



HAL
open science

Adaptive deep learning models for omnidirectional images : from perception to navigation

Charles-Olivier Artizzu

► **To cite this version:**

Charles-Olivier Artizzu. Adaptive deep learning models for omnidirectional images : from perception to navigation. Computer Vision and Pattern Recognition [cs.CV]. Université Côte d'Azur, 2023. English. NNT : 2023COAZ4039 . tel-04260673

HAL Id: tel-04260673

<https://theses.hal.science/tel-04260673>

Submitted on 26 Oct 2023

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.

THÈSE DE DOCTORAT

Modèles d'apprentissage profond adaptés aux images omnidirectionnelles : de la perception à la navigation

Charles-Olivier Artizzu

Laboratoire I3S, CNRS

Présentée en vue de l'obtention du grade de docteur en :
Automatique Traitement du Signal et des Images d'Université Côte d'Azur.

Dirigée par :

Guillaume Allibert, Maître de conférences HDR, Université Côte d'Azur, co-Directeur de thèse,
Cédric Demonceaux, Professeur des universités, Université de Bourgogne Franche-Comté,
co-Directeur de thèse.

Soutenue le : 14 Juin 2023.

Devant le jury, composé de :

Marie Babel, Professeure des universités, INSA Rennes, Examinatrice,
Nicolas Marchand, Directeur de recherche, CNRS, Rapporteur,
Anne Spalanzani, Professeure des universités, Université Grenoble Alpes, Examinatrice,
Pascal Vasseur, Professeur des universités, Université de Picardie Jules Verne, Rapporteur.

Le ciel est, par-dessus le toit,
Si bleu, si calme !
Un arbre, par-dessus le toit,
Berce sa palme.

La cloche, dans le ciel qu'on voit,
Doucement tinte.
Un oiseau sur l'arbre qu'on voit
Chante sa plainte.

Mon Dieu, mon Dieu, la vie est là,
Simple et tranquille.
Cette paisible rumeur-là
Vient de la ville.

Qu'as-tu fait, ô toi que voilà
Pleurant sans cesse,
Dis, qu'as-tu fait, toi que voilà,
De ta jeunesse ?

Paul Verlaine, *Sagesse* (1881).

Remerciements

Je saisis cette opportunité pour exprimer ma profonde reconnaissance envers les membres éminents de mon jury de thèse. Votre présence lors de la soutenance a témoigné de l'estime que vous portez à cette recherche, et je suis sincèrement honoré d'avoir pu bénéficier de vos perspectives éclairantes et des dialogues stimulants que nous avons partagés.

Un hommage tout particulier va à mes rapporteurs de thèse, Monsieur Nicolas Marchand et Monsieur Pascal Vasseur, pour avoir généreusement consacré leur temps à étudier minutieusement mon manuscrit et à élaborer des rapports détaillés. Mes remerciements vont également aux examinatrices, Madame Marie Babel et Madame Anne Spalanzani, pour leur évaluation attentive du manuscrit et de ma soutenance.

Je tiens à adresser mes plus sincères remerciements et une profonde gratitude à mes directeurs de thèse, Monsieur Guillaume Allibert et Monsieur Cédric Demonceaux. Leurs orientations avisées ont illuminé mon chemin de recherche et ont joué un rôle essentiel dans la réalisation de ce travail. Je suis profondément reconnaissant envers eux pour leur patience, leur expertise inestimable et surtout leur engagement indéfectible. Leur soutien sans faille a été la pierre angulaire de mon parcours jusqu'à la réussite de cette thèse.

Mes remerciements vont également à tous ceux qui ont partagé sans réserve leurs idées et leurs connaissances au cours de cette aventure intellectuelle. Je souhaite exprimer ma reconnaissance à Zongwei, Renato, Sardor, Rida et Théo pour leur contribution précieuse.

Enfin, mes pensées vont vers ma famille, Pauline, ainsi que vers mes amis proches, pour leur soutien inébranlable et leurs encouragements constants tout au long de ce périple académique, en particulier durant les périodes qui n'ont pas toujours été dépourvues de défis. Votre présence et votre confiance ont été des sources inestimables de motivation. Merci du fond du cœur.

Abstract

Omnidirectional¹ cameras for computer vision and robotics are becoming increasingly widespread. Indeed, thanks to their 360° field of view, they allow a global perception of each observed scene in a single shot. Moreover, the latest generation of spherical cameras is more accurate, lighter, and less expensive, encouraging their use in many mobile applications.

However, these images present significant distortions due to the spherical projection, such as in the polar regions of equirectangular images. As a result, conventional image processing methods often cannot recognize objects or understand what is happening in these areas. Thus, several methods have been proposed to overcome these distortions and often use a supervised learning on omnidirectional datasets. However, these spherical datasets are rare and usually limited to very specific use cases. In contrast, the perspective domain offers greater diversity and versatility. Therefore, in this thesis, we propose transferring perspective-based methods to omnidirectional content without additional training. Our simple and fast adaptation solution relies on distortion-aware convolutions using a local perspective projection on the sphere.

To prove the relevance and generalization of our method to any convolutional network, we apply it to three commonly used computer vision tasks: semantic segmentation, depth, and optical flow estimation. When tested on specially created datasets and real scenarios, the spherically adapted networks always perform better than the baseline version.

Following these results in computer vision, we focused on their use in robotics, particularly for drone navigation in complex, dense, and unstructured environments such as forests. Perception is crucial for image-based navigation, especially for obstacle avoidance. However, most current algorithms use images with a limited field of view.

