

# OMNI-DRL: Learning to Fly in Forests with Omnidirectional Images

Charles-Olivier Artizzu, Guillaume Allibert, Cédric Demonceaux

▶ **To cite this version:**

Charles-Olivier Artizzu, Guillaume Allibert, Cédric Demonceaux. OMNI-DRL: Learning to Fly in Forests with Omnidirectional Images. Symposium on Robot Control (SYROCO), Oct 2022, Matsumoto, Japan. hal-03777700

**HAL Id: hal-03777700**
**https://hal.science/hal-03777700**

Submitted on 15 Sep 2022

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# OMNI-DRL: Learning to Fly in Forests with Omnidirectional Images

**Charles-Olivier Artizzu** * **Guillaume Allibert** *
**Cédric Demonceaux** **

* *Université Côte d'Azur, CNRS, I3S, France. Emails: artizzu,allibert@i3s.unice.fr*
** *ImViA, Université Bourgogne Franche-Comté, France. Email: cedric.demonceaux@u-bourgogne.fr*

**Abstract:**
Perception is crucial for drone obstacle avoidance in complex, static, and unstructured outdoor environments. However, most navigation solutions based on Deep Reinforcement Learning (DRL) use limited Field-Of-View (FOV) images as input. In this paper, we demonstrate that omnidirectional images improve these methods. Thus, we provide a comparative benchmark of several visual modalities for navigation: ground truth depth, ground truth semantic segmentation, and RGB images. These exhaustive comparisons reveal that it is superior to use an omnidirectional camera to navigate with classical DRL methods. Finally, we show in two different virtual forest environments that adapting the convolution to take into account spherical distortions improves the results even more.

*Keywords:* Omnidirectional sensors, Perception and sensing, Mobile robots and vehicles, Learning robot control, Deep Reinforcement Learning.

## 1 Introduction

In recent years, goal-driven drone navigation in complex, unstructured outdoor environments has been the subject of many studies. Most state-of-the-art solutions (Loquercio et al., 2021; Guastella and Muscato, 2020; Kastner et al., 2021) combine two planners operating at different scales: a global planner to optimize the long-term navigation by creating high-level waypoints and a local planner to navigate from one waypoint to the next while avoiding obstacles.

This local planner is often based on machine learning, directly linking perception and motion commands. Typical methods (Osa et al., 2018) are Behavioral Cloning and Deep Reinforcement Learning (DRL). The first approach attempts to mimic an expert policy learned through supervised learning (Ross et al., 2013; Kim and Chen, 2015; Giusti et al., 2016). This strategy provides reasonable control over the agent's learned policy. However, it suffers from generalization capabilities to scenarios not included in the training data set, including critical failures.

Meanwhile, DRL proposes to learn the navigation policy through trial-and-error experiments (He et al., 2021; Kahn et al., 2021). The agent interacts with the environment based on its perception and state. It receives a reward for promoting or preventing a specific behavior and thus adapts its policy. The DRL method offers excellent generalization capabilities but requires a very long learning process, usually performed in virtual environments to allow thousands of trials and explore failure cases.

Perception is crucial for these DRL algorithms: the agent selects its next action only using its perception. Unfortunately, most state-of-the-art methods rely on capturing depth (He et al., 2021; Loquercio et al., 2021; Zhang et al., 2022), or RGB (Kahn et al., 2021) from sensors with a limited Field-Of-View (FOV). Therefore, we propose to extend these groundbreaking works by breaking the FOV limitation with omnidirectional sensors.

The contributions are threefold:

- First, 360° images are used as input to a DRL-based drone navigation algorithm. This solution is compared to its baseline using only FOV images limited to 90°. To our knowledge, this is the first comparison of omnidirectional and conventional images for DRL. That comparison confirms the advantage of omnidirectional systems.
- Second, a performance benchmark using two FOVs (90° and 360°) and three different visual modalities (ground truth (GT) depth, RGB, GT semantic segmentation) is performed in two virtual forest environments.
- Third, we propose an additional adaptation of the network to take into account the spherical distortions of equirectangular images used in the drone navigation algorithm.

