Towards Monocular Vision based Obstacle Avoidance through Deep Reinforcement Learning

Linhai Xie, Sen Wang, Andrew Markham and Niki Trigoni
Department of Computer Science, University of Oxford, Oxford OX1 3QD, United Kingdom
{firstname.lastname} @cs.ox.ac.uk

***Abstract*—Obstacle avoidance is a fundamental requirement for autonomous robots which operate in, and interact with, the real world. When perception is limited to monocular vision avoiding collision becomes significantly more challenging due to the lack of 3D information. Conventional path planners for obstacle avoidance require tuning a number of parameters and do not have the ability to directly benefit from large datasets and continuous use. In this paper, a dueling architecture based deep double-Q network (D3QN) is proposed for obstacle avoidance, using only monocular RGB vision. Based on the dueling and double-Q mechanisms, D3QN can efficiently learn how to avoid obstacles in a simulator even with very noisy depth information predicted from RGB image. Extensive experiments show that D3QN enables twofold acceleration on learning compared with a normal deep Q network and the models trained solely in virtual environments can be directly transferred to real robots, generalizing well to various new environments with previously unseen dynamic objects.**

I. Introduction

When mobile robots operate in the real world, subject to ever varying conditions, one of the fundamental capabilities they need is to be able to avoid obstacles. A long established problem in robotics, obstacle avoidance is typically tackled by approaches based on ranging sensors [4], e.g. laser scanner and sonar. However, ranging sensors only capture limited information and some of them are expensive or are too heavy/power consuming for a particular platform e.g. a UAV. Monocular cameras on the other hand, provide rich information about the robot's operating environments, are low-cost, light-weight and applicable for a wide range of platforms. However, when perception of range is obtained by monocular vision, i.e., RGB imagery, the obstacle avoidance problem becomes surprisingly difficult. This is because the 3-D world is flattened into a 2-D image plane, eliminating direct correspondence between pixels and distances.

A standard framework to solve this problem consists of two steps, the first of which utilizes visual information to infer traversable spaces and obstacles, and then secondly applying conventional path planning strategies. Recovering visual geometry is a common approach to detecting obstacles, e.g. through optical flow [18, 12], detection of vanishing points [1] and even visual SLAM [16]. Segmenting traversable areas, such as floors, based on visual appearance [22] is also a popular method. Once the surroundings are understood, various conventional path planners can then be employed to drive robots along traversable routes [5]. Although the described methods are able to decouple planning from visual information and hence benefit from conventional path planners, they usually require a large number of parameters which need to be manually tuned to plan feasible paths. It is also challenging for them to automatically adapt to the new operating areas.

Deep learning nowadays has shown its great performance in robotics and computer vision[24, 2, 3]. And supervised deep learning based path planning which learns how to avoid collision is becoming increasingly popular. In particular, with the recent advances of deep learning, several end-to-end supervised learning approaches are proposed to directly predict control policies from raw images [9, 7, 20] without following the previous two-step framework. Therefore, they can avoid complex modeling and parameter tuning of conventional path planners. Convolutional Neural Networks (CNNs), for example, are trained to enable flying robots to navigate in complex forest environments in [7]. However, due to their supervised nature, these approaches need manual labeling which is time-consuming and labor-intensive to obtain.

Self-supervised learning can be used to automatically generate labels for path planners with the aid of additional feedback. For instance, in [26] an algorithm is designed to label trajectory classes for a CNN based model by using 3D cloud points. Gandhi et al. [6] proposes to train a drone controller by predicting when it crashes. Although self-supervised learning is a feasible approach to benefiting from large dataset without human supervision, the learnt policy is essentially bounded by the label generating strategy.