Therefore, this thesis proposes a solution using omnidirectional images and compares it to its perspective reference. For all scenarios and visual modalities considered, our equirectangular image-based navigation solution is safer and faster than its perspective counterpart, even in a much more complex environment than the one observed during training. In addition, using distortion-aware convolutions in the navigation algorithm also improves flight performance.

Keywords:

Deep Learning, Omnidirectional Images, Perception, Drone Navigation.

¹In this thesis, we interchangeably use the terms omnidirectional image, 360° FOV image, spherical image, and equirectangular image.

Résumé

Les caméras omnidirectionnelles² sont de plus en plus répandues en vision par ordinateur et robotique. En effet, grâce à leur champ de vision à 360 degrés, elles permettent d’acquérir en une seule prise de vue une scène complète. De plus, la diminution du cout et du poids des dernières caméras sphériques facilite leur intégration dans de nombreuses applications mobiles.

Cependant, ces images présentent toujours des distorsions importantes en raison de la projection sphérique, comme dans les régions polaires des images équirectangulaires. Par conséquent, les approches traditionnelles de traitement d’image sont souvent incapables de reconnaître les formes des objets, entraînant ainsi une mauvaise compréhension de la scène observée. Plusieurs méthodes ont donc été proposées pour prendre en compte ces distorsions et nécessitent le plus souvent un apprentissage supervisé basé sur des images omnidirectionnelles. Cependant, les ensembles de données sphériques sont rares et généralement limités à des applications très spécifiques. À l’inverse, ceux regroupant des images perspectives offrent une grande diversité et polyvalence.

Dans cette thèse, nous proposons d’adapter des modèles entraînés avec des images perspectives et de les appliquer directement sur des données omnidirectionnelles en évitant tout apprentissage supplémentaire. Notre adaptation repose sur la prise en compte des distorsions sphériques lors des opérations de convolution par le biais de projections perspectives locales sur la sphère. La solution proposée est facile et rapide d’utilisation.

Afin de démontrer la pertinence et la généralisation de notre méthode à tout réseau convolutionnel, nous l’appliquons à trois tâches de vision par ordinateur couramment utilisées : l’estimation de la segmentation sémantique, de la profondeur et du flot optique. Testée à la fois avec des données virtuelles et des scénarios réels, les réseaux adaptés sont toujours plus performants que ceux de référence.

À la suite de ces résultats en vision par ordinateur, nous nous sommes penchés sur leur utilisation en robotique. La perception est une fonction cruciale de la chaîne de navigation, notamment pour éviter les obstacles dans le cadre de la navigation de drones dans des environnements complexes, denses et non structurés tels que les forêts. Cependant, la plupart des algorithmes actuels sont limités par un champ de vision perspectif.

Nous proposons donc ici une solution utilisant un champ de vision omnidirectionnel et la comparons à celle basée sur des images perspectives. Dans

²Dans cette thèse, nous utilisons indifféremment les termes d’image omnidirectionnelle, d’image à 360°, d’image sphérique et d’image équirectangulaire.

tous les scénarios testés, les images équirectangulaires permettent une navigation plus rapide et sûre, y compris dans des situations plus complexes que celles rencontrées lors de l'apprentissage. En outre, la prise en compte des distorsions telle que proposée en première partie améliore également les performances de vol.

Mots-clés:

Apprentissage profond, Images omnidirectionnelles, Perception, Navigation de drones.

Contents

1	Introduction	23
1.1	Motivations and Challenges	23
1.2	Outline	25
1.3	Contributions	26
1.3.1	Peer-Reviewed Publications	26
1.3.2	Open-Source Datasets and Softwares	26
I	Generalization of the Omnidirectional Distortion-Aware Convolutions	29
2	Omnidirectional Computer Vision	30
2.1	Physical Devices	31
2.1.1	Rotating Cameras	31
2.1.2	Catadioptric Cameras	32
2.1.3	Fisheyes Cameras	32
2.1.4	Polydioptric Cameras	33
2.2	Data-Driven Computer Vision	34
2.2.1	Deep Learning	34
2.2.2	Common Computer Vision Tasks	38
2.3	Omnidirectional Image Processing	44
2.3.1	Training on Omnidirectional Datasets	45
2.3.2	Spherical Latent Space	46
2.3.3	Multi-Projection Fusion	46
2.3.4	Deformable Convolutions	47
2.4	Proposed Spherical Adaptation Solution	48
2.4.1	Local Perspective Projection on the Sphere	48
2.4.2	Implementation in any Convolutional Network	50
3	Evaluation on Omnidirectional Images	52
3.1	Evaluation Datasets	52
3.1.1	Semantic Segmentation and Depth Evaluation	53
3.1.2	Optical Flow Evaluation	53
3.1.3	Additional Real-World Test Scenarios	56
3.2	Adapted and Baseline Models Comparison	57
3.2.1	Semantic Segmentation Comparison	57
3.2.2	Monocular Depth Comparison	61