The structure of this paper is the following. Section 2 describes the goal-driven navigation solution adopted here. Next, Section 3 presents the FOV and modality performance benchmark. Finally, Section 4 presents the proposed adaptation for spherical images and the associated results.

## 2 DRL-based navigation solution

Point-goal navigation and collision avoidance for Unmanned Aerial Vehicles (UAVs) are a decision-making problem with uncertainties. This problem can be modeled by a Markov decision process (MDP). In this MDP, the agent interacts with the environment by performing actions following a specific policy in a given state. From this same environment, the agent receives a reward, positive or negative, to promote or prevent certain behaviors. In our specific case, the drone is rewarded positively when it reaches its goal and negatively when it collides with obstacles, gets stuck in a loop, or moves too far away from its objective.

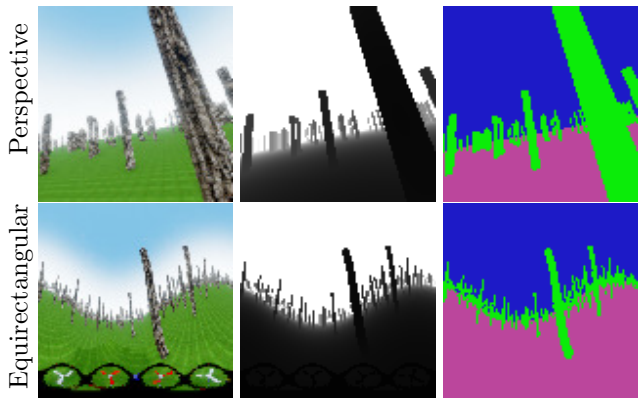



Fig. 1. *RDMAP* environment
(Left: RGB, Middle: depth, Right: segmentation).

### *2.1 State*

In DRL, the agent uses its current state to determine its following action. Therefore, the observation of this state must be accurate and complete enough to provide sufficient information to make an exact decision. But in return, a too exhaustive state will overload the agent with redundant parameters. Thus, in this study, we propose to use a state containing strictly critical information to achieve the two main objectives: point-goal navigation and obstacle avoidance.

For the navigation task, only the relative distance and direction of the goal are provided. At each time-step $t_k = k\Delta t$, where $\Delta t$ is the control sampling time, we define the distance $d_k$ and the angle $\theta_k$ to goal:

$$\begin{aligned} d_k &= \|P_k - P^\star\|_2, \\ \theta_k &= \arctan2\left(y^\star - y_k, x^\star - x_k\right) - \psi_k. \end{aligned} \tag{1}$$

with a drone at position $P_k = (x_k, y_k, z_k)$ heading towards a fixed goal at position $P^\star = (x^\star, y^\star, z^\star)$ with a yaw angle $\psi_k$. At the same time, the drone captures its surroundings with its perception sensor and transforms it into an image noted $I_k$. Finally the state of the drone is defined at each time-step $t_k$ by:

$$S_k = [d_k, \theta_k, I_k]. \tag{2}$$

First, we investigate the impact of the capture FOV by comparing two different set for each modality: a limited one and an omnidirectional one. Second, we propose to compare the pros and cons of several visual modalities for navigation: GT depth, RGB images, and GT semantic segmentation. Fig. 1 presents some observation examples used as input for the DRL solution.

### *2.2 Action*

In our test case, the drone must navigate between tree trunks without collision. Due to the structure of the trees, we propose to realize obstacle avoidance in an iso-altitude plane. Therefore, the drone controller keeps the drone's altitude constant during its flight while the pilot focuses on rotational movements. The agent moves by selecting a discrete action $a_k$ that corresponds to a desired rotation angle among $N_b$ directions $\in [-\pi, \pi]$ :

$$a_k = -\pi + \frac{2i}{N_b - 1}\pi \quad i \in \{0, .., N_b - 1\}. \tag{3}$$

This rotation is then sent to the low-level controller (Shah et al., 2018).

### *2.3 Network and Reward*

In this paper, we focus on the perception of the DRL algorithm using different FOV and modalities as input. Therefore, we first select an actor-critic network and a reward function from published and proven contributions to build our solution. Then, we adapt these bricks to our specific case.

