2025 IEEE/SICE International Symposium on System Integration (SII)
January 21-24, 2025, Munich, Germany

Exploring Robot Trajectories in Panoramic Vision-Based Control Using Deep Reinforcement Learning

Takieddine Soualhi*, Nathan Crombez, Yassine Ruichek, Alexandre Lombard, Stéphane Galland

***Abstract*— In this paper, we study the problem of direct trajectories in visual control of nonholonomic mobile robots. To address this challenge, we propose combining deep reinforcement learning with panoramic vision to learn a control policy that maps input images to control velocities. We demonstrate that the adopted approach can precisely drive the robot to a desired pose. Additionally, it enables the emergence of various control strategies to achieve interesting trajectories. Our approach is validated and evaluated in simulation and transferred to the real world.**

I. Introduction

Nonholonomic robots belong to a class of mobile robots that possess motion constraints arising from their physical architecture. These constraints limit the robots ability to reach arbitrary poses in its configuration space. The visual servoing (VS) of nonholonomic mobile robots is an established problem within robotics [1]. VS framework focuses on using visual input to regulate the robot at a given desired pose [2]. This is achieved by reducing the discrepancy between visual features obtained from a desired image and those captured by the robot as it moves. The main issue in solving VS of nonholonomic mobile robots is the motion constraints, which make it challenging to reach the desired pose without losing essential visual information from the cameras field of view (FOV) [3].

A prominent approach to tackle this problem is to increase the camera's FOV. For example, it has been proposed to mount a perspective camera on a pan and tilt unit [4], in order to track the desired visual content during the VS process. Using wide FOV cameras, i.e., fisheye lens or catadioptric systems, have also been studied, ensuring that the desired visual content is almost always visible despite the robot's displacement [1], [5], [6]. However, these methods present several limitations, such as reliance on feature tracking and matching or a limited convergence domain. Moreover, in classical VS, controlling the trajectory of the robot requires complex analytic control approaches.

Recently, deep reinforcement learning (DRL) has emerged as a prominent framework to address the challenges associated with vision-based robotics [7]. In parallel, panoramic vision have made significant contributions to the field of robotics, providing a perception of the whole surrounding at once [8]. In this context, we investigate solving VS problem for nonholonomic robots using panoramic images.

* Corresponding author.
Takieddine Soualhi, Nathan Crombez, Yassine Ruichek, Alexandre Lombard and Stéphane Galland are with Belfort-Montbéliard University of Technology, UTBM, CIAD, 90010 Belfort, France, e-mail: {firstname}.{name}@utbm.fr

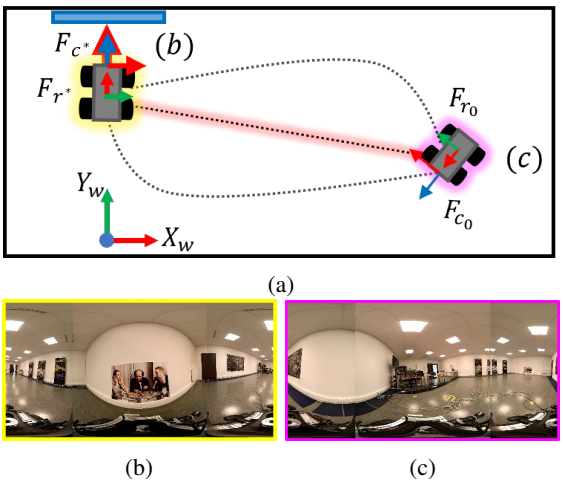

Fig. 1: Illustration of the problem: (a) An example scenario where a robot has to find the desired pose following the most direct trajectory highlighted by the red line, (b) The desired panoramic image captured at the desired pose, (c) The initially observed panoramic image.

Specifically, we focus on learning a control policy that is able to precisely regulate the robot at a given desired pose while maintaining an optimal trajectory. We consider the optimal trajectory to be the straight line defined between the initial position of the robot and the desired one (Fig. 1). Such trajectories require the robot to be able to decouple the control of its angular and linear control velocities.