3.2.3	Optical Flow Comparison	65
3.2.4	Computation Time Comparison	73
Conclusion of Part I		74
II Deep Reinforcement Learning Navigation using Omnidirectional Images		77
4	Image-Based Navigation	78
4.1	Image-Based Navigation Strategies	79
4.1.1	Map-Based	79
4.1.2	Learning-Based	80
4.1.3	Combining Model-Based and Learning-Based	82
4.1.4	Selected Navigation Method	83
4.2	Reinforcement Learning	84
4.2.1	Markov Decision Process	84
4.2.2	Discounted Expected Reward	85
4.2.3	Algorithmic diversity	85
4.2.4	Optimal Policy Search	86
4.2.5	Dynamic Programming	87
4.2.6	Monte-Carlo	88
4.2.7	Temporal Differences	89
4.3	Deep Reinforcement Algorithms	90
4.3.1	Value-Based Approaches	90
4.3.2	Policy Gradient Methods	90
4.3.3	Actor-Critic Strategies	91
5	Navigation Framework	92
5.1	Flight Environment	92
5.1.1	RDMAP: Simplified Training and Testing Environment	92
5.1.2	RDFOREST: Photorealistic Testing Environment	93
5.2	Proposed Framework	94
5.2.1	Action Space	94
5.2.2	Drone State	95
5.2.3	Visual Modalities used as Input	95
5.2.4	Actor-Critic Network	96
5.2.5	Reward	98
6	Navigation Evaluation	99
6.1	Metrics	99
6.2	Training Schedule and Hyperparameters	100
6.3	Reinforcement Learning Solver selection	100
6.3.1	Test with No Obstacle	101
6.3.2	Test with Obstacles	101
6.4	Omnidirectional versus Perspective Navigation	102
6.5	Distortion-Aware Convolutions for DRL	105

6.5.1	Actor-Critic Network Adaptation	106
6.5.2	MIDAS Network Adaptation	107
6.6	Generalization to a Photorealistic Forest	110
Conclusion of Part II		112
7	Conclusion	113
7.1	Summary of Contributions	113
7.2	Perspectives	114
7.2.1	Distortion-Aware Transformers	114
7.2.2	Solve the Periodicity Issue	115
7.2.3	Omnidirectional Image generation	115

List of Figures

1.1	Equirectangular image of an outdoor urban driving scene. The polar regions are strongly distorted, so much so that it is difficult to recognize the car in the lower part of the image.	24
2.1	Pan Tilt Zoom cameras. Left: Axis. Right: Honey Optics.	31
2.2	A catadioptric camera and a captured spherical image [11].	32
2.3	A fisheye camera and an image captured during urban driving [12].	32
2.4	The Ricoh Theta Z1 camera combines two fisheye lenses to reconstruct equirectangular images.	33
2.5	Polydioptric camera systems using circular rigs (GoPro Odyssey on the left) or spherical configurations (Panono 360 on the right).	33
2.6	Artificial Neuron.	34
2.7	Artificial neural network with a single hidden layer. More precisely, it is a fully connected architecture because each neuron is connected to all neurons of the previous layer.	36
2.8	Convolution operation of a 3×3 filter applied on a 7×7 input. .	37
2.9	The Vision Transformer architecture [24]. The image is split into fixed-size patches with position embeddings and fed into a Transformer encoder. The output is a class estimation.	38
2.10	An example of semantic segmentation [30].	39
2.11	Estimating depth from a single image is an ill-posed problem [47] since several real points project onto the same pixel. Using a second camera alleviates this ambiguity by matching points (drawing inspired by [48]).	40
2.12	Optical flow between two frames [61]. (Top left): RGB input at t , (top right): RGB input at $t + 1$, (bottom left): ground truth optical flow with reference color wheel, (bottom right): Flow vectors between pixels.	42
2.13	Equirectangular projection of the Earth’s globe. The Tissot’s indicatrix illustrate the amplitude of distortions [76]: a circle whose shape is regular near the equator is significantly distorted near the poles.	44
2.14	Stanford2d3d dataset [81].	45
2.15	The visualization of how the kernel is applied to polyhedra representation [90]. Yellow kernel shows the case when the kernel is located at the vertex of the icosahedron. Blue kernel shows the case when the kernel is located at the pole.	46

2.16	UniFuse network: [95]. The model extracts features from equirectangular and cubemap images and merges the contribution to estimate the depth in the observed scene.	46
2.17	Left: Illustration of 3 deformable convolutions [99]. Offsets are learned during training to adapt to various transformations for scale, aspect ratio, and rotation. Right: Each image shows the sampling locations of deformable filters for activation units on a small object and a large one.	47
2.18	The equirectangular image presents significant distortions in the polar regions. Convolution kernel shapes are modified according to their latitude. Blue : kernel center, Green : perspective kernel, Red : adapted equirectangular kernel.	49
2.19	One significant advantage of our proposed method is its ease of implementation in any existing convolutional network pre-trained with perspective images. Moreover, no additional training on omnidirectional datasets is required. At test time, the weights are directly transferred to the same architecture with distortion-aware convolutional filters to process equirectangular images. We compute these spherical offsets offline to avoid computational slowdowns . Although this figure illustrates the case of the semantic segmentation task, we apply the same strategy for monocular depth and optical flow estimation.	51
3.1	RWFOREST dataset. Here is the 256×256 resolution version.	53
3.2	3D models used to generate the Forest dataset.	54
3.3	OmniFlowNet [6], Flow360 [110], CityScene and EquirectFlyingThings [98] datasets.	55
3.4	Images captured with an omnidirectional camera during different moving scenes.	56
3.5	Semantic segmentation architecture used in this thesis proposed in [46] (encoder: ResNet50 [21], decoder: Pyramid Pooling Module [39]).	58
3.6	Prediction examples in the RWFOREST dataset. The spherical adaptation improves shape detection (tree canopy is better identified) and reduces erroneous class estimation. (Top left): RGB input, (top right): ground truth segmentation, (bottom left): prediction from the baseline network, (bottom right): prediction from the adapted network.	59
3.7	Urban driving example (Car2). The adapted network better identifies the tree canopy. A red circle at the top left of the image highlights the area with the most visible differences: the baseline network estimates the earth (in brown) class instead of trees (in green).	60
3.8	MIDAS lightest network [60].	61

