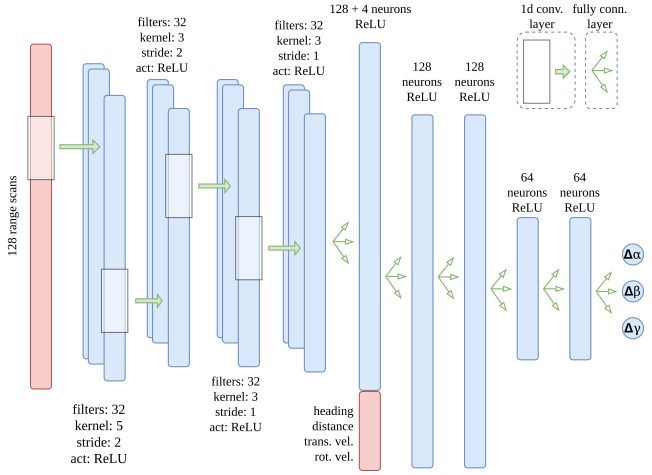

Fig. 2. The neural network transforms the range-scan of the robot through four convolutional layers. The output from the last convolutional layer is fed into a fully connected layer, after which it is concatenated with the information about the goal and the velocity of the robot. This complete information is then fed through four fully connected layers, before finally outputting the new parameters.

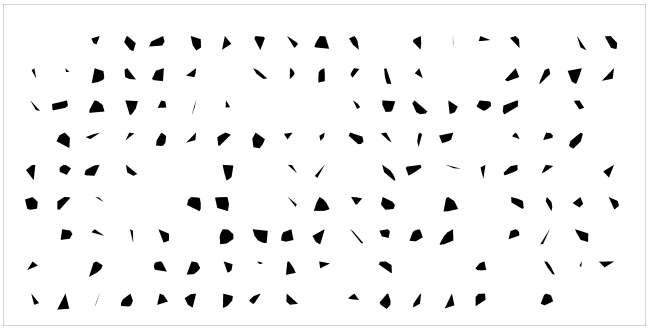

Fig. 3. The polygon used for training the network is a $20m$ by $40m$ hall filled with randomly generated convex obstacles.

(some of which are shown in Figure 4). We created this obstacle space entirely of randomly generated obstacles in order to prevent over-fitting of the trained network to existing features of certain environments, and force it to learn the underlying physics of the problem. Randomization during training has been found to be an effective technique for alleviating issues with over-fitting in reinforcement learning problems [32], [33], [34].

The simulation is implemented in ROS, with nodes simulating the robot and range-scanner. For our robot, the Turtlebot, we performed parameter identification and adapted the transition functions for the translational and rotational velocities in the simulator, to match them as close as possible to the real robot.

Each episode of interaction begins by randomly initializing the starting and target location of the robot within a predefined distance $\delta = 4m$. In each control cycle ($5Hz$) the neural network outputs $\Delta\alpha$,$\Delta\beta$ and $\Delta\gamma$ values which are added to the default values suggested by the authors of the original DWA. The system then navigates the robot to the target location until an episode is terminated, which can happen if: the robot reaches the target location (distance to the goal $d_g < 0.3m$), the robot exceeds 300 control cycles, or the robot gets stuck in a minimum from which it can not get out (has not moved for 20 control cycles).

These experiences are gathered into a buffer and when the buffer reaches 5000 steps an update to the policy is performed according to the PPO rules. This represents one epoch of training. We train the policy for 600 epochs, or a total of $3M$ steps. The training lasts for a couple of hours, when the simulation is using 4 cores of the Intel i7-6700 CPU and the policy is trained on a Nvidia GTX 1070 GPU. The trained network is then directly applied to the new domain, without any further training.

IV. EVALUATION

A. Experimental setup

Evaluating the effects of a local navigation on a realistic navigation scenario requires some consideration. For example, the ROS implementation of DWA[1] contains several techniques on top of the bare-bones algorithm for significantly improving its performance: the local goal is updated at each control-cycle; additional cost is added for adhering to the detailed path of the global planner; a flag for limiting oscillatory behaviour; additional scoring point away from the center of the robot; recovery behaviours etc. To produce comparable results, we do not implement these additional techniques, but use the method as it was originally presented and is implemented in previous research [7], [15], [11].