Our contributions can be summarized as follows:

- Application of DRL to solve VS for nonholonomic mobile robots with a panoramic camera.
- Study of the learned trajectories based on the structure of the employed reward function.
- Comparison with traditional image-based VS and the existing learning based VS approach, both with a panoramic camera.

The remainder of this paper is organized as follows. Section II presents the related work to this paper. Section III highlights the formulation of the problem within a DRL framework. Subsequently, Section IV presents the employed DRL approach, including the DRL model and the reward function design. Section V outlines the experimental setup and provides details on the training process. In Section VI, we present the outcomes of the simulations, offer comparisons, and discuss the results of real-world transfer. Finally, Section VII summarizes the conclusions, addresses the lim-

979-8-3315-3161-4/25/$31.00 ©2025 IEEE

itations, and suggests directions for future research.

II. Related work

This section is dedicated to showcasing related work for this paper, focusing literature that has applied DRL to VS in general, as well as the literature that combines DRL approaches and panoramic vision to address robotics problems.

Several works have been proposed to solve VS using DRL approaches. Sayesh et al. [9] introduced a method that combines convolutional neural networks (CNNs), DRL, and optimal control to address VS for robotic manipulators, demonstrating the potential of DRL-based methods compared to classic VS techniques . This was shown to be further improved by coupling DRL with imitation learning, particularly through the integration of demonstrations from multiple expert controllers [10]. The problem of deploying DRL-based VS methods on real robots was addressed in [11], where the authors studied generalization and real-time operation capacities of learning-based VS methods. While, these outlined methods studied mainly the VS of robotics manipulators. Li et al. [12] tackled the problem of VS for mobile robots by introducing a deep Q-network-based DRL model for learning VS policies. Nevertheless, similar to previous approaches, it primarily focused on perspective vision sensors .

In the exploration of robot navigation using panoramic images and DRL, several key works were proposed. Watkins-Valls et al. [13] introduced a novel approach to robot navigation using imitation learning and reinforcement learning, with an emphasis on panoramic images to improve robot training efficiency. Wang et al. [14] proposed a multi-robot navigation DRL method that relies on panoramic images, arguing that panoramic vision provides better perception of the environment and other robots . Artizzu et al. [15] explored the application of deep Q-learning for drone navigation in forests, leveraging 360 degree FOV images as input . In other work, authors explore the use of CNNs for lateral end-to-end control of autonomous cars, employing a single short-range fisheye camera instead of the typical multi-camera setup, with the final model achieving autonomy in urban road conditions [16]. Panoramic vision has also been shown to be useful for social navigation, as it allows for enhanced human perception [17]. Further works applied panoramic vision to simultaneous localization (SLAM) and mapping, such as the work in [18] which presented a framework that relies on panoramic images to build and update topological maps for efficient path planning, outperforming traditional metric-based SLAM. These studies collectively illustrate the diverse applications and methodological advances in the integration of panoramic imaging and DRL in robotics applications. However, translating panoramic images generated by simulated vision sensors to real panoramic vision sensors has not been investigated in the literature. Furthermore, to the best of our knowledge, the problem of learning VS for nonholonomic mobile robots from panoramic images has not been explored.

III. Problem formulation

Consider a scenario in which a nonholonomic mobile robot, equipped with a panoramic camera, operates with the goal of reaching a desired pose denoted as $p^* = (x^*, y^*, \beta^*)$, defined by a panoramic image I^*, relying on the current panoramic image I_t captured at $p_t = (x_t, y_t, \beta_t)$. Here, $m^* = (x^*, y^*)$ and $m_t = (x_t, y_t)$ are the desired and current positions, respectively. β^* and β_t represent the desired orientation and the robot's orientation at instant t. In a DRL context, at each time step t, the robot performs an action a_t and receives the next observation o_{t+1} and a reward r_t. The latter represents the feedback that the agent receives to assess the quality of its action choices. Our objective is to learn a control policy function $\pi(o_t, a_{t-1}, r_{t-1})$ that enables the robot to position itself accurately at the desired pose while following the most direct trajectory to reach it.

As our focus is to study the trajectories performed by the robot, DRL presents a compelling paradigm that enables the robot to learn various trajectories by altering the structure of the reward function, known as reward shaping, in a meaningful manner. We complete the definition of our framework in what follows.