3.9	Prediction examples in the RWFOREST dataset. The predicted depth images are visually challenging to compare. However, quantitative measurements have shown that the adapted version is numerically better than the baseline. (Top left) : RGB input, (top right) : ground truth monocular depth, (bottom left) : prediction from the baseline network, (bottom right) : prediction from the adapted network.	63
3.10	Urban driving examples. Red circles at the top of the image highlight the areas with the most visible differences. Sample 1 : The adapted network better estimates depth in the polar regions of the equirectangular images. Sample 2 : Less erroneous depth estimation from the adapted network.	64
3.11	GMFlow network [28]. The RAFT [72] convolutional encoder is used to preprocess the image input before the Transformer.	65
3.12	Ball1 and Ball2 cases. (Top) RGB input image , (bottom left) optical flow estimation from the baseline network, (bottom right) estimation from the adapted network. The adapted network better estimates the optical flow in the top polar region is better. Sample 1 : The ball is clearly more visible and smoother in that case. Sample 2 : The arm is smoother.	67
3.13	Ball 1 case with masked south pole. (Top left) original RGB input image, (top right) new RGB input image with masked south pole (top-right), (bottom left) spherically adapted optical flow estimation of the original image and (bottom right) spherically adapted optical flow estimation of the masked image. The optical flow computed with the masked images shows less noise in the south pole. Thus camera tripod and equirectangular reconstruction add noise to the optical flow estimation.	68
3.14	Prediction examples in the Flow360 dataset. Spherical adaptation allows better tracking of objects moving in polar regions. As a result, the estimation of the optical flow of the observed lamp post is significantly improved (area highlighted by the red circle). (Top left) : RGB input, (top right) : ground truth optical flow, (bottom left) : prediction from the baseline network, (bottom right) : prediction from the adapted network.	71
3.15	Ball1 throw sequences. The adapted network provides correct optical flow estimation, whereas the baseline version loses track of the ball. (Top left) : RGB input frame at t , (top right) : RGB input frame at $t + 1$, (bottom left) : prediction from the baseline network, (bottom right) : prediction from the adapted network.	72
4.1	Navigation pipeline in [123]. Depth and obstacles detection are merged to build an Octomap.	79
4.2	Network outputs from a perspective image of a indoor corridor. [127]	80

4.3	Network outputs from a perspective image of a forest trail. The drone commands are: turn left, go straight or turn right. [130] . . .	81
4.4	Illustration of a deep neural network predictive model. The convolutional encoder takes as input the current RGB image and processes it to form the initial hidden state of a recurrent LSTM unit. This recurrent unit takes as input H actions in a sequential fashion and produces H outputs. These outputs of the recurrent unit are then passed through additional fully connected layers to predict all K events for all H future time steps. These predicted future events, such as position or collision, enable action selection and achieve desirable events. [132]	81
4.5	Combining global way-points in a reconstructed map and local obstacle avoidance based on learning [125]. Colored pins represent these global waypoints in the figure. The drone trajectory (green) often differs from the direct trajectory (red) due to the various obstacles.	82
4.6	Reinforcement learning framework. The agent selects an action based on its current state and policy. This interaction with the environment results in a new state and an additional reward to evaluate the relevance of the applied policy.	84
4.7	Policy iteration [131].	88
4.8	The first-visit MC method for estimating V_π [131].	88
4.9	Sarsa: An on-policy TD control algorithm [131].	89
4.10	Q-learning: An off-policy TD control algorithm [131].	89
5.1	Overview of the RDMAP environment.	93
5.2	Overview of the RDFOREST environment.	93
5.3	General DRL framework. The drone state gathers information about the goal relative position (d_k, θ_k) and a visual capture of the agent's environment I_k as defined in Section 5.2.2.	94
5.4	Different visual modalities considered for drone navigation: (From left to right: RGB, ground truth depth, and estimated depth with baseline MIDAS network [60]).	96
5.5	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, θ_k) , another fully connected network (RFC) predicts the agent's next action a_k . The specific case of estimated depth as input is presented in this example. The MIDAS network is not used when the navigation focuses on RGB images, ground truth depth or semantic segmentation.	97
6.1	The drone starts at the center and must reach its target on a 20-meter circle without obstacles. DQN solution is jerky, whereas TD3 and PPO are smoother.	101