Reinforcement learning explores policies though trials, and has been applied to vision based obstacle avoidance in [13]. However, the raw image is encoded as several levels of depth to predict a suitable control strategy. Deep reinforcement learning (DRL) has recently been shown to achieve superhuman performance on games by fully exploring raw images [15]. Since DLR usually utilizes a much weaker learning signal compared with supervised learning, it requires a much larger training dataset. This makes it difficult to directly use DLR for robotic applications in reality. Therefore, simulations which have a failure-and-recovery mechanism are usually used for training rather than real world exploration [14]. The trained networks can then be transferred to real robots. Although this has been successful by using laser scanner [21] and depth images [19], it is significantly more difficult for vision based models [6]. Recently Sadeghi and Levine [17] propose to

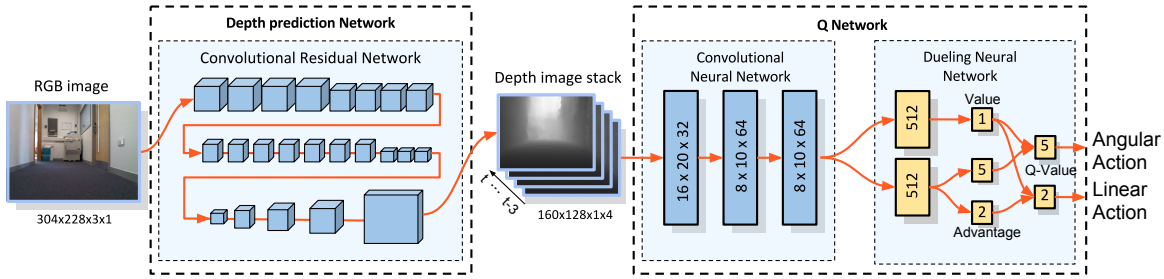


Fig. 1: Network architecture of monocular image based obstacle avoidance through deep reinforcement learning. A fully convolutional neural network is firstly constructed to predict depth from a raw RGB image. It is then followed by a deep Q network which consists of a convolutional network and a dueling network to predict the Q-value of angular actions and linear actions in parallel.

train a network as a collision predictor entirely in a 3D CAD model simulator and highly randomize the rendering settings, approximately regarding the real world as a subset of training data. Although their model can be extended into real world, it requires substantial computational resources to generate the huge dataset and train it.

In this paper, we focus on the problem of obstacle avoidance with monocular camera. More specifically, our contributions are:

- A two-phase deep neural network is proposed to achieve monocular vision based obstacle avoidance.
- Dueling architecture based deep double Q network (D3QN) is applied to obtain a high speed for end-to-end learning with limited computational resources for obstacle avoidance task.
- The knowledge learnt from simulation can be seamlessly transfered to new scenarios in the real world.
- Extensive real-world experiments are conducted to show the high performance of our network.

The rest of this paper is organized as follows. The proposed D3QN model is described in Section II. Experimental results are given in Section III, followed by conclusions drawn in Section IV.

II. Deep Q Network for Monocular Vision based Obstacle Avoidance

Since deep Q network (DQN) has been shown to be trainable directly benefit from raw images [15], most DQN models used for obstacle avoidance are based on this version [19, 27]. Although this architecture can eventually achieve reasonable results, it tends to overestimate Q values and takes a long time to train as discussed in [23, 25]. This leads to intensive computational resources for training in simulators. In this section, an advanced architecture, D3QN, is introduced to boost both performance and training efficiency for monocular vision based obstacle avoidance.

A. Problem Definition

The monocular vision based obstacle avoidance problem can be considered as a decision making process where the robot is interacting with environments with a monocular camera. The robot chooses an action $a_t \in \mathcal{A}$ according to the camera image x_t at time $t \in [0,T]$, observes a reward signal r_t produced by an assessor (reward function) and then transits to the next observation x_{t+1}. The aim of the algorithm is to maximize the accumulative future reward $R_t = \sum_{\tau=t}^{T} \gamma^{\tau-t} r_\tau$, where γ is the discount factor.

Given the policy $a_t = \pi(x_t)$, the action-value (Q-value) of a state-action pair (x_t, a_t) can be defined as follows

$$Q^\pi(x_t,a_t) = \mathbb{E}[R_t|x_t,a_t,\pi], \tag{1}$$

The Q-value function can be computed using the Bellman equation

$$Q^\pi(x_t,a_t) = \mathbb{E}[r_t + \gamma \mathbb{E}[Q^\pi(x_{t+1},a_{t+1})|x_t,a_t,\pi].$$

By choosing the optimal action each time where $Q^*(x_t,a_t) = max_\pi \mathbb{E}[R_t|x_t,a_t,\pi]$, we can have the optimal Q-value function

$$Q^*(x_t,a_t) = \mathbb{E}_{x_{t+1}}[r + \gamma \max_{a_{t+1}} Q^*(x_{t+1},a_{t+1})|x_t,a_t], \tag{2}$$

which indicates that the optimal Q-value we can obtain at time t is the current reward r_t plus the discounted optimal Q-value available at time $t+1$. Rather than computing the Q-value function directly over a large state space, the problem can be solved by approximating this optimal Q-value function with deep neural networks, which is the main principle behind DQN.