Observation space: At each time step t, the robot receives an observation that consists of a pair of images, i.e., the desired and the current panoramic images: $o_t = \{I_t; I^*\}$.

Action space: We train the robot on the continuous control of linear and angular velocities, denoted respectively as v_x and v_β. At each timestep t, the robot performs an action $a_t = (v_{x_t}, v_{\beta_t})$, expressed in the robot frame.

Reward function: At each timestep t, the robot receives a reward $r_t = r_t^{task} + \eta, r_t^{intr}$, which is the sum of two components: a task reward r_t^{task} related to the problem at hand and an intrinsic reward r_t^{intr} derived from the agent's intrinsic motivation mechanism [19]. Here, η is a weighting coefficient that determines the relative importance of the intrinsic reward compared to the task reward. The formulation and computation of both task and intrinsic rewards are detailed in Section IV-B.

IV. Learning panoramic vision-based control

This section highlights our policy model and the various studied training rewards. The policy model aims at optimizing the DRL loss by maximizing the expected returns.

A. Policy model

Learning control policies from images requires an understanding of the relationship between the pose of the robot and its observations. To effectively address this requirement, a deep recurrent actor-critic network [20] is adopted, allowing efficient modeling of such dependencies (Fig. 2). Firstly, visual domain randomization is applied to input images to account for the variability when transferring the control policy to the real world, we specifically apply a combination of Gaussian noise and color jitter. The resulting images are then passed through a CNN encoder, trained from scratch, to extract visual features. The extracted features from the current and desired images are then concatenated with the

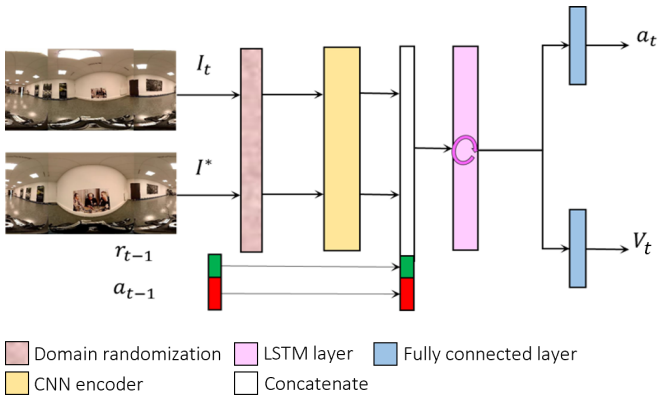

Fig. 2: Illustration of the policy model.

actions and rewards from instant $t-1$ to form a joint representation. This joint representation is then passed through a long short-term memory (LSTM) layer to enable the learning of a temporal representation of the inputs. Finally, the model outputs the robot's control velocities and an estimate of the value function V_t. To train our model, we use Proximal Policy Optimization (PPO) algorithm [21]. This choice stems from the stability of the algorithm throughout the training process, its ability to effectively parallelize multiple training environments, and its robustness to training hyperparameters.

B. Reward shaping

To effectively solve a VS problem, many task reward functions can be tailored to drive the robot to the given desired pose. Since the structure of the task reward determines the nature of the trajectory the robot learns, we study three distinct task rewards, which, intuitively, should lead the robot to learn a velocity control policy enabling it to converge to the desired pose following the optimal trajectory.

The first reward function r_{eucl}^{task} (t is omitted for reading simplicity) defines the Euclidean distance between the current pose of the robot p_t and the desired pose p^*. We express this task reward as a function of the sum of the distance between the current position and the desired position, denoted as d_{m_t}, and the distance between the current orientation and the desired orientation, denoted as d_{β_t}:

$$r_{eucl}^{task} = \lambda_d \; e^{-0.5 d_{m_t}} + \lambda_\beta \; e^{-0.5 d_{\beta_t}}, \tag{1}$$

where the function e^x is used to map the reward values within the range of 0 to 1.