6.2	Trajectory sample 1 for an identical goal at 60 meters on the RDMAP environment using 90° FOV (in orange on the left) and 360° FOV (in purple on the right) ground truth depth as input visual modality. Link to the video.	103
6.3	Trajectory sample 2 for an identical goal at 60 meters on the RDMAP environment using 90° FOV (in orange on the left) and 360° FOV (in purple on the right) ground truth depth as input visual modality. Link to the video.	104
6.4	Example of kernels with different latitude and longitude. In blue is the center of the kernel, in green the perspective kernel and in red the adapted equirectangular one. The wider distortions are near the poles.	105
6.5	Pre-computed spherical adapted offsets are added to the four convolution layers of the Actor-Critic network during training and testing of the navigation solution.	106
6.6	The convolutions of the MIDAS decoder are modified without additional network training to take into account spherical distortions for depth prediction. Therefore, the final proposed DRL pipeline has two spherical adaptations: one for the MIDAS network and one for the Actor-Critic network.	107
6.7	Trajectory sample 1 for an identical goal at 60 meters on the RDMAP environment using 90° FOV (in orange on the left) and 360° FOV (in purple on the right) estimated depth as input visual modality. The 360° FOV solution on the right use distortion-aware convolutions in the actor-critic and MIDAS networks. Link to the video.	108
6.8	Trajectory sample 2 for an identical goal at 60 meters on the RDMAP environment using 90° FOV (in orange on the left) and 360° FOV (in purple on the right) estimated depth as input visual modality. The 360° FOV solution on the right use distortion-aware convolutions in the actor-critic and MIDAS networks. Link to the video.	109
6.9	RDFOREST: (left to right: RGB, ground truth depth, and estimated depth with baseline MIDAS network [60]).	110
6.10	Trajectory sample for an 60 meters goal on the RDFOREST environment using 360° FOV estimated depth as input visual modality with spherical adaptation in DRL and MIDAS networks. Link to the video.	111
7.1	(c) Standard Patch Embeddings. (d) Deformable Patch Embedding for panoramas. [156].	114

7.2	Cyclic Flow Estimation. Partitioned feature maps are extracted from the encoder of the attended frame and the target frame. According to the cyclicity of the left and right boundaries of a panoramic image, the features extracted via the encoder, are re-grouped into two feature pairs and sent to the decoder to obtain the complementary optical flow field. The 360° flow can finally be obtained via <i>min</i> operations [110].	115
7.3	Controlling Stable Diffusion with ADE20K [58] segmentation map [159]. On the left is the control semantic segmentation control input used to create the different landscape images on the right. .	116

List of Tables

2.1	Comparison of different imaging systems.	31
2.2	Common activation functions and their derivatives.	35
2.3	Performances of deep learning based optical flow estimation methods. The comparison metric usually used is the End-Point-Error (EPE) [73] for Sintel and Things dataset and the percentage of optical flow outliers (F1) for KITTI dataset. For each method the computational time needed is also presented.	43
3.1	Different camera’s orientation given as Euler Angles (ϕ, θ, ψ) . . .	54
3.2	Comparison of spherically adapted and baseline semantic segmentation networks on RWFOREST 256×256 , 512×256 , and 1024×512 . Each dataset contains 1000 images.	58
3.3	Comparison of adapted and baseline depth estimation networks on RWFOREST 256×256 , 512×256 , and 1024×512 . Each dataset contains 1000 images.	62
3.4	Metrics value for the depth estimations in Figure 3.9.	63
3.5	Comparison of adapted and baseline LiteFlowNet2 on OmniFlowNet dataset. This dataset gathers 1200 images distributed in three different scenes: CartoonTree, Forest, and LowPolyModel.	66
3.6	Comparison of adapted and baseline optical flow networks on OmniFlowNet, Flow360, CityScene, and EquirectFlyingThings (EFT) datasets.	70
3.7	Comparison of the running time and model size between spherically adapted and baseline networks. We give the average computation time for processing one omnidirectional RGB image on a RTX3060 GPU. This average is computed using the complete datasets of 1000 images for RWFOREST and 1200 images for OmniFlowNet. Model and offset tables sizes are also provided.	73
3.8	Comparison of adapted and baseline networks on three different visual modalities. The error metric used for semantic segmentation is the complement of the Mean Average of Intersection Over Union, for optical flow is the End-Point Error, and for depth is the Absolute Relative Error.	75
4.1	V-table	87
4.2	Q-table	87

5.1	Network architecture for the actor-critic pipeline. Total: 60814 parameters.	97
6.1	Simulation hyperparameters.	100
6.2	Comparing Success Rate (SR in %) with TD3 and PPO as solver. Each evaluation is performed on 100 runs.	101
6.3	Comparing 90° and 360° modalities. Each evaluation is performed on 600 runs and 3 goal-distances (20, 40 and 60 meters). Ground Truth (GT), Estimated Depth (ED).	102
6.4	Performances in RDMAP.	106
6.5	Performances in RDMAP.	107
6.6	Performances in RDFOREST.	110

Abbreviations

AECE	Averaged Erroneous Class Estimate
AI	Artificial Intelligence
CNN	Convolutional Neural Network
DDPG	Deep Deterministic Policy Gradient
DP	Dynamic Programming
DQN	Deep Q-learning
DRL	Deep Reinforcement Learning
ED	Estimated Depth
EPE	End-Point Error
FC	Fully Connected
FOV	Field Of View
GPS	Global Positioning System
GPU	Graphics Processing Unit
GT	Ground Truth
LIDAR	LIght Detection And Ranging
LSTM	Long Short-Term Memory
MB	MegaByte
MDP	Markov Decision Process
MIoU	Mean Intersection over Union
PPO	Proximal Policy Optimization
RMSE	Root Mean Square Error
SLAM	Simultaneous Localization And Mapping
SPL	Success weighted by Path Length
SR	Success Rate
TD	Temporal Differences
TD3	Twin Delayed Deep Deterministic Policy Gradient
TRPO	Trust Region Policy Optimization
UAV	Unmanned Aerial Vehicle

Chapter 1

Introduction

1.1 Motivations and Challenges

Perception refers to the capacity to perceive and interpret an environment using different sensors. It is a complex process that involves collecting and analyzing sensory data to create a representation of the surroundings. Millions of years of evolution allow all living things to accomplish these tasks almost instantly and effortlessly. Computers, on the other hand, do not acquire these skills innately. As a result, computer vision has been studied intensively since the 1980s [1] to address these limitations, primarily to mimic what humans do naturally. The desire to develop autonomous mobile robots has accentuated this need and led to the development of numerous image-processing algorithms.