B. Dueling Network and Double Q-Network

With the intuition that it is unnecessary for all actions to be estimated at each state s, Wang et al. [25] propose the dueling network architecture. In traditional DQN only one stream of fully connected layers is constructed after the convolution layers to estimate the Q-value of each action-state pair, given the current state. However, in the dueling network, two streams of fully connected layers are built to compute the value and advantage functions separately, which are finally combined together for computing Q-values. This two-stream dueling network structure is shown in the last section of Fig.1. It has demonstrated a large improvement either on performance or training speed in a number of ATARI games (but not all).

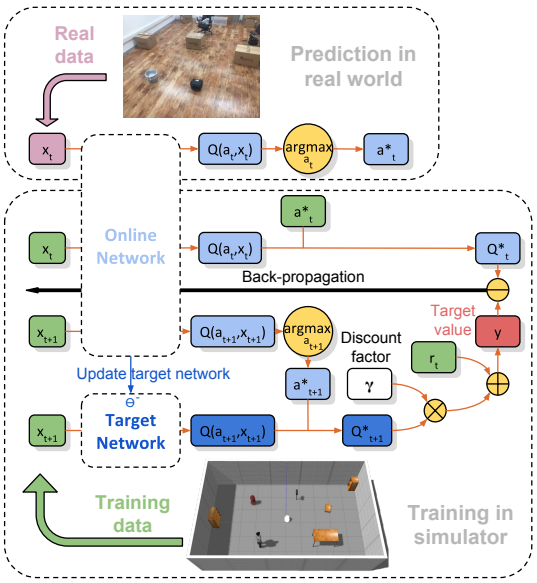


Fig. 2: When given a batch of training data, including current state x_t, action a, reward r, and resulting state x_{t+1}, the training procedure of D3QN is shown in the figure. $\oplus$, $\ominus$ and $\otimes$ are element-wise operation for addition, subtraction and multiplication.

Thus, it is exploited in our model to facilitate the learning of obstacle avoidance.

The prototype DQN in [15] utilizes a target network alongside an online network to stabilize the overall network performance. The target network is a duplicate of the online one. However, unlike the online network which updates weights by back-propagation at every training step, the weights of the target network are fixed over a short period and then copied from online network. Based on this two-network framework, Van Hasselt et al. [23] claim that the online network should be used to select actions while the target network should be used solving the problem of overoptimistic value estimation [8]. This procedure is shown in Fig.2. More specifically, the resulting state x_{t+1} is employed by both the online and target network to compute the optimal value Q'^* for time $t+1$. Then, with the discount factor γ and current reward r_t, the target value y at t is obtained. Finally, the error is calculated by subtracting the target value with the optimal value Q^* predicted by the online network, given current state x, and is then back-propagated to update the weights.

With these two techniques, the proposed D3QN is expected to be more data efficient to speed up learning. We will discuss the details of the model architecture in Section II-D.

C. From Appearance to Geometry

Since DRL needs huge amounts of data and time to train, its performance is usually demonstrated in simulated environments. In order to apply it in practice for robotic applications, a feasible solution is to train the models in simulator and then transfer them to real robots. However, this

(a) RGB (b) Predicted Depth (c) Kinect's Depth

Fig. 3: Images of RGB, predicted depth and Kinect's depth. Note the noisy depth predicted from the network.

TABLE I: Parameters of D3QN Model for Obstacle Avoidance

Name of layer	Size of filters or number of neurons	Stride
Conv 1	(10, 14, 32)	8
Conv 2	(4, 4, 64)	2
Conv 3	(3, 3, 64)	1
FC 1 for advantage	512	-
FC 1 for value	512	-
FC 2 for advantage of angular actions	5	-
FC 2 for advantage of linear actions	2	-
FC 2 for value	1	-

is highly challenging for vision based techniques due to the significant differences between virtual and real environments due to appearance, illumination, etc. To solve this problem, we propose to derive a geometric representation from the RGB imagery.