To isolate the effects of the local-planner on the success of navigation we thus simplify the interaction between the local and the global planner as much as possible. The global planner is run only to create navigation scenarios and does not interact with the evaluated navigation methods in any other way. This also results in some goals being placed in places where pure local-navigation can not reach the goal effectively. In production, a navigation system would include a more complex interaction between the global and the local planner, such as taking care where the targets for the local planner were placed given the velocity, position and orientation of the robot, would include recovery behaviours and other interactions. However, in this evaluation we focus only on the local planer, since the improvement in the performance of this component naturally translates in the improvement of the overall performance as well.

For evaluation we used a subset of the Gibson dataset [17], which contains 3D data of scanned indoor spaces, to generate realistic 2D floor-plans (six of which are depicted in Figure 4) of indoor office and home environments[2]. If the space contained multiple floors we generated a separate 2D map for each floor. In each map we randomly sampled 10 starting and goal locations, while taking care that there is at least one obstacle between the start and the goal. In this way we generated a total of 170 non-trivial navigation episodes.

[1] https://github.com/ros-planning/navigation/tree/melodic-devel/dwa_local_planner

[2] http://www.vicos.si/Downloads/IN2D

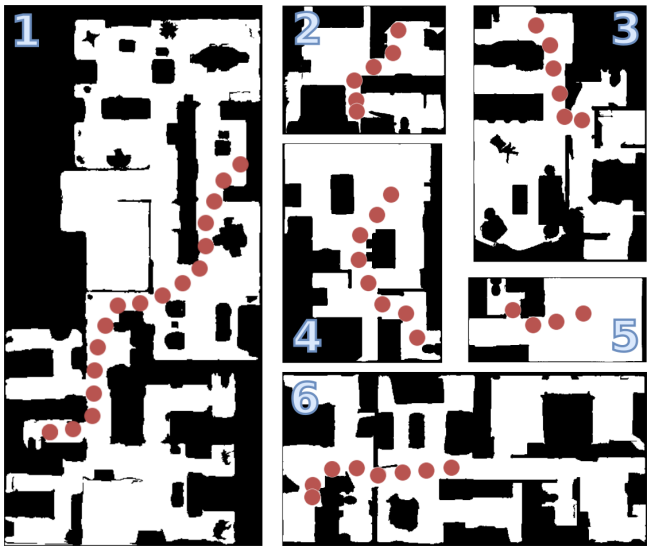

Fig. 4. Visualizations of 6 of the total 17 environments used for evaluation of the navigation methods. The environments are real floor plans of 3D scanned indoor spaces and contain realistic configurations of various obstacles. One randomly chosen navigation episode is visualized for each environment. These maps were loaded in ROS and a simulation of the Turtlebot robot was navigated on each map.

The $Theta*$ [35] global planner was executed on each pair of starting and goal location for every map and generated the shortest trajectories. On this trajectory we chose intermediate goals for navigation such that consecutive goals are $1m$ apart (Euclidean distance), except the final goal which is within $(0, 1]m$ away from the previous goal. The robot was initialized in the starting location and navigated to each intermediate goal until it reached the final goal, after which an episode was ruled successful. If it failed to reach an intermediate goal the complete episode was ruled unsuccessful. Each goal was considered reached when the center of the robot was within $0.3m$ of its location.

B. Sensitivity of DWA

TABLE I
THE BEST PARAMETERS FOR DWA

α	β	γ	# completed ep.	# reached goals
0.8	0.1	0.1	66	511
1.0	0.1	0.3	60	488
1.0	0.1	0.5	52	403
1.0	0.1	0.1	50	435
0.8	0.5	0.1	50	392

To examine the sensitivity of DWA to the weights in the cost function we performed a grid search of the α, β and γ weights, evaluating the performance of the algorithm with the different weights on the complete dataset for each generated combination. The five best results achieved in this search are shown in Table I. From the results we can confirm that the DWA is sensitive to the weights in the cost function. More precisely, the results show that in order to get acceptable behaviours α should be about $4 - 10$ times larger than β and γ. In the instances where this was not the case, the algorithm failed almost completely. These results show that the performance clearly depends on these parameters.

