A Comparison of Visual Navigation Approaches based on Localization and Reinforcement Learning in Virtual and Real Environments

Marco Rosano[1,2], Antonino Furnari[1], Luigi Gulino[2] and Giovanni Maria Farinella[1]
[1]*Università degli Studi di Catania, Catania, Italy*
[2]*OrangeDev s.r.l., via Vasco de Gama 91, Firenze, Italy*

Keywords: Visual Navigation, Reinforcement Learning, Image-based Localization, First Person Vision.

Abstract: Visual navigation algorithms allow a mobile agent to sense the environment and autonomously find its way to reach a target (e.g. an object in the environment). While many recent approaches tackled this task using reinforcement learning, which neglects any prior knowledge about the environments, more classic approaches strongly rely on self-localization and path planning. In this study, we compare the performance of single-target and multi-target visual navigation approaches based on the reinforcement learning paradigm, and simple baselines which rely on image-based localization. Experiments performed on discrete-state environments of different sizes, comprised of both real and virtual images, show that the two paradigms tend to achieve complementary results, hence suggesting that a combination of the two approaches to visual navigation may be beneficial.

1 INTRODUCTION

In robotics, one of the most required ability for an intelligent robot that has to operate in a given environment is to autonomously navigate inside it with a certain degree of accuracy. While humans can easily navigate in a wide variety of spaces and adapt to unseen environments, the autonomous robot navigation problem is still unsolved, even if several methods have been proposed over the years. Classic "map-based" approaches to visual navigation generally need 1) the construction of an explicit map of the environment and 2) a path planning strategy to exploit the acquired knowledge (Thrun et al., 2005).

To avoid the limitations of map-based approaches, recent works have tackled the navigation problem considering "map-less" methods , which only consider a form of implicit representation of the geometry of the space, usually obtained using Convolution Neural Networks (CNNs) directly trained on images of the environment (Mirowski et al., 2016; Zhu et al., 2017; Gupta et al., 2017). In particular, Deep Reinforcement Learning (DRL) models recently emerged as promising methods to learn navigation policies in a end-to-end manner (Mirowski et al., 2016; Zhu et al., 2017). Such methods consider high dimensional data as input (often in the form of images) and return the actions required to reach a specific target location. While we would expect DRL navigation approaches to be applied to real world contexts, due to the learning-by-simulation approach imposed by reinforcement learning, most research studies focus on simulated data (Mnih et al., 2013; Mirowski et al., 2016; Kempka et al., 2016), even able to recreate the appearance of real environments from images and to emulate interactions with the agent (i.e. collisions) (Xia et al., 2018; Savva et al., 2019). Despite such efforts, the substantial domain gap between real and simulated environments can harm the ability of these approaches to generalize to real environments.

In this paper, we compare map-based and map-less visual navigation approaches considering both single-target and multi-target settings. The considered map-based algorithms rely on a simple image-retrieval localization approach (Orlando et al., 2019) coupled with a path planning routine based on the computation of the shortest path between the current and target locations. The considered map-less approaches are based on reinforcement learning and consider both single (Konda and Tsitsiklis, 2000; Mnih et al., 2016) and multi-target variants (Zhu et al., 2017). We performed the experiments in environments comprising both real and simulated images, characterized by different sizes, varying the number of target states to be reached. In the case of multi-target methods, we tested the generalization ability to navigate to target states at given distances from the

Rosano, M., Furnari, A., Gulino, L. and Farinella, G.
A Comparison of Visual Navigation Approaches based on Localization and Reinforcement Learning in Virtual and Real Environments.
DOI: 10.5220/0008950806280635
In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2020) - Volume 5: VISAPP*, pages 628-635
ISBN: 978-989-758-402-2; ISSN: 2184-4321

ones seen during training. We report results in terms of success rate (whether the agent can reach the target in a limited number of steps) and average number of steps required to reach the target state. The considered approaches are compared with respect to two baselines: an ORacle agent (OR), which always follows the optimal path to the target state, and a Random Walker agent (RW), which follows a random policy to reach the target.

The results highlight that map-less approaches based on Reinforcement Learning can achieve optimal performances when navigating to targets seen during training, but are limited when it comes to generalizing to unseen targets and scaling to large environments. Map-based approaches based on localization obtain encouraging and complementary results, despite the poor performance of the localization module alone, which suggests that combining approaches based on RL and localization can be beneficial.

