Collision anticipation via deep reinforcement learning for visual navigation

Eduardo Gutiérrez-Maestro, Roberto J. López-Sastre, and
Saturnino Maldonado-Bascón

GRAM, University of Alcalá, Alcalá de Henares, Spain
http://agamenon.tsc.uah.es/Investigacion/gram/

Abstract. Visual navigation is the ability of an autonomous agent to find its way in a large and complex environment based on visual information. It is indeed a fundamental problem in computer vision and robotics. In this paper, we propose a deep reinforcement learning approach which is able to learn to navigate a scene to reach a given *visual* target, but anticipating the possible collisions with the environment. Technically, we propose a map-less-based model, which follows an actor-critic reinforcement learning method where the reward function has been designed to be collision aware. We offer a thorough experimental evaluation of our solution in the AI2-THOR virtual environment, where the results show that our proposed method: 1) improves the state of the art in terms of number of steps and collisions; 2) is able to converge faster than a model which does not care about the collisions, simply searching for the shortest paths; and 3) offers an interesting generalization capability to reach visual targets that have never been seen during training.

Keywords: visual navigation, deep reinforcement learning, robotics, computer vision

1 Introduction

We need robots to learn to navigate as we humans do: taking into account the environment and objects that surround us. This is the main objective of our work. The problem of navigation has been addressed in great depth in recent years. Most of the navigation solutions are considered as map-based methods, that is, a map of the environment is needed in order to make decisions for navigation (*e.g.* [1,2]). Others autonomously reconstruct a map of the environment on the fly and use it to navigate, *e.g.* [3,4,5], or use a human-guided experience to build the map, *e.g.* [6,7]. Finally, we find the map-less approaches, *e.g.* [8], which do not require a map as they do not have any assumption on the landmarks of the scenes.

Our approach follows this map-less principle, and we present a visual navigation solution which combines recent advances in deep and reinforcement learning. The main goal of our work is to boost robotic navigation towards a human-like behaviour. With this aim, the next question immediately arises: *how does a human learn to navigate within an indoor environment?* Look at Figure 1. Would

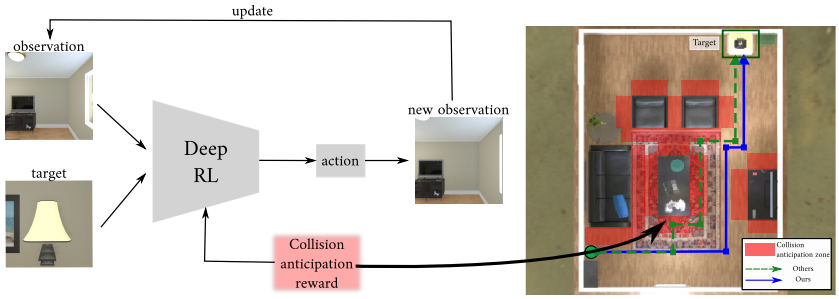


Fig. 1. Would a human in the virtual environment shown in the figure opt to follow the blue or the green path? We propose a deep reinforcement learning model which is designed to learn to navigate towards a visual target in a human-like way. Our model has to look for short trajectories, but also for paths where the collisions can be anticipated, maintaining a reasonable separation margin with the objects in the surroundings.

a human follow the blue or green path to reach the lamp? We believe that we humans tend to navigate avoiding possible collisions, anticipating them. In other words, we move trying to maintain a margin of separation with the possible obstacles of the environment that surround us.

In this paper we propose a deep reinforcement learning approach which is designed to navigate towards visual targets selecting the shortest path that is separated from possible obstacles. The **scientific contributions** of this work are as follows: **1)** We introduce a deep reinforcement learning collision aware solution, by the appropriate design of a novel reward function, which possibilitates to minimize the number of steps and to anticipate and avoid the collisions; **2)** Our thorough experimental evaluation shows that our solution improves the state of the art in the AI2-THOR virtual environment in terms of number of steps and collisions; **3)** Furthermore, results confirm that the proposed collision aware solution exhibits a better generalization capability and that is able to converge faster than a model which does not explicitly consider collisions during learning.