As shown in Fig.1, the first part of the D3QN model is inspired by a fully convolutional residual network (FCRN) in [11], predicting depth information from a single RGB image. However, as the depth image used is estimated by a deep neural network rather than obtained from a 3D sensor, e.g., Kinect, they are very inaccurate in practice, see Fig.3. This makes it impossible to directly use traditional ranging sensor based methods for obstacle avoidance.

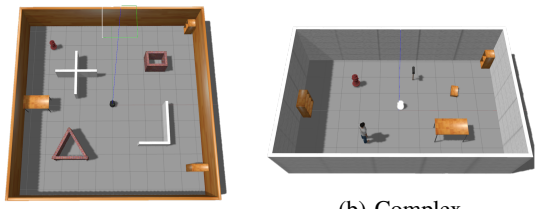
In order to tackle this serious challenge, the depth images used for training in the simulator are corrupted with random noise and image blur. We found this is critical to ensure the trained models are transferable from simulation to reality, and generalize well in real world.

D. Model and Training Settings

The D3QN model is built based on the dueling and double techniques. Its architecture is shown in Fig.1 and corresponding parameters are given in Table I. Specifically, it has three convolutional layers, specified with filter size (height, width, channel), and three fully connected layers for two streams of dueling architecture discussed in II-B.

To train the network to produce feasible control policy, robot actions need to be properly defined. Instead of the simple commands e.g. "go ahead", "turn left or right", the actions in our network are defined to control the linear and angular velocities separately in a discretised format.

The instantaneous reward function is defined as $r = v * cos(\omega) * \delta t$ where v and ω are local linear and angular velocity respectively and δt is the time for each training loop which is set to 0.2 second. The reward function is designed to let the



(a) Simple (b) Complex

Fig. 4: Two simulation worlds in Gazebo used for training.

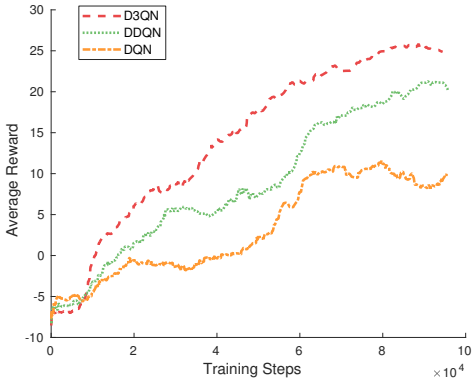


Fig. 5: Smoothed learning curves of the three models with average rewards acquired by robot.

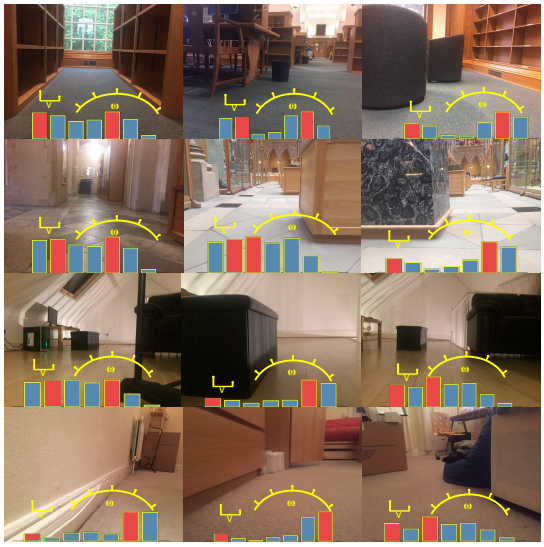


Fig. 6: Experiments in different indoor environments, e.g. library, museum, attic and bedroom (from top to bottom). The underlying bars demonstrates the Q value for each linear and angular action predicted by network, where the red ones indicate the actions greedily selected by network. Notice that the first two are for linear speed actions while the rest are for steering actions.

robot run as fast as possible and be penalized by simply rotating on the spot. The total episode reward is the accumulation of instantaneous rewards of all steps within an episode. If a collision is detected, the episode terminates immediately with an additional punishment of -10. Otherwise, the episode lasts until it reaches the maximum number of steps (500 steps in our experiments) and terminates with no punishment.