The remainder of the paper is organized as follows. Section 2 describes the related works. In Section 3, we describe the compared approaches. Experimental settings are discussed in Section 4 and results in Section 5. Section 6 concludes the paper and discusses future works.

2 RELATED WORKS

Previous works have investigated visual navigation according to two main settings: map-based and map-less. Map-based navigation relies on a map of the environment either created beforehand or built on the fly during navigation. Map-less navigation relies on Deep Neural Networks to extract an implicit representation of the world from images or other sensory input. This representation is then used to perform the navigation task. The learning process can be performed using Imitation Learning (IL) and Reinforcement Learning (RL) as discussed in the following.

Map-based Visual Navigation. Methods falling into this category assume a map of the environment to be known and employ path planning (Hong Zhang and Ostrowski, 2002) or obstacle avoidance algorithms (Ulrich and Borenstein, 1998) to navigate to the destination. Convolutional Networks have been also used to create a top-down spatial memory from first-person views (Gupta et al., 2017) or to localize the agent using image-based localization techniques (Kendall et al., 2015; Orlando et al., 2019), and then apply a path-planning algorithm to find the optimal route to the target.

Image-based Localization. Image-based localization plays a central role in the task of sensing an environment by an autonomous robot. Localization methods based on classification usually discretize the environment by dividing it into classes and address localization as a classification task (Ragusa et al., 2019). Despite the good performances of these approaches, they are not able to provide an accurate estimation of the camera pose. Image-retrieval methods rely on image representation techniques (Jégou et al., 2010; Arandjelovic et al., 2016) and reduce the pose estimation problem to a nearest neighbor search in the feature space (Orlando et al., 2019). Image localization can also be tackled as a regression problem, where a CNN is used to directly estimate the camera pose from images (Kendall et al., 2015). 2D-3D matching approaches start by extracting 2D feature points from images, then create a correspondence between these 2D points and 3D points of a given model of the scene. The 3D model could be known beforehand (Sattler et al., 2016) or incrementally constructed (Schnberger and Frahm, 2016).

Map-less Visual Navigation based on Imitation Learning. These approaches allow to plan the sequence of actions to be performed directly from raw images of the environment, taking advantage of the ability to learn deep models end-to-end. In Imitation Learning, the policy is obtained from expert demonstrations, as in a classic supervised learning setup (Bojarski et al., 2016; Giusti et al., 2015). Unfortunately, this approach often leads to unstable policies, since the model is hardly able to recover in case of drift (Ross et al., 2010). To overcome the problem of unseen situations due to limited demonstrations, several strategies have been adopted to increase the number of labelled samples (Bojarski et al., 2016; Giusti et al., 2015).

Map-less Visual Navigation based on Reinforcement Learning. In navigation approaches based on Reinforcement Learning (RL), the agent starts exploring the environment following a random navigation policy and learns the optimal set of actions by receiving a positive reward signal when it reaches the goal state, after performing several navigation episodes. In the case of single-target navigation, the model learns to navigate to only one destination and the policy optimization depends only on the collected egocentric views acquired along the trajectory (Mnih et al., 2016). In the case of multi-target navigation, the optimization of the policy also depends on a target image, which is given as input together with the current state (Zhu et al., 2017). The need of a single model capable of learning a multi-target navigation policy drove the authors of (Zhu et al., 2017) to propose a DRL model trained considering as input both an image of the the target state and the current state repre-

sentation in a indoor simulated environment.

3 METHODS

We compare the following approaches to visual navigation: i) map-less single-target Reinforcement Learning; ii) map-less multi-target Reinforcement Learning; iii) map-based navigation using image-retrieval for localization. Please note that the last approach is by design a multi-target approach, as it does not require a target-specific policy learning. We further compare these approaches with respect to two baselines consisting in a Random Walker agent (RW) which follows a random navigation policy and an Oracle agent (OR) which always follows the shortest path to the destination.

Single Target Reinforcement Learning Method (A2C). In classic RL models the goal is fixed (Mnih et al., 2013; Kempka et al., 2016). To perform simulations, the starting state is randomly sampled from the set of all states in order to obtain a better generalized policy across starting locations. In our experiments, we trained a A2C actor-critic model (Konda and Tsitsiklis, 2000) to accomplish the navigation task.