The PPO algorithm (Schulman et al., 2017) processes its state using an actor-critic network to predict the next best action. The architecture proposed in this paper is based on contributions that have already proven effective for UAV navigation (Mnih et al., 2015; He et al., 2021; Loquercio et al., 2021). First, the visual part $I$ of the state $S$ is preprocessed using a succession of convolutions (CONV) and fully connected networks (FC). The resulting 32-dimensional vector is then combined with the goal information $(d_k, \theta_k)$ to determine the next action $a_k$ using another fully connected network (RFC). The global network is shown in Fig. 2 and requires 60k parameters (more details in Appendix A).

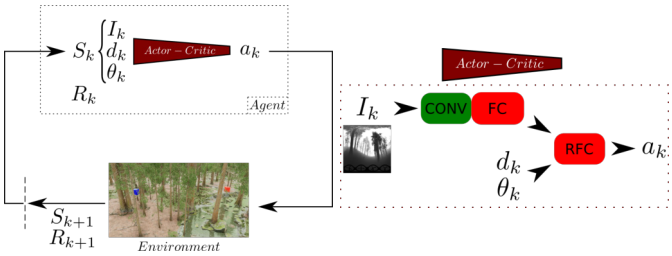



Fig. 2. Left: At each time-step $t_k$ the agent chooses an action $a_k$ based on its state $S_k$ and its policy. This interaction with the environment results in a new state $S_{k+1}$ and a reward $R_{k+1}$ to evaluate the previously followed policy. Right: The visual observation $I_k$ is encoded into a 32-dimensional vector using convolutional operations (CNN) and a fully connected network (FC). Then, combining this output vector and the goal information $(d_k, \theta_k)$, another fully connected network (RFC) predicts the agent's next action $a_k$.

The reward function is crucial in DRL. It describes how the agent achieves its goal. In (Zhang et al., 2022), the authors propose a solution that shows promising results in goal-driven navigation and obstacle avoidance. The reward function is ideally suited for our problem. However, they use depth sensors in their model for flying at a safe distance to the obstacles. In our paper, in order to be independent

to the visual modality used, we have removed this depth part of our final reward function.

At each time-step $t_k$, the reward is computed using the relative goal state $(d_k, \theta_k)$, a penalization term each time-step $(-0.02)$, and an additional reward if the agent reached termination $R_{end}$. The resulting function is :

$$R_k = -0.1d_k - 0.05\|\theta_k\| - 0.02 + R_{end}. \quad (4)$$

There are three different terminal states possible for an episode

- the drone collides with an obstacle, so it receives a large negative reward of $R_{end} = -5$;
- the drone reaches its goal $(d_k < 1m)$, so it receives a great positive reward of $R_{end} = 5$;
- the drone is stuck in a loop (more than 200 time-steps) or going too far away from its goal $(d_k > 100m)$, so it receives a small negative reward of $R_{end} = -2$.

## 3 Results

In this section, we present the performance of our proposed solution with different visual inputs in two different forest environments: a simplified one with only tree trunks and a photorealistic one with complex tree shapes, foliage, rocks, water reflections, and elaborate lighting. In both cases, the agent is trained only in the simplified environment.

### *3.1 Metrics*

To compare the performance of each solution, we used two standard metrics in navigational agent evaluation:

- Success Rate *SR* (the percentage of successful runs divided by the total number of runs) to directly assess the drone's abilities to reach its goal $(d_k < 1m)$;
- Success weighted by path length (Anderson et al., 2018):

$$SPL = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{\ell_i}{p_i}, \quad (5)$$

where $S_i$ is 0 or 1 depending on the success of the episode, $p_i$ is the length of the drone's trajectory and $\ell_i$ is the shortest distance between the initial and goal points. Thus, the closer the drone trajectory is to the shortest path, the closer the SPL is to 100%. At the same time, failed tests are strongly penalized by the boolean value $S_i$.