III. Experimental Results

In this section, the proposed D3QN model is evaluated in different environments. Two simulation environments, simple and complex ones, are built in Gazebo simulator for training, see Fig.4. The D3QN model is firstly trained in the simple environment before being further trained in the complex scenario. The trained model in the simulator is directly tested in several different real world scenarios. The linear velocity is set to be 0.4 or 0.2 m/s, while the angular velocity is $\frac{\pi}{6}$, $\frac{\pi}{12}$, 0, $-\frac{\pi}{12}$ or $-\frac{\pi}{6}$ rad/s, producing ten different behaviors. Throughout our experiments, a NVIDIA TitanX GPU is used for training while a laptop equipped with a NVIDIA GTX 860 GPU is used for real-time inference and testing in reality. The learning rate is set to 10^{-5} in an Adam optimizer [10]. A Turtlebot robot is used to test the control strategy in real-time.

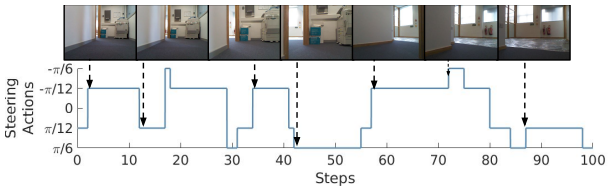
A. Training Efficiency with Different Models

To analyse the training efficiency and performance of the D3QN model and the advantage of introducing dueling and double techniques for obstacle avoidance, deep double Q network (DDQN) and DQN are compared. As shown in Fig.5, D3QN model outperforms the others two both on the training speed and performance. Unlike DQN whose average reward only reaches 10, networks with a double network structure learn policies with higher rewards. This may be because, for the obstacle avoidance problem, the overestimation of Q value is not a problem that can be alleviated by getting more exploration. Conversely, with a longer training period, it might be more severe, preventing DQN from obtaining high performance. Therefore, the D3QN architecture is about two times faster on training than the widely used normal DQN, which not only demonstrates its appealing performance on obstacle avoidance but also suggests an interesting direction of applying it on other robotic applications.

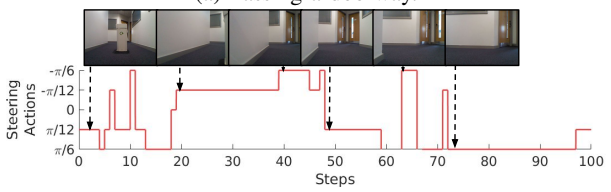
B. Real World Tests

Several experiments are conducted to directly test the trained models in real world.

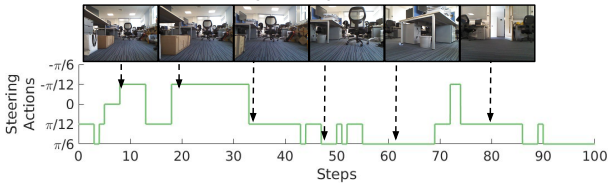
1) Action Prediction from Static Images: Firstly, we examine whether for arbitrary, complex scenarios, the network is able to predict a reasonable action that will avoid obstacles. As shown in Fig.6, a number of RGB images taken by a hand-held camera in a variety of environments including library, museum, attic and bedroom are used to predict actions. The bars in the figure indicate the Q value of each action: the first two bars are for linear velocity 0.2m/s and 0.4m/s, while the rest are for the five angular velocity $\frac{\pi}{6}$rad/s, $\frac{\pi}{12}$rad/s, 0rad/s, $-\frac{\pi}{12}$rad/s and $-\frac{\pi}{6}$rad/s. Note that these scenarios are more complicated than the simulation ones used for training and none of them has been "seen" by the model before. It can be seen that the trained D3QN model is capable of producing



(a) Passing a doorway.



(b) Driving through a corridor.



(c) In an office room.