2 Related work

The evolution of technology in the field of computer vision has seen breathtaking changes, many of which have to do with the use of reinforcement learning based methods. Reinforcement Learning (RL) has experienced an increasing use in different applications. In [9], a RL-based solution is proposed for the real world problem of elevator dispatching. In [10], it is made use of RL to facilitate an autonomous flight. In [11], a vision-based method using RL is used to teach a robot to shoot a ball into a goal. In [12], it is investigated this type of algorithms in order to provide to a four-legged robot a walking behaviour.

We have also the family of methods that combine deep learning with RL. In [13], the authors make use of these methods to learn control policies from high-dimensional sensory inputs to play ATARI games. In [14], it is developed a system able to detect objects within a scene, in which the agent can find a specific bounding box by using deep reinforcement learning during the agent's training. In [8], the authors introduce a visual-based algorithm for learning how to navigate in indoor scenarios by using a virtual environment to train the models. Our approach directly leverages the model described in [8], to propose a novel visual-navigation solution able to anticipate collisions, hence providing the agent the knowledge to navigate in a secure way. We do this by integrating a collision aware reward function into an actor-critic deep RL model.

3 Navigation with Collision Anticipation

3.1 Navigation problem

The navigation solution proposed in this work is inspired by the hypothesis that we humans navigate trying to maintain a margin of separation with the possible obstacles of the environment that surround us (see again Figure 1).

In this work, we propose a system with this ability. We make use of deep reinforcement learning, designing a reward function which is able to transfer to the agent the policy knowledge that maps the visual sensory input into actions in a 3D world, so that both the number of steps and the number of collisions are minimized.

3.2 Model formulation

To achieve our goal we use a model based on deep reinforcement learning. This section details the different components of our approach to provide a solution that anticipates collisions in indoor environments while the agent moves.

As it is shown in Figure 1, the inputs to our model are just two images. Technically, we feed a deep siamese network, *i.e.* with shared weights, with both agent's observation and navigation goal images. This way, the target is implicitly in the input data.

We now define the three key parts of our deep reinforcement learning approach: i) action space; ii) observation and goal; and iii) reward function.

Action space. The objective of our visual navigation model is to map a high-dimensional 2D input into an action in a 3D space. Here we define the actions that our agent will take according to the input data. All the indoor environments used in this work have been discretized, so that, a *grid* of the world is created. For each cell within this grid, the agent is able to take four different actions: move forward, turn right, turn left, move backward.

Observation and goal. The agent is provided with a first-person-view camera. Note that each of the observations of the agent represents a state s_t at time t of all possible states $\mathcal{S}$ within the environment. The agent must navigate towards the position where the target image has been taken.

Reward. In reinforcement learning, the reward is defined as the feedback by which the success or failure of an agent's action is measured. Every time the agent selects an action, the environment returns a reward r_t at time t. In this work we propose a collision aware reward function, so that the agent is able to learn to anticipate collisions while it navigates. We first define the four main states that guide our reward assignment process. Our first state is called s_{step}, which corresponds to every time the agent executes an action in the navigation environment, *i.e.* it moves to a position in the grid, but considering the following exceptions. If the agent arrives to the target, the state is called s_{terminal}. If the agent is in a cell with an object or wall that could cause a collision by moving forward or backward, the state is named $s_{\text{collision}}$. Finally, if the agent is in a cell previous to a $s_{\text{collision}}$ state, *i.e.* in two cells before a collision could occur, we call it $s_{\text{collision}-1}$. Note how these last two states allow the agent to anticipate possible collisions with objects in the environment.