### *3.2 Simplified evaluation environment*

Unreal Engine (Games, 2020) was chosen as rendering software for its wide range of available scene complexity, from very simplified scenes to photo-realistic environments. Connected to Airsim (Shah et al., 2018), an open-source robotics simulation platform, the simulator can provide high-fidelity modality captures and a low-level controller to stabilize a drone.

With these softwares, we have built a simplified forest environment named *RDMAP*. This $200 \times 200$ meters terrain consists of many vertical cylinders randomly placed schematizing a dense forest of tree trunks. Fig. 3 shows an overview of this simplified environment.

### *3.3 Training*

The proposed DRL solution is trained in *RDMAP* using several visual modalities: GT depth, RGB images, and GT semantic segmentation. In addition, for each modality, two

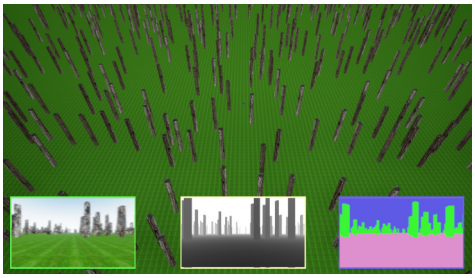


Fig. 3. Overview of the *RDMAP* environment.

different FOVs are tested: perspective images of 90° FOV and equirectangular images (360° FOV). Each training takes 100k time-steps and uses the same schedule and parameters for a fair comparison. We use the standard PPO parameterization as tunning (list of hyper-parameters in Appendix B). During training, the distance between the initial position of the drone and its target is always 20 meters.

The training is performed on Nvidia Tesla-V100 graphic cards and takes about 8 to 10 hours. Inference on 300 drone trajectories takes between 120 and 180 minutes.

### *3.4 Inference in RDMAP*

After training, the resulting policies are tested similarly: point-goal navigation is proven on the same 600 randomly selected drone-goal pairs on the map. However, unlike training, the goal distances are not only $20m$: we also used distances of $40m$ and $60m$ to test our solution and prove its robustness to new situations during training.

The objectives of the inference are twofold. First, we investigate the impact of the observation Field-Of-View (FOV) for each modality. Second, we compare the advantages and disadvantages of all the proposed modalities.

*3.4.1 Ground truth depth* This study compares two different FOVs for the observation used by the DRL algorithm: one limited (90°) and the other omnidirectional (360°).

Depth is the most commonly used observation in drone navigation. So we choose first to investigate the impact of the FOV on this modality. The depth value is cropped to the $[0, 5]$ meter range to remain representative of the capabilities of a small LIDAR mounted on a UAV. The image resolutions are identical for both perspective and equirectangular images ($100 \times 100$ pixels) to keep the same network architecture and compare them fairly. Table 1 shows the comparison between the FOV depth captures of 90° and 360°.

| RUN | *SR* (%) | *SPL* (%) |
|---|---|---|
| 90° FOV GT depth | 76.0 | 54.0 |
| 360° FOV GT depth | **88.8** | **68.6** |

Table 1. Comparing 90° and 360° FOV depth.

As expected, the omnidirectional case presents significantly better performance than the perspective case. The larger FOV optimizes both the navigation and obstacle avoidance tasks. It allows the detection of more objects, usually unseen in the perspective FOV (larger SR). Furthermore, it optimizes the UAV trajectory according to its whole surroundings (larger SPL).

*3.4.2 RGB and ground truth semantic segmentation*
The previous paragraph was limited to the study of ground truth depth, which is the usual observation for UAV navigation. Here, we propose to compare the performance of depth against two other visual modalities: RGB images and ground truth semantic segmentation (three classes of objects are present in RDMAP: ground, trees, and sky). As above, each visual modality was tested with two different FOVs. Table 2 shows the inference results.

| RUN | *SR* (%) | *SPL* (%) |
|---|---|---|
| 90° FOV RGB | 68.3 | 52.8 |
| 360° FOV RGB | **85.7** | **62.6** |
| 90° FOV GT SEG | 72.7 | 52.4 |
| 360° FOV GT SEG | **88.7** | **60.0** |

Table 2. Comparing 90° and 360° modalities.