Multi Target Reinforcement Learning Method (A3C). Recent works tackled the multi-goal navigation problem designing models that consider as input both the current state and an RGB image of the target location. The policy is hence learned on the pair of current and target images. In our experiments, we used the method proposed in (Zhu et al., 2017) to handle multi-target navigation.

Method based on Visual Localization (LB). This method relies on a localization module which allows to estimate the pose of an image inside an environment performing a nearest neighbor search in the representation space. Based on the estimated location, the shortest path to the destination is computed, from which we can derive the optimal sequence of actions to be taken to reach the target. To perform localization, we rely on a separate set of images, in which each image has been attached its position in the environment using *Structure From Motion* (Schnberger and Frahm, 2016). Then image representations are extracted using VGG16 (Simonyan and Zisserman, 2015) CNN pre-trained on ImageNet and the position of the agent in the environment is estimated using a nearest neighbor search in the representation space.

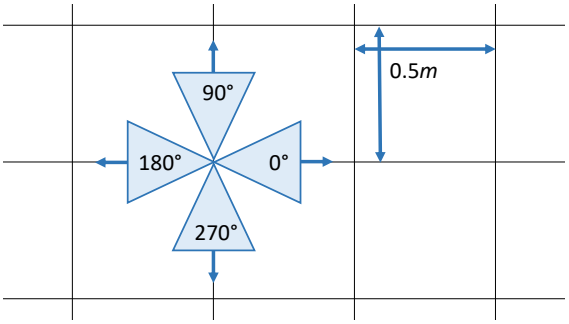

Figure 1: Images of the environments were collected following a grid pattern with a step size of $0.5m$. For each point of the grid, we collected four images at the four main orientations: $[0°, 90°, 180°, 270°]$.

4 EXPERIMENTAL SETTINGS

In this section, we report details about our experimental setup, including the environments and how the navigation simulations are performed, the training details of the navigation methods, and the evaluation procedure.

4.1 Environments and Simulations

To perform experiments on both synthetic and real environments, we follow the setup of (Zhu et al., 2017). This setup involves running the simulations on a grid of possible states, sampled at a regular step of $0.5m$. Each point of the grid corresponds to four states characterized by the same position (the position on the grid), combined with four possible orientations: $[0°, 90°, 180°, 270°]$. For each state (position-orientation pair), an RGB image is collected whether from the real-world or virtual environment. Figure 1 provides a visual example of the grid pattern used to collect the images. Each collected image represents one discrete state in which the agent can be. The agent can navigate through the states by performing four possible actions: *go forward, turn left, turn right, go backward*. A state-transition matrix specifies which state can be reached from another state performing one of the actions and which states do not allow to perform any action. A simple routine which allows to retrieve the image observed of the next state when taking an action at the current state is used both at training and test time to simulate navigation. We refer to the execution of one of the action in a given state as a "step".

For our experiments, we considered the four virtual environments proposed in (Zhu et al., 2017). Given the scarcity of datasets of real images for visual navigation (Zhu et al., 2017), we also collected