Our reward must be collision aware. Therefore, we design the following reward function that returns a different numerical value depending on the new state induced by the action taken by the agent at a certain time, *i.e.* s_t:

$$f(s_t) = \begin{cases} 10 \text{ if } \ s_t = s_{\text{terminal}} \\ \alpha \text{ if } \ \ s_t = s_{\text{step}} \\ \beta \text{ if } \ s_t = s_{\text{collision}} \\ \gamma \text{ if } s_t = s_{\text{collision}-1} \end{cases} . \tag{1}$$

The terminal state receives the maximum reward, indicating to the algorithm that the agent has arrived to the target. For the rest of states we assign different negative small values, being $\alpha = -0.01$, $\beta = -0.02$ and $\gamma = -0.011$. Note that for the collision anticipation states their associated negative values are higher than for the step stage. Our intention is that the agent learns to navigate away from those $s_{\text{collision}-1}$ and $s_{\text{collision}}$ states. In other words, the agent learns to anticipate the states where a collision could occur, which results in a safer navigation. Another way to understand the function of these states is shown in Figure 1. There, one can observe the top-view of a scene, where there are some red-shadowed areas. These are the zones we do not want our agent to navigate. This fact does not mean that the agent cannot navigate through those states, since the proposed algorithm of deep RL uses *cumulative* rewards. That is, we want the agent to find the combination of states that, at the end of the episode, arises the highest reward as possible. In other words, in certain situations the agent will decide to navigate through these states previously mentioned, so that the target will be achieved earlier. Thus we leave it to the RL model to decide which is the best option autonomously.

According to the model described, at each time step t, our agent receives a representation of the state of the environment, *i.e.* $s_t \in \mathcal{S}$. Based on that information the agent selects an action, $a_t \in \mathcal{A}(s_t)$, where $\mathcal{A}(s_t)$ represents all possible actions for state s_t. As a consequence of the action taken, the agent receives a reward, $r_t \in \mathcal{R}$, and discovers a new state s_{t+1}. The way an agent selects an action according to the current state follows a learning *policy*, which

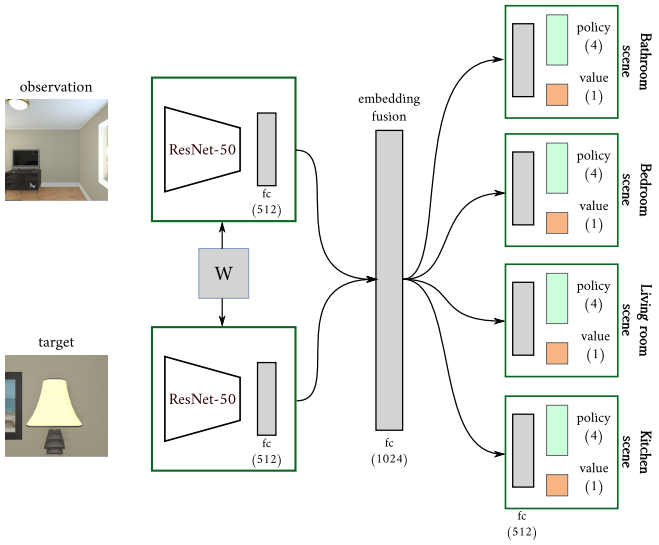


Fig. 2. Here it is shown the deep siamese network used in our work, which is fed by the agent's observation and the navigation target. Siamese layers are ResNet-50 features (truncated the softmax layer) pre-trained on ImageNet. ResNet parameters are frozen during the training stage. It can be appreciated the shared weights updated by each of the trained agents in the global network. At the right side of the figure, we have the specific network layers, which output the policy and value according to the visual sensory input for each particular type of scene.

is denoted by π_t. Therefore, $\pi_t(s_t, a_t)$ indicates the probability of the agent selecting the action a_t if the agent's state is s_t. The agent changes its policy through a reinforcement learning method, so that the agent gets the best amount of reward in each training episode.