However, these methods have often been limited by sensor characteristics, particularly the Field Of View (FOV) for cameras. Traditional cameras have a field of view that ranges from 40 to 90 degrees. In contrast, human vision covers up to 140 degrees. This gap has led to the recent development of cameras with a wider field of view, notably inspired by nature with the fisheye lenses. These latest camera models allow omnidirectional vision, i.e., capturing the entire surroundings in a single shot.

Furthermore, computer vision has always been very demanding regarding computational power. Only thanks to the development of very powerful dedicated computing units, such as Graphics Processing Units (GPU), could image processing be performed in a limited time. Traditional model-based computer vision methods use a pre-existing model or theory to represent the problem accurately. However, the recent explosion in computing power has made it possible to process gigantic amounts of information compared to before. In particular, this has led to new data-driven approaches, which involve collecting and analyzing data to discover patterns, trends, and relationships between them.

Most computer vision is now done by data-driven algorithms and, more particularly, by deep learning [2]. This very recent method proposes to mimic the functioning of the human brain. By stacking layers of neurons, it can learn to perform very complex tasks. They have surpassed previous approaches and allowed significant breakthroughs in vision thanks to the first architectures of Convolutional Neural Networks (CNN).

Nevertheless, the use of omnidirectional vision is not trivial. All spherical projections have significant distortions [3], for example, the polar areas of equirectangular images, as shown in figure 1.1. As a result, the shapes of the objects in these areas are significantly altered: the car in the lower part of the image is difficult to identify. Because of these distortions, classical perspective machine vision methods, which often consider the position of points relative to each other, are often unsuitable for wide-angle images. Therefore, it is necessary to adapt the usual methods by taking into account the particularities of these spherical images.



Figure 1.1: Equirectangular image of an outdoor urban driving scene. The polar regions are strongly distorted, so much so that it is difficult to recognize the car in the lower part of the image.

Data-driven methods are powerful but highly sensitive to domain change: a situation in which the distribution of data used to train the model differs from the distribution on which the model will be applied. Several methods propose adaptation methods to deal with spherical distortions, but they all rely on creating an omnidirectional dataset and training a new model. Therefore, we offer an alternative solution to avoid these two laborious and time-consuming tasks and benefit from decades of computer vision development using perspective images. Our method relies on distortion-aware convolutions that modify the local operations performed on the image during its processing.

To demonstrate the generalization of our spherical adaptation, we apply it to three major computer vision tasks: semantic segmentation, depth, and optical flow estimation. We evaluate the baseline and spherically adapted models on virtual outdoor images and complex real-world scenarios not seen during training. The adapted methods demonstrate improved estimation accuracy and smoothness for each task considered.

Robotics benefits directly from improvements in computer vision. Indeed, thanks to these new methods, the perception of robots is considerably improved, allowing them to perform more complex tasks. For example, autonomous navi-

gation in dense and unstructured environments such as forests has always been a significant challenge in robotics. However, thanks to the latest technological advances, navigation reaches now outstanding levels of autonomy [4].

Applying the distortion-aware convolution strategy in robotics is a logical continuation of the first step in computer vision. In [5], the authors demonstrated that omnidirectional vision significantly improves Simultaneous Localization And Mapping (SLAM) compared to a conventional perspective camera. However, the camera’s capabilities often limit most newer and modern navigation algorithms, such as Deep Reinforcement Learning (DRL).

Deep reinforcement learning proposes to learn a navigation policy through trial and error experiments where the robot interacts with the environment based on its perception. A reward promotes or prevents specific behaviors and thus influences the learned policy. This strategy offers excellent generalization capabilities but requires a very long learning process, usually performed in virtual environments, to allow thousands of trials and exploration of failure cases. Nevertheless, this method has gained significant interest in solving robot navigation with increased computational power and realistic simulators.

Therefore, we propose a DRL aerial navigation solution based on equirectangular images and demonstrate its relevance, especially compared to its perspective version. Perception being crucial for navigation, we study several visual modalities: RGB and depth. For this last modality, we consider both ground truth depth and depth estimated from 360° RGB images using deep learning methods. In all cases studied, navigation using omnidirectional images outperforms the perspective reference.

In addition, we integrate the distortion-aware convolution strategy into our navigation solution to take into account the distortions in omnidirectional images. Once again, it significantly improves image processing, resulting in a higher navigation success rate. Finally, we evaluate the most promising models on a more photorealistic forest environment without additional training and demonstrate the robustness of the navigation algorithm.

1.2 Outline

We divide this manuscript into two parts. Part I presents the study focused on computer vision, while part II develops robot navigation.

Thus, in the first part, chapter 2 presents the different models of spherical cameras, an overview of the main methods applied in computer vision, especially focused on the approaches explicitly developed for omnidirectional images, and finally, the proposed spherical adaptation strategy: distortion-aware convolutions. Next, chapter 3 presents the virtual and real test datasets considered and gathers the evaluation of our spherical adaptation strategy, particularly comparing it to the baseline results for all studied visual modalities: semantic segmentation, depth, and optical flow.