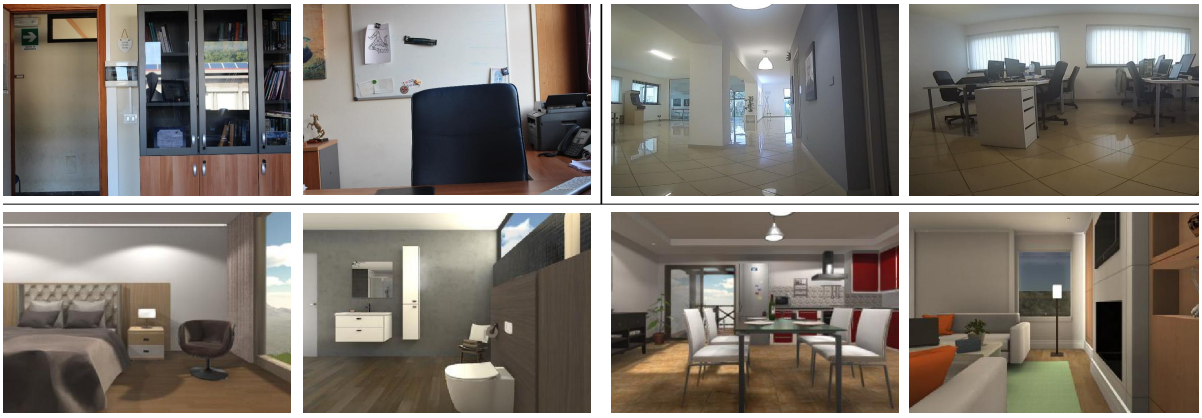

Figure 2: Sample RGB images of the real and virtual environments considered in this study. The Real Small Environment (RSE) (top left) consists of 148 images/states. The Real Big Environment (RBE) (top right) consists of 979 images/states. The virtual environments (bottom) have different sizes (see Table 1).

Table 1: List of the environments used to perform our experiments along and related properties.

Env. name	Real	#states	Source
RSE	yes	148	this work
bathroom_02	no	180	(Zhu et al., 2017)
bedroom_04	no	408	(Zhu et al., 2017)
living_room_08	no	468	(Zhu et al., 2017)
kitchen_02	no	676	(Zhu et al., 2017)
RBE	yes	979	this work

images from two additional real environments for this study. The list of all environments and their properties is reported in Table 1. The Real Small Environment (RSE) consists of images of a small office, collected with a reflex camera at a resolution of 5184×3456 pixels. The Real Big Environment (RBE) consists of images of an open-plan office, collected using a robotic platform[1] with an on-board camera at a resolution of 1192×670 pixels. All the other environments in Table 1 have been acquired from (Zhu et al., 2017). Sample images from the considered environments are shown in Figure 2.

4.2 Navigation Methods based on Reinforcement Learning

Feature Extraction. We run all RL methods on features extracted from the input images. To this purpose, each RGB image has been resized to 400×300 pixels. The image is then processed by a ResNet-50 CNN (He et al., 2016), pre-trained on ImageNet (Deng et al., 2009) to extract its representation vector. This has been done by removing the last classification layer from the CNN in order to obtain 2048-dimensional representation vectors.

Training. Each of the considered models gives as output the value of the current state and a probability distribution over the 4 actions which can be performed by the agent: *go forward*, *turn left*, *turn right*, *go backward*. Training is performed by sampling a number of targets proportional to the environment size, following a uniform distribution. The second column of Table 2 reports the number of target states used for training on each environment. The agent is then placed at a new random starting position and navigates to the target state following the current policy. The episode ends when the agent reaches the target state (success) or when the maximum number of allowed steps (set to 5000) is reached (fail). The target state reward was set to 20, whereas we introduce a penalty equal to -0.1 for each navigation step or collision. At the end of each episode, the policy is updated to maximize the cumulative reward.

A2C - Architecture and Testing. The architecture consists of 2 fully connected layers with 512 and 128 units respectively, Rectified Linear Units (ReLUs) as activation functions, and an output layer with 5 units (4 to represent a probability distribution over actions and 1 to represent the value of the current state). Since the method is single-target, we trained as many models as the number of targets over all environments. This amounts to a total of 33 models. We trained all the models for 50.000 episodes, obtaining a good convergence to the optimal policy. At test time, for each target state, we performed different navigation trials starting from different random initial states. The third column of Table 2 reports the number of initial states selected per target state, for each environment.

A3C - Architecture and Testing. This method follows the architecture proposed in (Zhu et al., 2017).

[1]We used the Sanbot Elf Robotic Platform. *http://en.sanbot.com*.

Table 2: Number of targets seen during training and number of test trials performed for each target. In the case of multi-target methods, test targets are sampled at given distances from the ones seen during training. See text for details.

Env. name	# target states	# test trials (x target)
RSE	3	3
bathroom_02	3	3
bedroom_04	5	5
living_room_08	5	5
kitchen_02	7	7
RBE	10	10

As proposed by the authors, we feed the model with the concatenated representations of the images of the last 4 visited states to provide a history to the model. The concatenated features are hence fed to a layer of 512 units. The target to reach is specified by providing also the representation of the target image as input. As suggested in (Zhu et al., 2017), this input is replicated 4 times to balance with respect to the number of history images and fed to another layer of 512 units. The 512-dimensional resulting embeddings are concatenated and passed to the final part of the architecture, composed of 2 fully connected layers with 512 units each and a final output layer with 5 units to represent a probability distribution over actions and the value of the current state. Since the method is multi-target, we trained only one model for each environment until convergence, for a total of 6 models. At test time, to evaluate the ability of the model to generalize to unseen targets, we sampled a set of testing targets at several distances (0, 1, 2, 4, 8 steps) from the ones used for training. For each of the selected testing target, we performed different navigation trials starting from different random initial states. The third column of Table 2 reports the number of initial states selected in each trials.