Technically, we propose to follow an actor-critic learning model known as Asynchronous Advantage Actor-Critic (A3C) [15]. A3C is able to use multiple independent agents (in our case, deep networks) with their associated weights. These agents interact in parallel with different copies of the learning environment. In our case, each agent learns to navigate in the same environment but to a different target. Therefore, every agent is trained in parallel and updates in an asynchronous way the weights of a global network, which holds the shared parameters. Specifically, for the deep network of the agent we follow the siamese architecture proposed in [8], see Figure 2. Our objective is to use a deep siamese actor-critic network to capture the relations between the current agent's location and the target's location, by projecting them to the same embedding space. Therefore, we need the inputs of the two siamese networks to be the observation of the agent and the visual target. The features learned by the siamese networks are fused to build the joint representation that is used by the final scene specific

layers, which are in charge of generating the policy and value outputs of the actor-critic A3C algorithm.

4 Experiments

4.1 Experimental Setup

Dataset. We obtain the data to train and evaluate our model from AI2-THOR [16]. It is a framework that provides environments that look similar to real world scenes: a total of 120 different scenes covering four different categories (kitchens, living rooms, bedrooms, and bathrooms). It also includes actionable objects that an AI agent can interact with. For our *learning to navigate* objective, AI2-THOR results an excellent resource to provide our learning model with photo-realistic pictures of indoor domestic environments where our agent can learn to move. Technically, we follow the experimental setup released in [8], which comprises 4 different scenes, one for each room category.

Evaluation metric. To evaluate the navigation performance, we use, as in [8]: 1) the number of steps needed by the agent to reach the targets; and 2) the number of collisions during the navigation. For both metrics, the lower the better. We also propose a generalization experiment where we aim to evaluate if the agent is able to navigate to unseen targets during training. For this second experiment we use the success score (sc). This score is defined as follows. Given a scene and a target for that scene, with a fixed number of episodes, we compute sc as the number of episodes for which the agent is able to reach the target with a number of steps that is lower than 500. Since for each target we test the model one hundred times, the resulting sc measures the percentage of times that the agent reaches the goal below a fixed threshold of 500 steps.

4.2 Navigation results

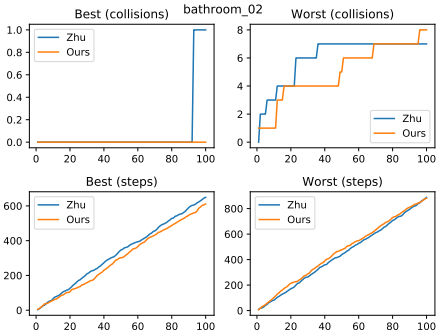
Navigation experiment. We compare our collision anticipation model with the state-of-the-art navigation model of Zhu *et al.* [8]. Note that in [8] the authors do not consider collisions during learning, optimizing the agent to find the shortest paths towards the targets only. Table 1 shows the main results.

First, one observes that the number of collisions considerably decreases (an order of magnitude) following our approach. This fact validates the proposed reward function integrated in the reinforcement learning approach. However, it is worth to note that our solution not only minimizes the number of collisions, but also the number of steps. In other words, our model is able to jointly seek the shortest trajectories with the fewest collisions.

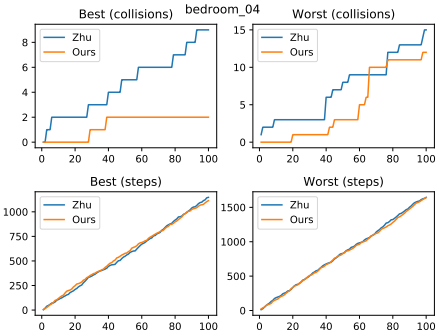
In Figure 3 we show how the number of collisions and number of steps are accumulated over 100 navigation episodes. For both methods (ours and Zhu [8]) we show the best and the worst cases. In terms of number of steps, we can conclude that both models obtain similar solutions. Since the agent starts the training in random positions within the scene, and taking into account the size

	Scene	Bathroom	Bedroom	Living_room	Kitchen	Average
Zhu [8]	Steps	7.17	14.82	15.2	21.38	14.7
	Collisions	0.04	0.12	0.25	0.2	0.15
Ours	Steps	7.33	14.81	14.9	20.83	**14.47**
	Collisions	0.03	0.04	0.15	0.12	**0.082**

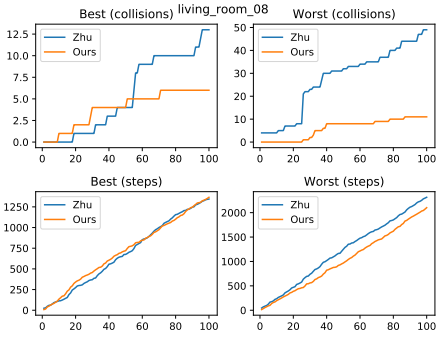
Table 1. Navigation results.