In the second part, chapter 4 explores different image-based navigation methods and presents the chosen strategy based on deep reinforcement learning. Then, chapter 5 introduces the virtual flight environments and develops the framework

adopted for our solution. Finally, chapter 6 gathers all the evaluations of the considered navigation solution: the comparison between omnidirectional and perspective images, the improvements thanks to the distortion-aware convolutions, and the tests on a photorealistic environment without additional training.

Finally, chapter 7 presents the general conclusion and perspectives.

1.3 Contributions

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

1.3.1 Peer-Reviewed Publications

This manuscript is based on the material published in the following papers:
For Part I:

- C.-O. Artizzu et al. “OmniFlowNet: a Perspective Neural Network Adaptation for Optical Flow Estimation in Omnidirectional Images”. In: *Proceedings of the International Conference on Pattern Recognition (ICPR)*. Milan, Italy: IEEE, Jan. 2021, pp. 2657–2662 [6]
- C.-O. Artizzu et al. “OMNI-CONV: Generalization of the Omnidirectional Distortion-Aware Convolutions”. In: *Journal of Imaging* 9.2 (2023), pp. 1–16 [7]

For Part II:

- C.-O. Artizzu et al. “OMNI-DRL: Learning to Fly in Forests with Omnidirectional Images”. In: *Proceedings of the Symposium on Robot Control (SYROCO)*. Matsumoto, Japan: IFAC, Oct. 2022, pp. 1–6 [8]
- C.-O. Artizzu et al. “Deep Reinforcement Learning with Omnidirectional Images: application to UAV Navigation in Forests”. In: *Proceedings of the International Conference on Control, Automation, Robotics and Vision (ICARCV)*. Singapore, Singapore: IEEE, Dec. 2022, pp. 1–6 [9]

This article was Finalist for the Best Student Paper Award.

1.3.2 Open-Source Datasets and Softwares

We have released the following open-source datasets and software:

- Source code and dataset for “OmniFlowNet: a Perspective Neural Network Adaptation for Optical Flow Estimation in Omnidirectional Images” [6] can be downloaded from the project GitHub. This omnidirectional optical flow dataset gathers 1200 RGB images and the associated ground truth optical flow.

- Source code and dataset for "OMNI-DRL: Learning to Fly in Forests with Omnidirectional Images" [8] and "Deep Reinforcement Learning with Omnidirectional Images: application to UAV Navigation in Forests" [9] can be downloaded from the project GitHub. The RDMAP environment executable is provided with AirSim [10] simulator embedded.
- Source code for "OMNI-CONV: Generalization of the Omnidirectional Distortion-Aware Convolutions" [7] can be downloaded from the project GitHub.

Part I

**Generalization of the
Omnidirectional
Distortion-Aware Convolutions**

Chapter 2

Omnidirectional Computer Vision

A standard perspective camera has a relatively limited field of view, usually in the range of 40 to 90, which is a limitation in many applications. An omnidirectional sensor overcomes this drawback by providing a panoramic view of the scene up to 360°. As a result, the use of spherical cameras for robotic applications is proliferating. Recently, considerable effort has been devoted to the development and commercialization of more accurate and affordable devices. The latest generation cameras offer representative spherical images with up to 4K resolution, such as Instax 360 One X, Ricoh Theta Z1, and Samsung Gear 360, to cite a few.

In parallel, deep learning has significantly contributed to computer vision over the past decade. Thanks to the increased computing power of GPU, computers can now process large amounts of data. As a result, model-based image processing has been gradually replaced by learning-based approaches. Inspired by how the human brain works, deep learning algorithms can learn different tasks from specific datasets and then repeat them. However, these approaches are very sensitive to domain shift: the differences in data distribution or characteristics between the training and test datasets.

In particular, all spherical projections contain significant distortions. These distortions can significantly reduce performance if not taken into account during image processing. Some works propose adaptations to improve existing methods. In this thesis, we have chosen an elegant and simple approach that can be easily integrated into already-developed networks. To demonstrate the effectiveness and relevance of our solution, we applied it to three commonly used computer vision tasks: semantic segmentation, depth, and optical flow.

In this chapter, we first present the different models of spherical cameras, particularly the ones used during this thesis. Then, we briefly overview the deep learning algorithms used in computer vision, especially for the three visual modalities studied. Finally, we present the different methods to improve spherical image processing.

2.1 Physical Devices

The construction of omnidirectional cameras usually faces several challenges, such as field of view, focal length, resolution, and volume. Depending on the intended use scenarios, a trade-off must be found between these parameters. Several physical solutions have been proposed and can be classified as rotating, catadioptric, fisheye, or polydioptric cameras.

Table 2.1: Comparison of different imaging systems.

System	Real time	Image stitching	Compactness	Blind area
Rotating	No	Yes	Low	No
Catadioptric	Yes	No	Medium	Yes
Polydioptric	Yes	Yes	High	No

2.1.1 Rotating Cameras

Rotating cameras are the oldest method used to create omnidirectional images. The first panoramic image was captured in the 1900s using a perspective camera rotating around a vertical axis. The spatial resolution of the image is directly linked to the rotation speed. More recent versions of such devices use a larger number of freedom, such as Pan Tilt Zoom cameras commonly used in surveillance. They can create high-resolution omnidirectional images at the cost of a long acquisition time.