4.3 Navigation Method based on Localization

We compared the approaches based on Reinforcement Learning with respect to a more classic navigation approach which relies on a visual localization module based on image-retrieval. The goal of the localization module is to determine in which state the agent is located from the observation of the current state. Once this information is obtained, the agent navigates to the target by following the minimum path computed with the Dijkstra algorithm (Cormen et al., 2001). Since the localization algorithm may be inaccurate and fail in some circumstances, we repeat localization and computation of the minimum path every 5 steps, when an invalid action is performed or when a loop in the followed path is detected.

To perform the image-based localization, we collected an additional set of images of the RBE environment, consisting of 4072 RGB images, with a resolution of 1192×670 pixels. The images have been acquired along random straight trajectories that covered the entire space. A 3D model of the environment has been created using *Structure From Motion* (Schnberger and Frahm, 2016). The model has then been aligned to a map of the environment to correct for translation, orientation and scale. This process allows to label each of the images with a 3DOF pose which can be used for localization. Starting from this set of images, we created a secondary regular grid of images following the grid pattern used to acquire the main dataset and sampling the nearest image to each grid-point, in terms of euclidean distance and absolute angle difference. We extracted representation vectors from each image of both the environment and the secondary set of images using a VGG-16 CNN (Simonyan and Zisserman, 2015) pre-trained on ImageNet (Deng et al., 2009), after resizing them to a resolution of 256×256 pixels and applying a center crop, obtaining a final size of 224×224 pixels. The final classification layer of the CNN has been removed to obtain 4096-dimensional representation vectors. Given the image of the current state, the localization is performed with a nearest neighbor search in the representations space, which allows to estimate the state in which the actor is currently located.

4.4 Evaluation

We evaluated the performances of the different models in terms of average number of steps required to reach the target and success rate (i.e., whether the agent can reach the target in a limited number of steps). All results have been obtained by averaging the performance scores over all episodes obtained starting from the randomly sampled locations. We set a threshold of 100 steps to determine if an episode is successful or not. We think this value is reasonable to allow the agent to navigate without a too strict restriction, but still in a limited number of steps. It is important to point out that all target and starting states have been initially sampled and saved. All models have been hence evaluated using the previously sampled starting/target pairs for fair comparison.