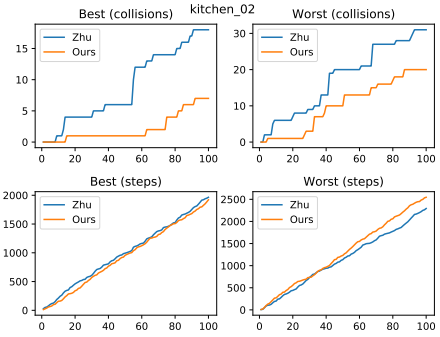
(a) Bathroom scene.



(b) Bedroom scene.



(c) Living room scene.



(d) Kitchen scene.

Fig. 3. Number of steps and collisions accumulated during the evaluation phase for each scene. We show the best and worst cases for each model.

of the scenes, which differs among the different categories, to outperform the state-of-the-art model proposed by Zhu *et al.* [8] in terms of number of steps is more complicated. However, our model is below this result as reported in table 1. But if collisions are considered, our agent clearly performs better, even for the worst cases. Finally, we show qualitative results in the following video[1].

[1] `https://www.youtube.com/watch?v=Eyxw-FY-iM0`

Fast convergence experiment. While training our model we are able to examine how the reward function evolves over time, and there is a fact that catches our attention: our model is capable of converging at least to a sub-optimal solution much faster than the approach of Zhu *et al.* [8].

To prove this in a quantitatively way, we proceed to evaluate the navigation performance of the models for 5 millions and 10 millions training frames. Figure 4 shows the average for the number of steps and collisions. One can conclude that our approach learns faster. Interestingly, in our model the number of steps reported for 5M is pretty close to the final one reported for 10M. And with respect to the collisions, it is worth to note that although the model of Zhu *et al.* [8] is able to also indireclty minimize them, our model drastically reduces them even for the initial stages of the learning process. As a conclusion: our collision anticipation navigation solution not only outperforms the results in terms of steps and collisions, but also learns faster.

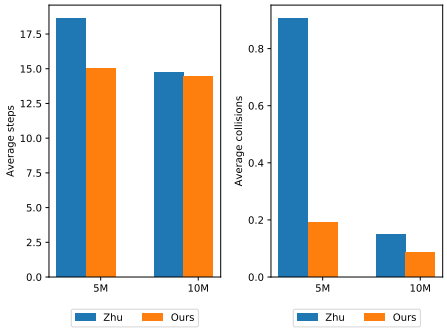


Fig. 4. Average steps and collisions for the fast convergence experiment.

Hard targets experiment. For all the previous navigation experiments we strictly follow the original experimental setup provided by Zhu *et al.* [8], using the fixed set of 5 different targets per scene selected by the authors. We here propose to perform an evaluation when a different set of targets is considered. We choose the new set of targets, with the aim of giving robustness to the main objective of our work: collision anticipation for visual navigation systems. For this purpose, in this experiment we select 5 targets per scene, which in terms of collisions and navigation, are harder to reach.