TABLE II
EVALUATION OF APPROACHES ON THE GIBSON DATASET, SEQUENTIAL NAVIGATION

method	# completed ep.	# reached goals
Best DWA[7]	66	511
ANFIS DWA[15]	39	337
Ours	**89**	**624**

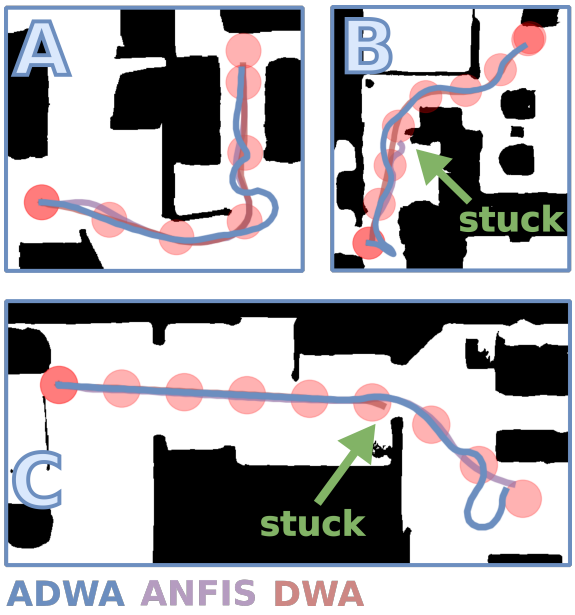

Fig. 5. In the navigation episode in A all three methods reach the goal. In B the ANFIS DWA method is stuck near a sharp corner, while in C the DWA gets stuck in front of a protruding obstacle (a wall).

C. Experimental results

Using the same obstacle course as depicted in Figure 3, we generated 10000 episodes of experience and trained an ANFIS controller for DWA as specified in [15]. This navigation method was then run on the same navigation episodes as the DWA. As can be seen in Table II, it did not manage to outperform DWA with a good combination of parameters. We evaluated our trained network on the same pre-generated episodes.

The results of all three methods are presented in Table II. As can be seen, our approach can navigate about 35% more episodes than the vanilla DWA with the best combination of parameters that we managed to find. The ANFIS method on the other hand scored worse than the DWA with the best parameters, completing about 40% less episodes. A visualization of the generated trajectories on three navigation episodes is shown in Figure 5.

We also note that for all navigation approaches most episodes contain an intermediate goal which the approaches did not manage to reach. Upon closer inspection we noticed

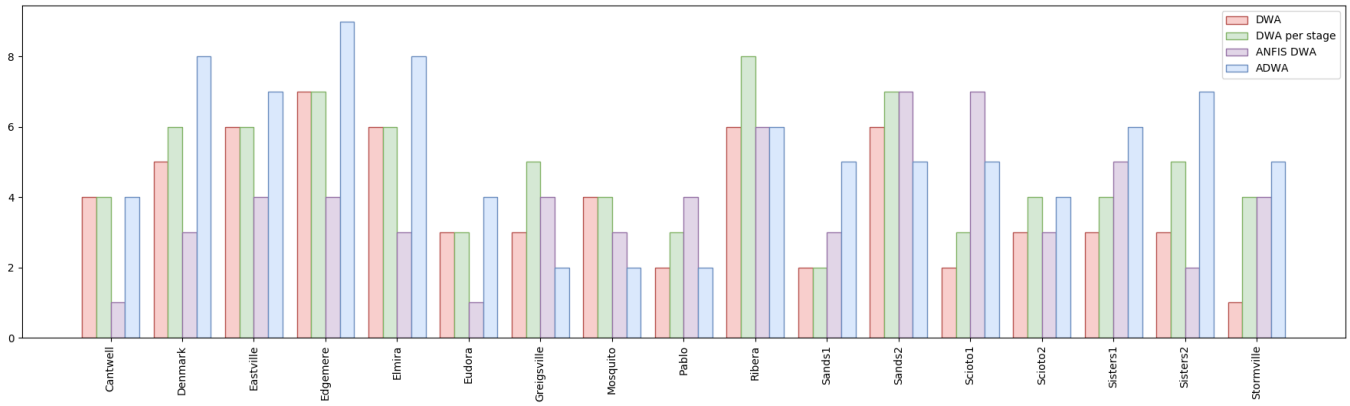

Fig. 6. Completely finished episodes for each environment in the dataset.

that these are mostly situations where the next goal is both around a corner and close to an obstacle (wall).