In the second reward function r_{head}^{task}, we include information about the heading orientation of the robot in addition to the Euclidean distance between the current pose and the desired pose:

$$r_{head}^{task} = \lambda_d \; e^{-0.5 d_{m_t}} + \lambda_\beta \; e^{-0.5 d_{\beta_t}} + \lambda_\omega \; e^{-0.5 d_{\omega_t}} \; \mathbb{1}_{d_{m_t}}, \tag{2}$$

where $\omega_t = \arctan\left(\frac{y^* - y_t}{x^* - x_t}\right)$ is the heading angle. d_{ω_t} is defined as the distance between the current orientation β_t and the heading orientation ω_t. The intuition is to enforce the robot to quickly rotate and align with the desired pose, which would bias the robot towards finding the shortest trajectory during training. As the robot gets closer to the desired pose, we assume that this angle is not important, and thus it can be neglected. The operator $\mathbb{1}_{d_{m_t}}$ represents the Heaviside function, the value of which is zero for $d_{m_t} \leq 0.2m$ and one for $d_{m_t} > 0.2m$.

Finally, the third evaluated task reward function r_{proj}^{task} is the sum of two entities: the Euclidean distance between the current position of the robot and the desired position, and the distance to the closest point on the optimal trajectory:

$$r_{proj}^{task} = \lambda_d \; e^{-0.5 d_{m_t}} + \lambda_u \; e^{-0.5 d_{u_t}}. \tag{3}$$

The idea here is to learn the optimal trajectory by defining a distance term that pushes the robot towards the optimal trajectory during the training phase. In this setting, this distance is denoted as $d_{u_t} = \frac{\|(m_t - m_0) \times (m^* - m_t)\|}{\|m^* - m_0\|}$, where m_0 is the initial position of the robot. The weighting of the different reward functions is empirically set to adjust the relative importance of the information of each component.

On the other hand, we use the approximation of the Rényi divergence [22] to compute the intrinsic rewards r_t^{intr}. This choice is justified by the finding that improving exploration enhances the quality of the trajectories. The intrinsic reward is obtained following:

$$r_t^i(o_t) = L(\xi) \left(\frac{\mu_k(e_t, \xi)}{v_k(e_t, \tilde{\xi}) + \epsilon} \right)^{1-\alpha}, \tag{4}$$

where $\xi = \{o_t\}_{t=1}^T$ and $\tilde{\xi} = \{\tilde{o_t}\}_{t=1}^T$ are the sets of observations seen during the current and past episodes, respectively. $e_t = f_\phi(o_t)$ refers to the embedding vector generated by a CNN encoder denoted as f_ϕ. μ_k and v_k denote the Euclidean distance between e_t and the k-th nearest neighbor in the sets ξ and $\tilde{\xi}$, respectively. ϵ is a fixed small constant that aims at avoiding the denominator approaching zero. In this formulation, $L(\xi) = \tanh\left(\frac{1}{T} \sum_{t=0}^{T} \mu_1(e_t, \xi)\right)$ is a scaling coefficient which aims to avoid the robot lingering in a small region and not exploring the environment.

V. Experimental Setup

This section presents detailed descriptions of the training protocol we used and the evaluation setup.

A. Training

To train our simulated mobile robot, we used the Pybullet simulator [23] with a synthetic environment resembling the real-world scene to enable zero-shot transfer of the trained policy to the real robot (Fig. 3). This transfer aims at studying the transferability of panoramic images, generated by projecting a cubemap into an equirectangular image, from simulation to the real-world.

Throughout each training process, the desired pose is fixed at $p^* = (0.0m, 8.0m, 0.0°)$, which is situated in front of a poster. The robot's initial positions are then randomly sampled to ensure that the initial distance error falls within the range of 0.5 to 7.0 meters. Additionally, the robot's initial

(a) (b)

Fig. 3: View of (a) the synthetic scene used for training and (b) the real environments.

orientations are randomly initialized with no bounds. The weighting coefficients for different reward functions (Table I) were empirically chosen to reflect the importance of their corresponding component. All models were trained on two Nvidia A6000 GPUs, requiring approximately 60 hours per training. This cost stems from generating equirectangular images, which involve creating six perspective images. Results are averaged over three random seeds.