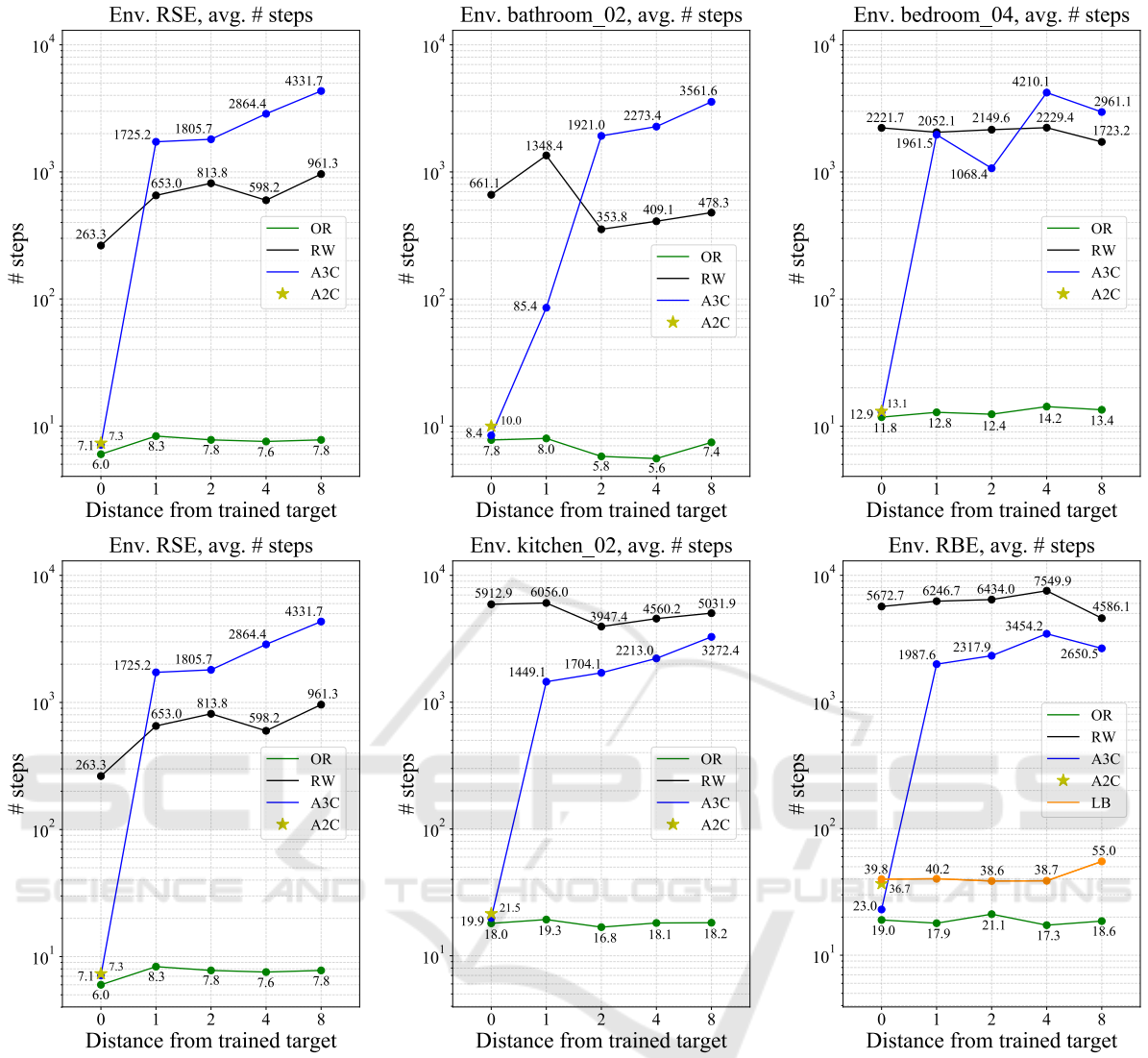

Figure 3: Performances of visual navigation methods in terms of average number of steps required to reach the target states from random initial positions. The target states have been sampled at a distance of 0, 1, 2, 4, and 8 steps from the ones used during training in the case of multi-target methods. This is a graphical representation of the results reported in Table 3.

5 RESULTS

Table 3 report the results of the considered visual navigation methods in terms of average number of steps and success rate required to reach the targets in the different environments. Figure 3 further shows the average number of steps in a visual form. The A3C model achieves near-optimal results, comparable to the ones obtained by the OR strong baseline on targets seen during training (distance 0). The performances of the A2C and A3C models on targets seen during training (distance 0) are comparable over all environments, except for RBE. This suggests that A3C can benefit more from the additional training signal provided by larger and more complex environments. It is worth noting tat the A3C algorithm shows a good ability to learn multiple optimal policies to different target states of the same environment within a single model. This is different from A2C, which can reach similar performance on most of the environments learning a different policy for each target. This makes A3C more data and computation efficient, requiring to train only one model per environment, rather than one model per target. Somewhat surprisingly, despite the good results obtained on targets seen during training, the performances quickly deteriorate when the target states

Table 3: Performances of visual navigation methods in terms of average number of steps and success rate (in parentheses) required to reach the target states from random initial positions. The target states have been sampled at a distance of 0, 1, 2, 4, and 8 steps from the ones used during training in the case of multi-target methods.