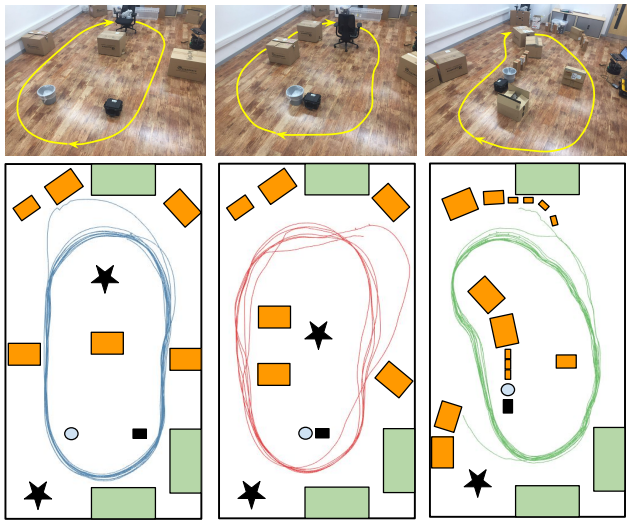
Fig. 7: Real world tests in three different scenarios. The curve below the image streams shows the steering actions selected by robots at each step.

reasonable actions to drive the robot according to the estimated Q values.

2) Tests in Three Different Scenarios: The trained D3QN model is tested for short-term (20s) navigation in three different scenarios including a doorway, a corridor and an office. The steering actions and some sample images of the three tests are given in Fig.7. Specifically, Fig.7a shows the procedure of the robot passing the doorway. Although the steering action of the robot is a little bit unstable when approaching an unseen obstacle (printer), it can still pass the doorway successfully. For the corridor case, an obstacle is placed in the middle of the corridor. As shown in Fig.7b, the robot can navigate smoothly through the narrow space between the obstacle and the wall. Similarly, robot can be controlled safely in an office room which is a more complicated environment with many previously unseen objects in the simulator. The experiments validate that the trained D3QN model is able to enable the robot to avoid obstacles by only using a monocular camera in different real environments by benefiting from knowledge learnt in virtual environments.

3) Tests in a Cluttered Environment: Several long-term experiments are conducted in a cluttered room to further test the performance with dynamic layouts and objects. A Vicon system is used to record the ground truth poses of the robot.

Fig.8 records the trajectories of the robot when it is operating around many obstacles. Green rectangles are fixed furniture while the orange ones are movable boxes. Other obstacles include two chairs (stars), a trashcan (circle) and a small suitcase (back rectangle). From the results we can see



(a) (b) (c)

Fig. 8: Real world tests in a room with different number and placement of obstacles. Rectangles show boxes while stars and circles are chairs and trash cans respectively.

that the robot usually chooses to go along a similar path. This is because that after the Q value of each state and action pair is predicted by network, the action is selected by a greedy policy, resulting a fixed policy for all states. Since the reward function defined in the training phase prefers going in a straight line than turning, the robot navigates as a loop with the smallest curvature to maintain a maximum linear speed.

Fig.9 presents the results when the robot is tested on two dynamic environments with different complexities. Although we tried to significantly change the dynamic objects in the environments, the robot was able to avoid them by using a monocular camera, which further verifies the effectiveness of the proposed method. The video of another test is available at https://youtu.be/qNIVgG4RUDM.

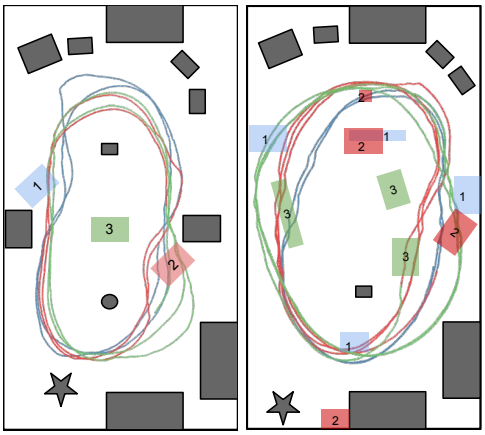
IV. Conclusion

In this paper, a deep reinforcement learning based algorithm is proposed for obstacle avoidance by only using monocular RGB images as input. The network can be trained solely in the simulator and then directly transferred to the real-world tasks. D3QN, which is based on dueling and double network techniques, demonstrates a higher learning efficiency than normal DQN in this task and can learn from very noise depth predictions. Extensive experiments in reality demonstrate the feasibility of transferring visual knowledge of the trained network from virtual to real and the high performance of obstacle avoidance by using monocular vision.