Reward function	λ_d	λ_β	λ_ω	λ_u
r^{task}_{eucl}	0.8	0.2	-	-
r^{task}_{head}	0.5	0.2	0.3	-
r^{task}_{proj}	0.5	-	-	0.5

TABLE I: Values of reward functions weighting coefficients.

B. Evaluation

For evaluation, we consider a pre-sampled test set of 100 initial poses. To quantitatively compare the resulting behaviors, we consider four metrics:

- Position error (ΔM): the position error at convergence.
- Orientation error ($\Delta\beta$): the orientation error at convergence.
- Trajectory Length (TL): total distance travelled by the robot during an episode.
- Trajectory Area (TA): area between the trajectory of the robot and the straight line between the initial and desired positions.

The first two metrics quantify the convergence accuracy, whereas the later metrics provide an assessment of the displacement followed with respect to the considered optimal trajectory.

VI. Experimental Results

This section examines the trajectories learned by the robot under the different reward functions, offers a comparative analysis, and shows both simulation and real-world results.

A. Analysis of trajectories based on reward shaping

Fig. 4 Illustrates a representative subset of the trajectories performed by the robot on the test set, trained with our proposed approach, and under the three task reward functions r^{task}_{eucl}, r^{task}_{head}, and r^{task}_{proj}.

Regardless of the employed task rewards, the robot is able to locate the desired image and navigate towards it,

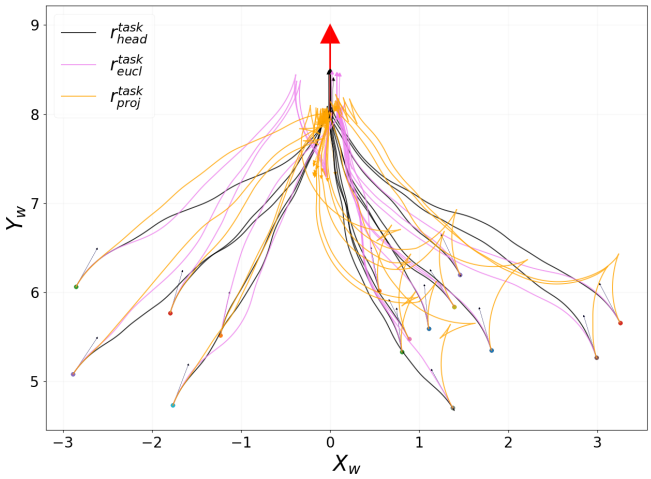

Fig. 4: The trajectories performed by the robot under different task reward functions. The red arrow represents the desired pose.

showcasing varying degrees of precision at convergence. r^{task}_{eucl} enables the robot to minimize the distance to the desired position rapidly. However, in some cases, the robot wanders a bit further than the desired pose, resulting in adaptive behavior by going backwards and regulating itself. On the other hand, the incorporation of the heading angle into r^{task}_{head} allowed the robot to keep track of the optimal trajectory and thus, did not need to adapt in order to regulate itself to the desired pose. The task reward function r^{task}_{proj} resulted in an unexpected behavior where the robot learned to first turn and then drive itself backward to the desired pose. While unexpected, this result shows that it is very difficult to predict in advance what behavior will emerge when solving problems using DRL.

The average of the used metrics computed across the whole test set are shown in Table II. It is noteworthy that using r^{task}_{head} allowed the robot to achieve the lowest pose error while maintaining the best trajectories, yielding centimeter-level and sub-degree errors. r^{task}_{eucl} on the other hand, achieves lower accuracy due to its simplicity and its lack of training signals about the heading of the robot. r^{task}_{proj} which does not carry information about the robot's orientation showcases an interesting behavior where, at convergence, the robot's orientation is approximately 172° with respect to the desired orientation. While the origin of this behavior remains unclear, we postulate that it stems from the randomness of exploration in DRL algorithms. Note that TL and TA are higher for

Task reward	$\Delta M(m)$	$\Delta\beta(^\circ)$	$TL(m)$	$TA(m^2)$
r^{task}_{eucl}	0.0719	1.0714	4.4783	0.4784
r^{task}_{head}	**0.0290**	**0.9339**	**3.1998**	**0.2195**
r^{task}_{proj}	0.1467	171.6009	4.9398	0.7207

TABLE II: Averages of position errors ΔM, orientation errors $\Delta\beta$, Trajectory Lengths TL, and Trajectory Areas TA.

r_{eucl}^{task} and r_{proj}^{task} due to the maneuvers of the robot. In the following sections, we will use r_{head}^{task} to train all DRL models, as it is the reward function with the best performance.