First, FOV has the same impact on all modalities tested: the omnidirectional model always performs better than the limited model. The RGB solution shows a really great improvement in success rate using omnidirectional images.

Second, the ground truth depth is the best performing modality overall (best SR and SPL). This comforts the fact that depth is the most crucial modality to detect and avoid obstacles and is the commonly used modality in navigation. Nevertheless, RGB based DRL-solution shows very promising performances especially when using 360° images. This particular case shows a great improvement in SR and SPL against the 90° FOV ground truth depth.

*3.5 Inference in RWFOREST*

To reduce the gap with reality, we build *RWFOREST* using the best rendering capabilities of Unreal Engine and forest textures from its marketplace. Fig. 4 shows some observations of the *RWFOREST* environment. This photo-realistic environment features complex lighting and trees of different diameters with branches and dense foliage instead of the simple cylinders used in *RDMAP*. As a result, *RWFOREST* is a significantly more challenging environment with much more complex observations to analyze and translate into action.

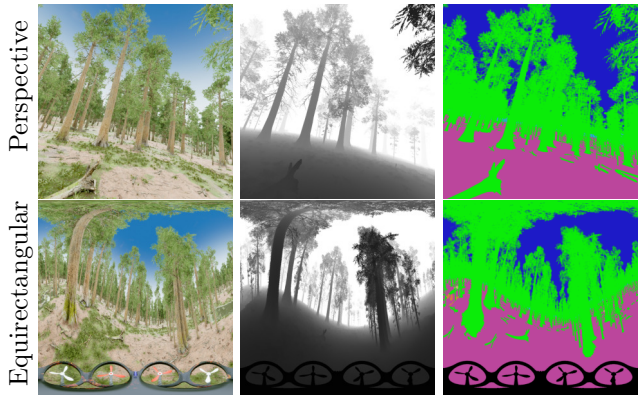



Fig. 4. *RWFOREST* environment
(Left: RGB, Middle: depth, Right: segmentation).

No additional training was performed in this new environment to test the robustness of our proposed solution. Instead, the pre-trained models of the section 3.3 were directly tested on 600 objectives of $20m$, $40m$, and $60m$. The Table 3 presents the result of these inferences.

Although tested on significantly more complex observations than those observed in training, our proposed solution still performs well in terms of success rate: over 80% SR when using 360° FOV ground truth depth images or RGB images. We also find that similar to the findings in Section 3.4, 360° FOV images perform better than 90° FOV images. Even though the information in the equirectangular images is denser than in the perspective images (they both have the same resolution here), the model can still detect more obstacles and better optimizes its trajectory with a wider FOV.

| RUN | *SR* (%) | *SPL* (%) |
|---|---|---|
| 90° FOV GT depth | 81.0 | 69.1 |
| 360° FOV GT depth | **82.2** | **75.7** |
| 90° FOV RGB | 71.7 | 57.8 |
| 360° FOV RGB | **80.2** | **58.9** |

Table 3. Models trained in *RDMAP* environment and tested in *RWFOREST*.

For 360° FOV images, the performances of ground truth depth and RGB images are very close in terms of success rate. But depth outperforms RGB in SPL. Therefore, both modalities can detect the same obstacles, but depth gives additional information allowing faster (better optimized) trajectories to fly to the same goal.

## 4 Spherical Adaptation

The proposed DRL-based navigation solution shows promising results using 360° FOV observations as inputs. However, all spherical projections come with distortions. In particular, equirectangular images show significant distortions near the polar regions. Therefore, these distortions must be considered when using these images to maintain local consistency between the convolved pixels.

*4.1 Distortion-aware convolution*

Several methods deal with spherical distortions in convolutional neural networks (CNNs). A first approach consists in modifying the entire feature maps using Fourier transforms (Cohen et al., 2018) or spherical polyhedra (Lee et al., 2019). A second technique directly changes the shape of the convolutional kernels. The new kernels shapes can either be learned (Su and Grauman, 2019) or specifically adapted to the equirectangular projection (Fernandez-Labrador et al., 2020; Artizzu et al., 2021). This latter method does not require new variables or additional learning, nor a complete modification of the network architecture. For these reasons, we choose to implement this distortion-aware convolution using a pre-computed offsets table.