The results obtained for this experiment are shown in Table 2. Interestingly, the performance of both models, in terms of steps and collisions, decreases, a fact that confirms that the selected targets are actually harder than the original ones. Note that for the evaluation, the agent is thrown into random positions on the scenes, and we measure the number of collisions and steps until he reaches the target. The living room scene contains different objects (sofa, table, chairs, etc.) that difficult the navigation of both models, clearly. One can observe that the

	Scene	Bathroom	Bedroom	Living_room	Kitchen	Average
Zhu [8]	Steps	7.43	13.76	1323.88	19.17	341.06
	Collisions	0.06	0.15	247.78	0.30	62.07
Ours	Steps	7.29	14.63	568.78	18.89	152.39
	Collisions	0.03	0.08	61.66	0.25	15.5

Table 2. Evaluation of the model for the hard targets navigation experiment. 10M training frames.

performance of our model for the living room class is much better, making the difference between Zhu's model and ours. However, the number of steps needed for this scene is higher than 500 for our model. Overall, there is a huge gap between our model and Zhu *et al.* [8]: the trajectories obtained by our model are shorter and contain less collisions on average. If the methods want to report a similar performance to the original one (shown in Table 1, for the original easy targets), they must be trained for twice as many iterations (for 20M), as Table 3 reveals.

	Scene	Bathroom	Bedroom	Living_room	Kitchen	Average
Zhu [8]	Steps	7.46	13.92	18.12	18.03	14.38
	Collisions	0.04	0.12	0.2	0.19	0.14
Ours	Steps	7.16	14.15	17.73	17.86	14.225
	Collisions	0.01	0.03	0.05	0.07	0.04

Table 3. Evaluation of the model for the hard targets navigation experiment. 20M training frames.

4.3 Target generalization experiments

The generalization ability of deep reinforcement learning models is an interesting aspect to evaluate. So, the question here is: are our agents able to navigate towards targets that have *not* been considered during training? We design the following experiments with the objective of evaluating whether our collision anticipation navigation model generalizes appropriately.

We propose two different experiments. The first one evaluates the navigation of our model toward targets that are at 1, 2, 4 and 8 steps away from the original trained targets. In other words, what we have done here is to use the model trained for the original targets, whose results were reported in Table 1, but now, the evaluation is performed using *only* the new set of unseen targets. We measure the *sc* of the four scenes, and provide the average.

Figure 5 shows the main results. As expected, the higher the number of steps with respect to the trained targets, the lower the success score, since the problem becomes more difficult. If we compute the average for the 4 situations (1 step, 2

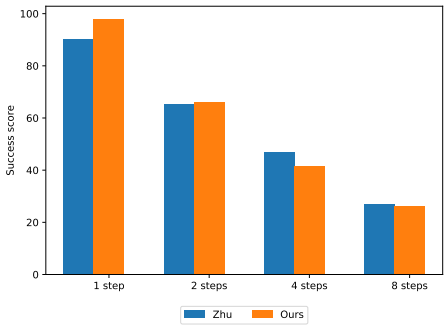


Fig. 5. Success score of the target generalization experiment.

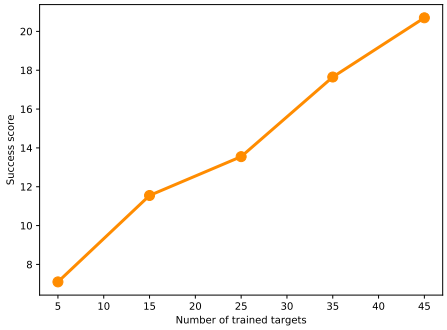


Fig. 6. Evolution of the success score obtained as the number of trained targets is increased.

steps, 4 steps and 8 steps) for every method, the model proposed by Zhu *et al.* [8] obtains $sc = 57,4\%$, while our model reports $sc = 58,1\%$.

Our second experiment aims to evaluate how the number of objectives used during training affects the ability to generalize. Technically, we select five different fixed targets per scene to evaluate the model. Then, we proceed to train our model for 5, 15, 25, 35 and 45 targets that are incrementally generated. We guarantee that the test targets are never included in the training sets, *i.e.* they always remain unseen during learning.

Figure 6 shows the *sc* of our model as the number of training targets increases. As expected, the *sc* increases when more and more training targets are considered. In conclusion, the generalization ability of the proposed deep RL navigation model needs of some improvements. A simple solution could consist on enriching the training set of targets, as this last experiment has confirmed. But, in any case, further efforts will be necessary in order that the agents achieve new targets, requiring a minimum adjustment of the parameters, or even that they can navigate in scenes never seen before.