B. Comparison to existing methods

In this section, we compare the following methods:

- IBVS-Pano: Classic image-based visual servoing with panoramic cameras using point features. Using IBVS-Pano for comparaison is meant to highlight the differences between classic VS methods and our DRL-based approaches, and is not intended to a fair comparison.
- DQN-Pano: We adapt the learning-based VS approach proposed by Li et al. [12] to panoramic cameras. This method uses dense correspondence between current and desired images as input to a deep Q-network. It uses a discretized action space composed of seven actions.
- RPPO-Pano w/ intr motivation: Our policy model trained with both task and intrinsic rewards.
- RPPO-Pano w/o intr motivation: Our policy trained solely with task rewards.

Fig. 5 shows the trajectories performed by the compared methods on a set of initial poses with varying degrees of complexity, some of which require completely decoupling the control of angular and linear velocities to reach the desired pose while maintaining an optimal trajectory.

The classical approach IBVS-Pano allows approaching the desired image, but does not accurately converge. In some cases, it diverges and the robot fails in initial poses that require a correction of large angles ($> 90°$). This is due to the classical approach not decoupling the control velocities, which is crucial for precise and stable control, especially for nonholonomic mobile robots. Although decoupling control velocities can be done in IBVS-Pano, it requires complex analytic control approaches to enable such a control strategy.

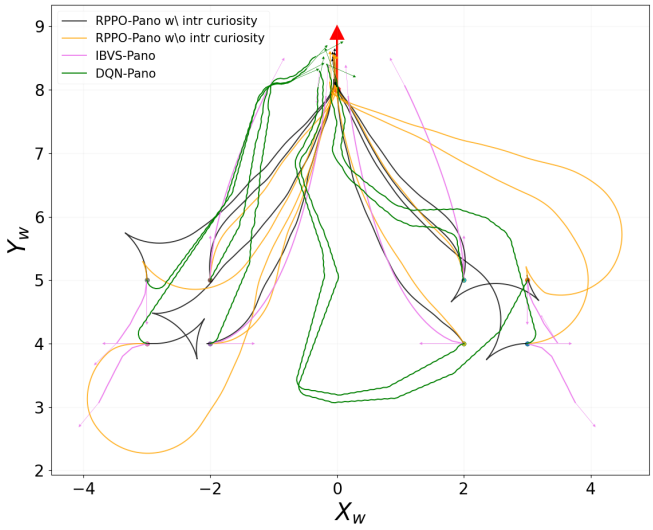

Fig. 5: The trajectories performed by the robot under different methods. The red arrow represents the desired pose.

On the other hand, the learning-based approach DQN-Pano [12] allows for tackling challenging initial poses; however, it is not able to accurately regulate the robot at the desired pose, yielding very winded trajectories. We postulate that this is due to training instability and the discrete action space in DQN. Our approach allows the robot to tackle challenging initial poses and learn to decouple the angular and linear control velocities due to the continuous action space and the robustness of PPO, which allows learning a decoupled control of linear and angular velocities. The impact of intrinsic motivation is also highlighted, as we find that it allows the robot to improve convergence and refine the quality of the trajectories by enhancing the exploration of the environment.

Table III shows the metrics computed across the test set. Our approach outperforms the compared methods by a large margin, maintaining the highest convergence precision and trajectory quality. We note that IBVS-Pano achieves shorter trajectories (TL) due to its lower convergence precision (ΔM).

Method	$\Delta M(m)$	$\Delta\beta(°)$	$TL(m)$	$TA(m^2)$
IBVS-Pano	0.7513	19.4290	**2.8993**	0.9345
DQN-Pano	0.7159	48.9535	3.5896	1.2522
RPPO-Pano w/o intr motivation	0.0514	2.3052	4.0825	0.2419
RPPO-Pano w/ intr motivation	**0.0290**	**0.9339**	3.1998	**0.2195**

TABLE III: Averages of position errors (ΔM), orientation errors ($\Delta\beta$), trajectory lengths (TL), and trajectory areas (TA) for the compared methods.