The four convolutional layers of the proposed feature extraction architecture are modified. Prior to training, the

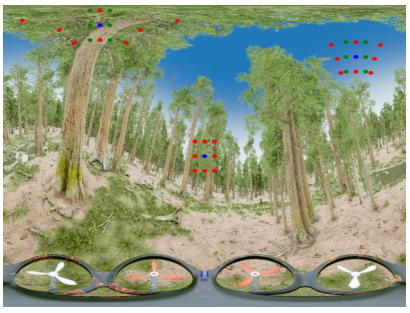


Fig. 5. Kernels with different latitude and longitude. (Blue: the kernel center, Green: the perspective kernel, Red: the adapted equirectangular kernel).

offsets tables are computed based on the resolution of the observation used and the different parameters of each adapted convolutional layer. Fig. 2 shows the additional plugin applied on the CNN part of the network.

Then, during training and inference, these offsets are applied to change the position of each kernel to project the spherical image locally onto its perspective equivalent. For example, Fig. 5 shows different distortion-aware kernel shapes in function of their position in the equirectangular image.

The proposed navigation solution is trained in the *RDMAP* environment in a process similar to that presented in Section 3.3.

### *4.2 Results in RDMAP*

The equirectangular adapted DRL solution is tested in the *RDMAP* environment and compared to its baseline from Section 3.4. Table 4 shows the performances of 360° FOV navigation based on ground truth depth or RGB. The baseline results are directly reused from Table 2.

| RUN | *SR* (%) | *SPL* (%) |
|---|---|---|
| 360° FOV GT depth (baseline) | 88.8 | 68.6 |
| 360° FOV GT depth (DRL adapted) | **94.8** | **78.3** |
| 360° FOV RGB (baseline) | 84.5 | 62.6 |
| 360° FOV RGB (DRL adapted) | **85.7** | **67.4** |

Table 4. Performances in *RDMAP*.

The baseline and distortion-aware (adapted) solutions show slightly similar performances in terms of success rate, but the SPL is greatly improved by the distortion-aware convolutions. Thus, maintaining a better local pixel coherence during convolutions helps the proposed navigation solution to optimize the drone trajectory between obstacles.

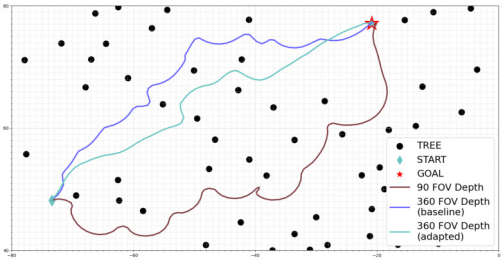



Fig. 6. Drone trajectory between its initial position and goal for three different DRL-solutions in the *RDMAP* environment: the 90° FOV GT depth, the 360° FOV GT depth (baseline) and the 360° FOV GT depth adapted solution.

A sample trajectory of the test set is provided in Fig. 6 to illustrate this difference in trajectory optimization. The drone must avoid numerous obstacles before reaching its goal, located at 60 meters. Three different navigation solutions are compared: 90° FOV ground truth depth, 360° FOV GT depth baseline and 360° FOV GT depth adapted. The latter solution has the shortest and most optimized trajectory. On the other hand, the solution using 90° FOV images reaches the goal but makes a considerable detour.

### *4.3 Results in RWFOREST*

The same comparison is also performed between the distortion-aware solution and its baseline in the *RWFOREST* environment. Only ground truth depth is tested here as it showed the best performances in the last section. Table 5 shows the differences between the adapted 360° FOV solution and its baseline solution from Section 3.5.

| RUN | *SR* (%) | *SPL* (%) |
|---|---|---|
| 360° FOV GT depth (baseline) | 82.2 | 75.7 |
| 360° FOV GT depth (DRL adapted) | **89.7** | **76.9** |

Table 5. Performances in *RWFOREST*.