Env. RSE

Distance	A3C	A2C	RW	OR
0	7.1 (100)	7.3 (100)	263.3 (11)	6.0 (100)
1	1725.2 (44)	-	653.0 (11)	8.3 (100)
2	1805.7 (33)	-	813.8 (0)	7.8 (100)
4	2864.4 (33)	-	598.2 (0)	7.6 (100)
8	4331.7 (0)	-	961.3 (11)	7.8 (100)

Env. bedroom_04

Distance	A3C	A2C	RW	OR
0	12.9 (100)	13.1	2221.7 (8)	11.8 (100)
1	1961.5 (52)	-	2052.1 (4)	12.8 (100)
2	1068.4 (60)	-	2149.6 (12)	12.4 (100)
4	4210.1 (8)	-	2229.4 (8)	14.2 (100)
8	2961.1 (4)	-	1723.2 (12)	13.4 (100)

Env. kitchen_02

Distance	A3C	A2C	RW	OR
0	19.9 (100)	21.5 (100)	5912.9 (4)	18 (100)
1	1449.1 (61)	-	6056 (8)	19.3 (100)
2	1704.1 (38)	-	3947.4 (6)	16.8 (100)
4	2213 (22)	-	4560.2 (8)	18.1 (100)
8	3272.4 (6)	-	5031.9 (8)	18.2 (100)

Env. bathroom_02

Distance	A3C	A2C	RW	OR
0	8.4 (100)	10 (100)	661.1 (11)	7.8 (100)
1	85.4 (66)	-	1348.4 (33)	8 (100)
2	1921 (44)	-	353.8 (33)	5.8 (100)
4	2273.4 (11)	-	409.1 (22)	5.6 (100)
8	3561.6 (22)	-	478.3 (33)	7.4 (100)

Env. living_room_02

Distance	A3C	A2C	RW	OR
0	15.8 (100)	19.2 (100)	2836 (8)	14.2 (100)
1	504.9 (64)	-	3780.6 (4)	15.8 (100)
2	3950 (0)	-	2766.3 (4)	15.4 (100)
4	3117.4 (28)	-	3117.4 (4)	15.4 (100)
8	2563.2 20)	-	4322.9 (16)	15.6 (100)

Env. RBE

Distance	A3C	A2C	LB	RW	OR
0	23 (100)	36.7 (100)	39,8 (98)	5672.7 (7)	19 (100)
1	1987.6 (32)	-	40.2 (95)	6246.7 (3)	17.9 (100)
2	2317.9 (15)	-	38.6 (97)	6434 (3)	21.1 (100)
4	3454.2 (19)	-	38.7 (96)	7549.9 (5)	17.3 (100)
8	2650.5 (32)	-	55 (95)	4586.1 (7)	18.6 (100)

are sampled even at small distances from the trained ones. These results are sub-optimal even with respect to the RW baseline for smaller environments in terms of number of steps (see Table 3 and Figure 3), whereas marginally lower numbers of steps are required in large environments compared to RW. Nevertheless, when the models are evaluated in terms of success rate, A3C is consistently worse than RW independently from the type/size of environment. This suggests that 1) the multi-target A3C model needs to see large amounts of data in order to even partially generalize to unseen targets and, 2) it is more prone to learn a limited number of navigation policies related to the targets seen during training, rather than a generalized policy useful for navigating to unseen targets. Moreover, we noticed that the model hardly converged to optimal trajectories when trained on a number of targets greater than 10, implying a limited ability of the training procedure to scale to a larger number of targets/policies. In addition, the success rate of A3C on the larger environment report an average value lower than 0.4 for targets at a distance of 1 step. The results suggest that more efforts should be devoted to designing RL models able to learn a better state representation space, in order to have a consistent neighbourhood relationship between near states, to efficiently transfer the learnt knowledge to previous unseen samples.

We finally compare the approaches based on reinforcement learning with the baseline relying on visual localization (LB) on RBE. As can be assessed from Figure 3 and Table 3, LB achieves performances comparable to A2C and suboptimal with respect to the ones obtained by A3C when the targets are the ones seen during the training of the RL methods. Notably, while the performances of A3C quickly deteriorate for targets not seen during training, the performances of LB remain overall constant. This is due to the fact that, since the policy is not explicitly learned in the case of LB, the method does not suffer from over-fitting to a specific target. Additionally, when evaluated in terms of success rate, the LB approach achieves results close to 100% and comparable to the ones obtained by the OR baseline. While such advantages over RL approaches are obtained at the expenses of building a visual localization system, it should be noted that methods based on localization can achieve good results even in the presence of poor localization. Indeed, the visual localization method of LB achieved an average localization error of 3.86m and an average orientation error of 53.85°. The complementarity of the results obtained by methods based on reinforcement learning and the baseline relying on visual localization suggest that better results, especially for the multi-target case, can be obtained from the integration of the two approaches. For instance, RL methods could benefit from a coarse localization obtained using simple techniques such as image-retrieval. We leave the investigation of such integration to future works.