C. Real-world transfer

We deployed our best-trained model, i.e., RPPO-Pano w/ intr motivation trained with r_{head}^{task}, on a four-wheeled nonholonomic mobile robot (Robotnik Summit-XL) equipped with a dual-hemispherical camera (Insta360 ONE X2) (Fig. 3b). We implemented our approach using the Robot Operating System (ROS)-Noetic middleware that includes the drivers of the SummitXL mobile base and the panoramic camera. We also used openai-ros package, which provides an interface for easily deploying DRL models on real robots. Fig. 6 shows a schematic diagram of these components. The employed camera outputs two hemispherical images, which require preprocessing, including stitching the images and scaling them to fit the input of the neural network. Then, openai-ros connects the DRL model to the robot and

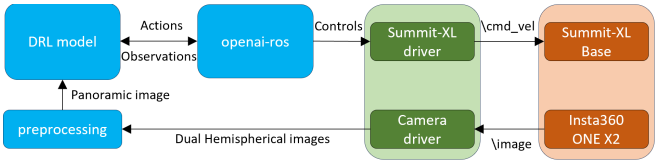

Fig. 6: Illustration of the real robot software stack pipeline.

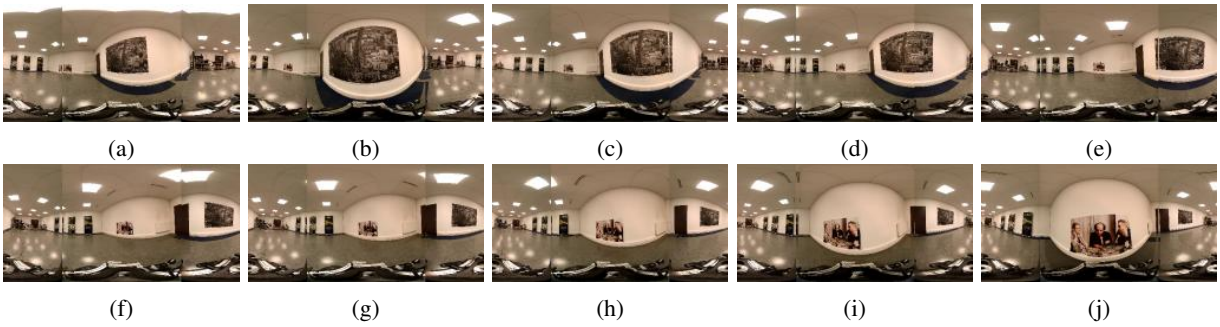

Fig. 7: Experiment conducted on the real robot: (a) The initial image, (b-i) The images acquired during the VS process, and (j) The desired image.

sends control velocities to the robot. The policy was directly deployed on the real robot on, without any fine-tuning on real data. The hardware of the real robot consists of an NVIDIA RTX 2080 GPU, an Intel i7 CPU, and 8GB of RAM.

Fig. 7 presents a sequence of images captured by the real robot. Initially, the robot is positioned approximately 3.5 meters from the desired pose with an initial orientation error of 120°. In this experiment, the robot follows a trajectory similar to the one observed in simulation and exhibits a behavior that is consistent with it, i.e., decoupling control velocities. Furthermore, it demonstrated robustness in scenarios not encountered during training, such as slippery ground or the presence of novel objects in the camera's FOV. We also note that the robot is able to converge towards the desired pose following an almost straight trajectory despite the large discrepancy in visual content between the initial and desired images. However, we observe that the precision at convergence is lower than in simulation, which is attributed to the reality gap.

VII. Conclusion

This paper studies panoramic vision-based control for non-holonomic mobile robots using deep reinforcement learning. We proposed to employ recurrent policies with intrinsic motivation and reward shaping to address this problem. The obtained results demonstrate that deep reinforcement learning is capable of accurately driving the robot to a given desired pose with the emergence of complex control strategies determined by the employed task reward function. Future work will focus on understanding more deeply the reality gap and scaling the training environment.