We examined the results per different environments in the dataset, the results of which are presented in Figure 6. Our method outperforms the DWA approach even if we compare our method to the best performing parameters tuned for each environment separately, which confirms our hypothesis that we are able to effectively predict suitable parameters for the cost-function on the basis of the distribution of obstacles around the robot.

To examine the behaviour of the navigation methods on longer navigation tasks, we generated new navigation episodes. In this test, each episode contains a starting location and just a single goal such that the distance between the start and the goal is between 0 and 4 meters. The start and goal locations are sampled containing at least one obstacle in between, so that there are no trivial navigation tasks. We generated 100 tasks per environment or 1700 in total. On this dataset we evaluated the best DWA, ANFIS DWA and our method. The results are presented in Table III.

TABLE III
EVALUATION OF APPROACHES ON THE GIBSON DATASET, SINGLE DISTANT GOAL.

method	# completed ep.
Best DWA[7]	294
ANFIS DWA[15]	340
Ours	**520**

From the presented results we can see that the performance of DWA approach drops as the distance of the goal location is increased, being able to complete only 17% of the tasks. On these longer navigation tasks the ANFIS DWA outperforms the vanilla DWA, finishing about 16% more navigation tasks. Our method outperforms both methods again, reaching about 76% more episodes than DWA. The experiments also show that our method needs about 3.3% additional computation time compared to the vanilla DWA.

V. CONCLUSIONS

The main contribution of this paper is the proposed new local navigation method which uses the dynamic model of the robot and a deep neural network to reach a given goal. In this way we combine the power of data-driven learning, while also guaranteeing collision free movement and smooth trajectories.

Our method is based on the established DWA algorithm, and addresses the sensitivity of performance to the setting of the weights in the cost function. We do this by designing a deep convolutional neural network which predicts this parameters on the basis of the sensor readings of the robot, and we train this network using the state-of-the art PPO reinforcement learning algorithm. The network is trained in a reinforcement learning setting in order to optimize the long term goal of reaching the target location. Our usage of the DWA as the backbone, guarantees that the generated robot trajectories are smooth and collision free.

We also create a benchmark dataset, generated from the 3D scanned Gibson dataset, on which we evaluate our navigation method versus the vanilla DWA and previous work on adapting these parameters. Furthermore, by evaluating against the best setting of parameters for each environment in the dataset, we conclusively show that our method outperforms simple adaptations of the cost function parameters.

Continuing this work, we will address the problems of DWA that current approach did not manage to solve, namely the problems of navigating around corners to goals that are close to the wall. The solution of this problem will enable our algorithm to be used in longer horizon navigation tasks, making it applicable to a lot of navigation scenarios without the usage of a global planner.

ACKNOWLEDGEMENTS

This work was in part supported by the ARRS research project DIVID (J2-9433) and research programme Computer vision (P2-0214).

REFERENCES

[1] J.-A. Meyer and D. Filliat, "Map-based navigation in mobile robots:: Ii. a review of map-learning and path-planning strategies," *Cognitive Systems Research*, vol. 4, no. 4, pp. 283 – 317, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S138904170300007X

[2] A. S. H. H. V. Injarapu and S. K. Gawre, "A survey of autonomous mobile robot path planning approaches," in *2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE)*, Oct 2017, pp. 624–628.

[3] L. Xie, S. Wang, A. Markham, and N. Trigoni, "Towards monocular vision based obstacle avoidance through deep reinforcement learning," in *RSS 2017 workshop on New Frontiers for Deep Learning in Robotics*, 2017.

[4] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 3357–3364.

[5] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, D. Kumaran, and R. Hadsell, "Learning to navigate in complex environments," *CoRR*, vol. abs/1611.03673, 2016. [Online]. Available: http://arxiv.org/abs/1611.03673

[6] H. L. Chiang, A. Faust, M. Fiser, and A. Francis, "Learning navigation behaviors end-to-end with autorl," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2007–2014, April 2019.

[7] 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, March 1997.

[8] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, and K. Konolige, "The office marathon: Robust navigation in an indoor office environment," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 300–307.

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

[10] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 1, May 1999, pp. 341–346 vol.1.

[11] M. Missura and M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 8620–8626.