6 CONCLUSION

In this paper, we compared visual navigation methods based on reinforcement learning and localization. We performed experiments on differently sized discrete-state environments composed of both virtual and real images. The results suggest that, despite the availability of multi-target approaches, visual navigation methods based on reinforcement learning have difficulties to generalize to targets unseen during training. On the contrary, a simple baseline which relies on inaccurate localization achieves similar results on targets seen during training and generalizes better to unseen targets. These observations suggest that methods based on reinforcement learning could benefit even from inaccurate localization. Future works can investigate approaches to fuse visual navigation methods based on reinforcement learning and localization.

ACKNOWLEDGEMENTS

This research is supported by OrangeDev s.r.l. and Piano della Ricerca 2016-2018, Linea di Intervento 2 of DMI, University of Catania.

REFERENCES

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T., and Sivic, J. (2016). Netvlad: Cnn architecture for weakly supervised place recognition. In *CVPR*, pages 5297–5307.

Bojarski, M., Testa, D. D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L. D., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., and Zieba, K. (2016). End to end learning for self-driving cars. *CoRR*, abs/1604.07316.

Cormen, T. H., Stein, C., Rivest, R. L., and Leiserson, C. E. (2001). *Introduction to Algorithms*. McGraw-Hill Higher Education, 2nd edition.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR*.

Giusti, A., Guzzi, J., Cireşan, D. C., He, F.-L., Rodríguez, J. P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Di Caro, G., et al. (2015). A machine learning approach to visual perception of forest trails for mobile robots. *RA-L*, 1(2):661–667.

Gupta, S., Davidson, J., Levine, S., Sukthankar, R., and Malik, J. (2017). Cognitive mapping and planning for visual navigation. In *CVPR*, pages 7272–7281.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *CVPR*, pages 770–778.

Hong Zhang and Ostrowski, J. P. (2002). Visual motion planning for mobile robots. *T-RA*, 18(2):199–208.

Jégou, H., Douze, M., Schmid, C., and Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *CVPR*, pages 3304–3311.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. (2016). Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *CIG*, pages 1–8.

Kendall, A., Grimes, M., and Cipolla, R. (2015). Posenet: A convolutional network for real-time 6-dof camera relocalization. In *ICCV*, pages 2938–2946.

Konda, V. R. and Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *NIPS*, pages 1008–1014.

Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A. J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., Kumaran, D., and Hadsell, R. (2016). Learning to navigate in complex environments. *CoRR*, abs/1611.03673.

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *ICML*, pages 1928–1937.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

Orlando, S. O., Furnari, A., Battiato, S., and Farinella, G. M. (2019). Image-based localization with simulated egocentric navigations. In *VISAPP*.

Ragusa, F., Furnari, A., Battiato, S., Signorello, G., and Farinella, G. (2019). Egocentric visitors localization in cultural sites. *JOCCH*, 12:1–19.

Ross, S., Gordon, G., and Bagnell, J. (2010). A reduction of imitation learning and structured prediction to no-regret online learning. *JMLR*, 15.

Sattler, T., Leibe, B., and Kobbelt, L. (2016). Efficient & effective prioritized matching for large-scale image-based localization. *PAMI*, 39(9):1744–1756.

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., et al. (2019). Habitat: A platform for embodied ai research. *arXiv preprint arXiv:1904.01201*.

Schnberger, J. L. and Frahm, J. (2016). Structure-from-motion revisited. In *CVPR*, pages 4104–4113.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *ICLR*.

Thrun, S., Burgard, W., and Fox, D. (2005). *Probabilistic robotics*. MIT press.

Ulrich, I. and Borenstein, J. (1998). Vfh+: Reliable obstacle avoidance for fast mobile robots. In *ICRA*, volume 2, pages 1572–1577.

Xia, F., Zamir, A. R., He, Z., Sax, A., Malik, J., and Savarese, S. (2018). Gibson env: Real-world perception for embodied agents. In *CVPR*, pages 9068–9079.

Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A. (2017). Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*, pages 3357–3364.