As in the previous section, the performance of the adapted solution is better than the baseline. However, the trends are different than those in Table 4. When tested in the training environment, the distortion-aware and baseline solutions detect the same obstacles (same level of SR in Table 4). On the contrary, the adapted solution detects the obstacles much better when the test takes place in a new and more complex environment (much higher SR in Table 5). On the other hand, the trajectories are relatively identical in length (close SPL), so the trajectory optimization is similar (contrary to the previous case).

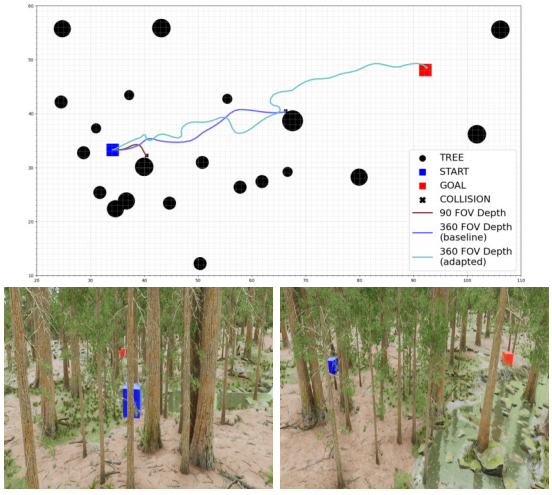



Fig. 7. Drone trajectory between its initial position (blue square) and goal (red square) for different DRL-solutions in the *RWFOREST* environment: 90° FOV GT depth, 360° FOV GT depth (baseline) and 360° FOV GT depth (DRL adapted). Front and lateral views in *RWFOREST* are provided below: the squares are not used in simulation only visualisation.

An example trajectory of the test set is provided in Fig. 7. The front and lateral views of the environment reveal the scene's complexity with several trees of different shapes and foliage. Only the distortion-aware solution adapted for the 360° ground truth depth reaches the goal, showing this solution's better overall performance than the perspective solution and the 360° FOV baseline.

## 5 CONCLUSION

Thanks to its wide FOV, an omnidirectional sensor allows a drone to capture its entire surroundings. In (Rituerto et al., 2010), the authors demonstrated that this type of sensor significantly improves SLAM compared to a conventional perspective camera. Similarly, our paper has shown that it also greatly enhances DRL-based navigation: more obstacles detection results in higher success rates and faster trajectories towards a fixed goal. We have also confirmed that depth is the best visual modality to navigate without collision. Finally, we have proposed an adaptation of the convolution layers to take into account

the spherical distortions. It significantly improves the performance of our proposed solution even when tested in a new environment.

Our code implementation and *RDMAP* environment will be open-source and available on GitHub at https://github.com/COATZ/OMNI-DRL.

Acknowledgements

This work was supported by the ANR CLARA project, grant ANR-18-CE33-0004 of the French Agence Nationale de la Recherche. This work was granted access to the HPC resources of IDRIS under the allocation AD011013128 made by GENCI.

References

Anderson, P., Chang, A., Chaplot, D.S., Dosovitskiy, A., Gupta, S., Koltun, V., Kosecka, J., Malik, J., Mottaghi, R., Savva, M., and Zamir, A.R. (2018). On evaluation of embodied navigation agents. *arXiv*, abs/1807.06757.

Artizzu, C.O., Zhang, H., Allibert, G., and Demonceaux, C. (2021). Omniflownet: a perspective neural network adaptation for optical flow estimation in omnidirectional images. In *2020 25th International Conference on Pattern Recognition (ICPR)*, 2657–2662. IEEE.

Cohen, T.S., Geiger, M., Koehler, J., and Welling, M. (2018). Spherical cnns. *2018 International Conference on Learning Representations (ICLR)*, abs/1801.10130.

Fernandez-Labrador, C., Facil, J.M., Perez-Yus, A., Demonceaux, C., Civera, J., and Guerrero, J.J. (2020). Corners for layout: End-to-end layout recovery from 360 images. *IEEE Robotics and Automation Letters*, 5, 1255–1262.