[12] A. Maroti, D. Szaloki, D. Kiss, and G. Tevesz, "Investigation of dynamic window based navigation algorithms on a real robot," in *2013 IEEE 11th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, Jan 2013, pp. 95–100.

[13] Z. Hong, S. Chun-Long, Z. Zi-Jun, A. Wei, Z. De-Qiang, and W. Jing-Jing, "A modified dynamic window approach to obstacle avoidance combined with fuzzy logic," in *2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, Aug 2015, pp. 523–526.

[14] O. A. Abubakr, M. A. K. Jaradat, and M. A. Hafez, "A reduced cascaded fuzzy logic controller for dynamic window weights optimization," in *2018 11th International Symposium on Mechatronics and its Applications (ISMA)*, March 2018, pp. 1–4.

[15] D. Teso-Fz-Betoño, E. Zulueta, U. Fernandez-Gamiz, A. Saenz-Aguirre, and R. Martinez, "Predictive dynamic window approach development with artificial neural fuzzy inference improvement," *Electronics*, vol. 8, no. 9, 2019. [Online]. Available: https://www.mdpi.com/2079-9292/8/9/935

[16] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *ArXiv*, vol. abs/1707.06347, 2017.

[17] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: real-world perception for embodied agents," in *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[18] R. Simmons, "The curvature-velocity method for local obstacle avoidance," in *In Proc. of the IEEE International Conference on Robotics and Automation*, 1996, pp. 3375–3382.

[19] Nak Yong Ko and R. G. Simmons, "The lane-curvature method for local obstacle avoidance," in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*, vol. 3, Oct 1998, pp. 1615–1621 vol.3.

[20] B. P. Gerkey and K. Konolige, "Planning and control in unstructured terrain," in *In Workshop on Path Planning on Costmaps, Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, 2008.

[21] P. Ogren and N. E. Leonard, "A convergent dynamic window approach to obstacle avoidance," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 188–195, April 2005.

[22] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Transactions on Robotics and Automation*, vol. 8, no. 5, pp. 501–518, Oct 1992.

[23] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," *CoRR*, vol. abs/1703.00420, 2017. [Online]. Available: http://arxiv.org/abs/1703.00420

[24] F. Leiva, K. Lobos-Tsunekawa, and J. Ruiz-del Solar, "Collision avoidance for indoor service robots through multimodal deep reinforcement learning," in *RoboCup 2019: Robot World Cup XXIII*, S. Chalup, T. Niemueller, J. Suthakorn, and M.-A. Williams, Eds. Cham: Springer International Publishing, 2019, pp. 140–153.

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A. a. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14236

[26] Z. Wang, T. Schaul, M. Hessel, H. V. Hasselt, M. Lanctot, N. D. Freitas, and G. Deepmind, "Dueling Network Architectures for Deep Reinforcement Learning," *Journal of Machine Learning Research*, vol. 48, no. 9, 2016.

[27] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-learning," *AAAI Publications, Thirtieth AAAI Conference on Artificial Intelligence*, 2016. [Online]. Available: http://arxiv.org/abs/1509.06461.pdf

[28] L. ji Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," in *Machine Learning*, 1992, pp. 293–321.

[29] T. Schaul, J. Quan, I. Antonoglou, D. Silver, and G. Deepmind, "Prioritized experience replay," *International Conference on Learning Representations*, 2016.

[30] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," 2015.

[31] J. Schulman, S. Levine, M. Jordan, and P. Abbeel, "Trust Region Policy Optimization," *Icml-2015*, p. 16, 2015. [Online]. Available: http://arxiv.org/abs/1502.0547

[32] OpenAI, "Learning dexterous in-hand manipulation," *CoRR*, vol. abs/1808.00177, 2018. [Online]. Available: http://arxiv.org/abs/1808.00177

[33] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 23–30.

[34] F. Sadeghi and S. Levine, "$(CAD)^2$RL: Real Single-Image Flight without a Single Real Image," *arXiv:1611.04201*, 2016. [Online]. Available: http://arxiv.org/abs/1611.04201

[35] K. Daniel, A. Nash, S. Koenig, and A. Felner, "Theta*: Any-angle path planning on grids," *Journal of Artificial Intelligence Research*, vol. 39, p. 533–579, Oct 2010. [Online]. Available: http://dx.doi.org/10.1613/jair.2994