5 Conclusions

This work proposes a deep RL approach whose main goal is to learn to navigate towards visual targets selecting the shortest path that is separated from possible obstacles, and hence anticipating collisions. According to the results shown in this work, we expose the following contributions. We have introduced a deep RL collision aware solution, by the appropriate design of a novel reward function. We have done an extensive experimental evaluation showing that our solution improves the state of the art in the AI2-THOR virtual environment in terms of number of steps and collisions. Results also confirm that our model is able to learn faster than a model which does not explicitly consider collisions during learning. Finally, the results show that the proposed collision aware solution exhibits a generalization capability that has to be improved to incorporate both new targets and scenes in the navigation.

Acknowledgments

This work is supported by project PREPEATE (TEC2016-80326-R), of the Spanish Ministry of Economy, Industry and Competitiveness. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPU used for this research.

References

1. Borenstein, J., Koren, Y.: Real-time obstacle avoidance for fast mobile robots. IEEE Transactions on Systems, Man, and Cybernetics **19**(5) (Sep. 1989) 1179–1187
2. Borenstein, J., Koren, Y.: The vector field histogram - fast obstacle avoidance for mobile robots. Robotics and Automation, IEEE Transactions on **7** (07 1991) 278 – 288
3. Wooden, D.: A guide to vision-based map building. IEEE Robotics Automation Magazine **13**(2) (June 2006) 94–98
4. J Davison, A.: Real-time simultaneous localisation and mapping with a single camera. Volume 2. (01 2003) 1403–1410
5. Tomono, M.: 3-d object map building using dense object models with sift-based recognition features. In: 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems. (Oct 2006) 1885–1890
6. Kidono, K., Miura, J., Shirai, Y.: Autonomous visual navigation of a mobile robot using a human-guided experience. Robotics and Autonomous Systems **40**(2) (2002) 121 – 130 Intelligent Autonomous Systems - IAS -6.
7. Royer, E., Bom, J., Dhome, M., Thuilot, B., Lhuillier, M., Marmoiton, F.: Outdoor autonomous navigation using monocular vision. In: 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems. (Aug 2005) 1253–1258
8. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. In: IEEE International Conference on Robotics and Automation. (2017)

9. Crites, R.H., Barto, A.G.: Improving elevator performance using reinforcement learning. In: Proceedings of the 8th International Conference on Neural Information Processing Systems. NIPS'95, Cambridge, MA, USA, MIT Press (1995) 1017–1023
10. Ng, A.Y., Coates, A., Diel, M., Ganapathi, V., Schulte, J., Tse, B., Berger, E., Liang, E.: Autonomous inverted helicopter flight via reinforcement learning. In Ang, M.H., Khatib, O., eds.: Experimental Robotics IX, Berlin, Heidelberg, Springer Berlin Heidelberg (2006) 363–372
11. Asada, M., Noda, S., Tawaratsumida, S., Hosoda, K.: Purposive behavior acquisition for a real robot by vision-based reinforcement learning. Machine Learning **23**(2) (May 1996) 279–303
12. Kimura, H., Yamashita, T., Kobayashi, S.: Reinforcement learning of walking behavior for a four-legged robot. IEEJ Transactions on Electronics, Information and Systems **122**(3) (2002) 330–337
13. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. In: NIPS Deep Learning Workshop. (2013)
14. Caicedo, J.C., Lazebnik, S.: Active object localization with deep reinforcement learning. In: The IEEE International Conference on Computer Vision (ICCV). (December 2015)
15. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Harley, T., Lillicrap, T.P., Silver, D., Kavukcuoglu, K.: Asynchronous methods for deep reinforcement learning. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48. ICML'16, JMLR.org (2016) 1928–1937
16. Kolve, E., Mottaghi, R., Han, W., VanderBilt, E., Weihs, L., Herrasti, A., Gordon, D., Zhu, Y., Gupta, A., Farhadi, A.: AI2-THOR: An Interactive 3D Environment for Visual AI. arXiv (2017)