Games, E. (2020). Unreal engine. *Epic Games*.

Giusti, A., Guzzi, J., Ciresan, D.C., He, F.L., Rodriguez, J.P., Fontana, F., Faessler, M., Forster, C., Schmidhuber, J., Caro, G.D., Scaramuzza, D., and Gambardella, L.M. (2016). A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters*, 1, 661–667.

Guastella, D.C. and Muscato, G. (2020). Learning-based methods of perception and navigation for ground vehicles in unstructured environments. *Sensors*, 21, 73.

He, L., Aouf, N., and Song, B. (2021). Explainable deep reinforcement learning for uav autonomous path planning. *Aerospace Science and Technology*, 118, 107052.

Kahn, G., Abbeel, P., and Levine, S. (2021). BADGR: An autonomous self-supervised learning-based navigation system. *Robotics and Automation Letters*, 6, 1312–1319.

Kastner, L., Zhao, X., Buiyan, T., Li, J., Shen, Z., Lambrecht, J., and Marx, C. (2021). Connecting deep-reinforcement-learning-based obstacle avoidance with conventional global planners using waypoint generators. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1213–1220. IEEE.

Kim, D.K. and Chen, T. (2015). Deep neural network for real-time autonomous indoor navigation. *arXiv*, abs/1511.04668.

Lee, Y., Jeong, J., Yun, J., Cho, W., and Yoon, K.J. (2019). Spherephd: Applying cnns on a spherical polyhedron representation of 360° images. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2019-June, 9173–9181. IEEE.

Loquercio, A., Kaufmann, E., Ranftl, R., Müller, M., Koltun, V., and Scaramuzza, D. (2021). Learning high-speed flight in the wild. *Science Robotics*, 6.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., and Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, 518, 529–533.

Osa, T., Pajarinen, J., Neumann, G., Bagnell, J.A., Abbeel, P., and Peters, J. (2018). An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7, 1–179.

Rituerto, A., Puig, L., and Guerrero, J. (2010). Comparison of omnidirectional and conventional monocular systems for visual slam. *10th OMNIVIS with Robotics: Science and Systems*.

Ross, S., Melik-Barkhudarov, N., Shankar, K.S., Wendel, A., Dey, D., Bagnell, J.A., and Hebert, M. (2013). Learning monocular reactive uav control in cluttered natural environments. In *2013 IEEE International Conference on Robotics and Automation*, 1765–1772.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv*, abs/1707.06347.

Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2018). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. *arXiv*, 621–635.

Su, Y.C. and Grauman, K. (2019). Kernel transformer networks for compact spherical convolution. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 9434–9443.

Zhang, S., Li, Y., and Dong, Q. (2022). Autonomous navigation of uav in multi-obstacle environments based on a deep reinforcement learning approach. *Applied Soft Computing*, 115, 108194.

## A Features extractor architectures

| LAYER | TYPE | NB_PARAMS |
|---|---|---|
| Visual capture ($I_k$) | | |
| C1 | CONV2D(8, K=3, S=2, P=1) | 80 |
| C2 | CONV2D(8, K=3, S=2, P=1) | 584 |
| C3 | CONV2D(8, K=3, S=2, P=1) | 584 |
| C4 | CONV2D(8, K=3, S=2, P=1) | 584 |
| FC1 | FC(8*7*7, 64) | 25152 |
| FC2 | FC(64, 32) | 2080 |
| + Goal state ($d_k, \theta_k$) | | |
| RFC1 | FC(34,64) | 2172 |
| RFC2 | FC(64,128) | 8320 |
| RFC3 | FC(128,128) | 16512 |
| RFC4 | FC(128,37) | 4773 |

Table A.1. CustomCNN

Total: 60814 parameters.

## B PPO hyperparameters

| Hyperparameter | Value |
|---|---|
| learning rate | 0.0003 |
| number of steps | 2048 |
| batch size | 64 |
| number of epochs | 10 |
| gamma | 0.99 |
| $\Delta_t$ | 200 ms |
| $N_b$ | 37 actions |

Table B.1. Simulation Hyperparameters