References

[1] H. Aliakbarpour, O. Tahri, and H. Araujo, "Visual servoing of mobile robots using non-central catadioptric cameras," *Robotics and Autonomous Systems*, 2014.

[2] F. Chaumette and S. Hutchinson, "Visual servo control. i. basic approaches," *IEEE Robotics & Automation Magazine*, 2006.

[3] N. R. Gans and S. A. Hutchinson, "A stable vision-based control scheme for nonholonomic vehicles to keep a landmark in the field of view," in *IEEE Int. Conf. on Robotics and Automation*, 2007.

[4] Y. Masutani, M. Mikawa, N. Maru, and F. Miyazaki, "Visual servoing for non-holonomic mobile robots," in *Proceedings of IEEE Int. Conf. on Intelligent Robots and Systems*, 1994.

[5] Y. Alj and G. Caron, "Featureless omnidirectional vision-based control of non-holonomic mobile robot," in *Int. Conf. on Ubiquitous Robots and Ambient Intelligence*, 2015.

[6] H. H. Abdelkader, Y. Mezouar, N. Andreff, and P. Martinet, "Image-based control of mobile robot with central catadioptric cameras," in *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, 2005.

[7] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine, "How to train your robot with deep reinforcement learning: lessons we have learned," *The International Journal of Robotics Research*, 2021.

[8] N. Crombez, J. Buisson, A. N. André, and G. Caron, "Dual-hemispherical photometric visual servoing," *IEEE Robotics and Automation Letters*, 2024.

[9] S. Asayesh, H. S. Darani, M. Mehrandezh, K. Gupta *et al.*, "Toward scalable visual servoing using deep reinforcement learning and optimal control," *arXiv preprint arXiv:2310.01360*, 2023.

[10] A. Aflakian, A. Rastegarpanah, and R. Stolkin, "Boosting performance of visual servoing using deep reinforcement learning from multiple demonstrations," *IEEE Access*, vol. 11, 2023.

[11] E. G. Ribeiro and V. Grassi Jr, "Real-time deep learning approach to visual servo control and grasp detection for autonomous robotic manipulation," *Robotics and Autonomous Systems*, 2021.

[12] Y. e. a. Li, "Learning view and target invariant visual servoing for navigation," in *IEEE Int. Conf. on Robotics and Automation*, 2020.

[13] D. Watkins-Valls, J. Xu, and N. Waytowich, "Learning your way without map or compass: Panoramic target driven visual navigation," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2020.

[14] H. Wang, W. Wang, X. Zhu, J. Dai, and L. Wang, "Collaborative visual navigation," *arXiv preprint arXiv:2107.01151*, 2021.

[15] C.-O. Artizzu, G. Allibert, and C. Demonceaux, "Drl with omnidirectional images: Application to uav navigation in forests," in *IEEE Int. Conf. on Control, Automation, Robotics and Vision*, 2022.

[16] M. Toromanoff, E. Wirbel, F. Wilhelm, C. Vejarano, X. Perrotton, and F. Moutarde, "End to end vehicle lateral control using a single fisheye camera," in *IEEE Int. Conf. on Intelligent Robots and Systems*, 2018.

[17] S. Liu, P. Chang, W. Liang, and K. Driggs-Campbell, "Decentralized structural-rnn for robot crowd navigation with deep reinforcement learning," in *IEEE Int. Conf on robotics and automation*, 2021.

[18] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[19] A. Aubret and S. Hassas, "A survey on intrinsic motivation in reinforcement learning," *ArXiv*, vol. abs/1908.06976, 2019.

[20] T. Ni, B. Eysenbach, and R. Salakhutdinov, "Recurrent model-free RL can be a strong baseline for many POMDPs," in *Int. Conf. on Machine Learning*. PMLR, 2022.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[22] M. Yuan, B. Li, X. Jin, and W. Zeng, "Rewarding episodic visitation discrepancy for exploration in reinforcement learning," *arXiv preprint arXiv:2209.08842*, 2022.

[23] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2016.