In the future, this network will be augmented to have a more complex structure and trained with auxiliary loss functions to learn tasks such as exploration and global navigation.

References

[1] Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single



(a) Simple (b) Complex

Fig. 9: Real world test in dynamic environments. The obstacles in gray are fixed while colored ones indicate the motion of movable obstacles at each time. The number on the obstacles indicates the changing sequence.

image perspective cues. In *ICRA*, pages 5776–5783. IEEE, 2011.

[2] Ronald Clark, Sen Wang, Niki Trigoni Andrew Markham, and Hongkai Wen. Vidloc: A deep spatio-temporal model for 6-dof video-clip relocalization. In *CVPR*, 2017.

[3] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *AAAI*, pages 3995–4001, 2017.

[4] CDR HR Everett. Survey of collision avoidance and ranging sensors for mobile robots. *RAS*, 5(1):5–67, 1989.

[5] Dieter Fox, Wolfram Burgard, and Sebastian Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics & Automation Magazine*, 4(1):23–33, 1997.

[6] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta. Learning to fly by crashing. *arXiv:1704.05588*, 2017.

[7] Alessandro Giusti, Jérôme Guzzi, Dan C Cireşan, Fang-Lin He, Juan P Rodríguez, Flavio Fontana, Matthias Faessler, Christian Forster, Jürgen Schmidhuber, Gianni Di Caro, et al. A machine learning approach to visual perception of forest trails for mobile robots. *RA Letters*, 1(2):661–667, 2016.

[8] Hado V Hasselt. Double q-learning. In *NPIS*, pages 2613–2621, 2010.

[9] Dong Ki Kim and Tsuhan Chen. Deep neural network for real-time autonomous indoor navigation. *arXiv:1511.04668*, 2015.

[10] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv:1412.6980*, 2014.

[11] Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *3DV*, pages 239–248. IEEE, 2016.

[12] Chris McCarthy and Nick Barnes. Performance of optical flow techniques for indoor navigation with a mobile robot. In *ICRA*, volume 5, pages 5093–5098. IEEE, 2004.

[13] Jeff Michels, Ashutosh Saxena, and Andrew Y Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *ICML*, pages 593–600. ACM, 2005.

[14] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, and et.al. Learning to navigate in complex environments. In *ICLR*, 2017.

[15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, and Alex et. al Graves. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[16] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE T-RO*, 31(5):1147–1163, 2015.

[17] Fereshteh Sadeghi and Sergey Levine. Rl: Real single-image flight without a single real image. arxiv preprint. *arXiv:1611.04201*, 12, 2016.

[18] Kahlouche Souhila and Achour Karim. Optical flow based robot obstacle avoidance. *IJARS*, 4(1):2, 2007.

[19] Lei Tai and Ming Liu. Towards cognitive exploration through deep reinforcement learning for mobile robots. *arXiv:1610.01733*, 2016.

[20] Lei Tai, Shaohua Li, and Ming Liu. A deep-network solution towards model-less obstacle avoidance. In *IROS*, pages 2759–2764. IEEE, 2016.

[21] Lei Tai, Giuseppe Paolo, and Ming Liu. Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation. *arXiv:1703.00420*, 2017.

[22] Iwan Ulrich and Illah Nourbakhsh. Appearance-based obstacle detection with monocular color vision. In *AAAI/IAAI*, pages 866–871, 2000.

[23] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, pages 2094–2100, 2016.

[24] Sen Wang, Ronald Clark, Hongkai Wen, Niki Trigoni, R Clark, S Wang, H Wen, N Trigoni, A Markham, A Markham, et al. Deepvo: towards end-to-end visual odometry with deep recurrent convolutional neural networks. In *ICRA*, 2017.

[25] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015.

[26] Shichao Yang, Sandeep Konam, Chen Ma, Stephanie Rosenthal, Manuela Veloso, and Sebastian Scherer. Obstacle avoidance through deep networks based intermediate perception. *arXiv:1704.08759*, 2017.

[27] Fangyi Zhang, Jürgen Leitner, Ben Upcroft, and Peter Corke. Vision-based reaching using modular deep networks: from simulation to the real world. *arXiv:1610.06781*, 2016.