Figure 2.1: Pan Tilt Zoom cameras. Left: Axis. Right: Honey Optics.

2.1.2 Catadioptric Cameras

Catadioptric cameras use both refractive and reflective optics to capture images. The term catadioptric originates from dioptrics, the science of light refraction (lenses), and catoptrics, the science of reflection (with reflective surfaces like mirrors). Catadioptric cameras are known for their ability to have a wide field of view while maintaining a relatively long focal length. Conventional cameras are combined with different mirror shapes, such as parabolic, hyperbolic, or elliptical, to capture omnidirectional images. Catadioptric cameras were democratized in robotics for their ability to capture a large area with a single camera without the need for synchronization (unlike a multi-camera system). However, there are some drawbacks, notably the size, the cost, and the low and non-uniform resolution.



Figure 2.2: A catadioptric camera and a captured spherical image [11].

2.1.3 Fisheyes Cameras

A fisheye camera uses a short focal lens to capture a wide field of view, typically around 180 degrees or more. It is called a fisheye lens because the produced image is distorted, with straight lines appearing curved, similar to how a fish might see the world from beneath the water. Similarly to the catadioptric systems, the spatial resolution of the captured image varies, with higher pixel density at the center. Its main advantage over the previous camera is the possibility to remove the blind spot characteristic of catadioptric systems.

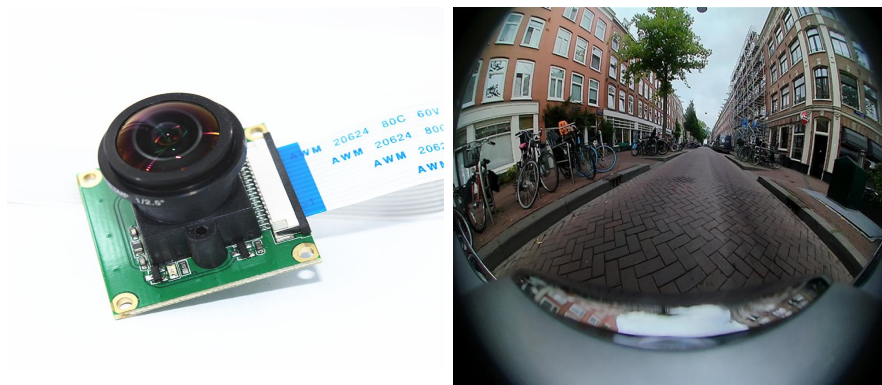


Figure 2.3: A fisheye camera and an image captured during urban driving [12].

2.1.4 Polydioptric Cameras

Polydioptric cameras contain multiple lenses of different focal lengths arranged in a way that allows them to capture images with a wide field of view. They can be used to create panoramic content or to correct for distortions in the image caused by the use of a single lens. These vision systems allow a quick acquisition of very high-resolution panoramic images. But the calibration task can be very challenging due to the lack of information on the inter-camera transformations or time synchronization between the cameras. The most common polydioptric systems are dual fisheye cameras, which use two fisheye lenses usually placed back-to-back and facing opposite directions, as in Figure 2.4.



Figure 2.4: The Ricoh Theta Z1 camera combines two fisheye lenses to reconstruct equirectangular images.

Some polydioptric systems use more lenses in specific circular or spherical configurations, as shown in Figure 2.5. These builds increase the stitching accuracy in return for higher computational cost and size. Each camera records a small area of the environment which is then stitched together to form an omnidirectional image. The number of cameras used depends on the lenses' focal length: the smaller the focal, the larger the field of view, and therefore fewer cameras are needed.

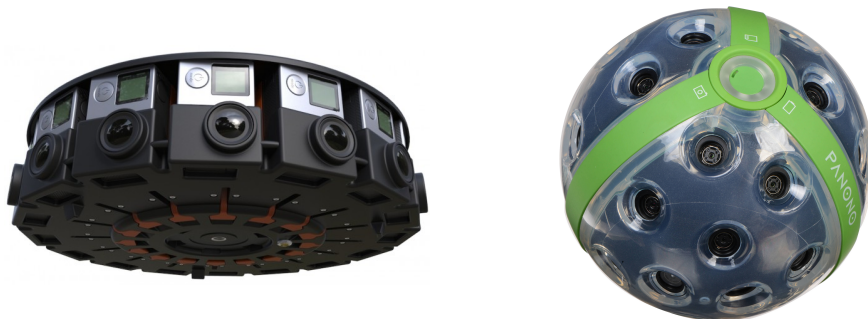


Figure 2.5: Polydioptric camera systems using circular rigs (GoPro Odyssey on the left) or spherical configurations (Panono 360 on the right).

2.2 Data-Driven Computer Vision

In the past decade, deep learning has made significant contributions to the field of computer vision and is now used in most state-of-the-art algorithms. One of the critical advantages of deep learning is its ability to learn and make decisions directly from data without the need for explicit programming or hand-engineered features. Previously, traditional machine learning approaches typically required manually transforming the data into a set of features suitable for the learning algorithm. Thanks to deep learning algorithms, the relevant features are directly learned from the data, leading to better performance and more accurate results. Here we provide a brief overview of deep learning approaches and their application to specific computer vision tasks.

2.2.1 Deep Learning

The key element of deep learning is artificial neural networks [2], which mimic the structure and function of the human brain. Composed of interconnected units called neurons, they learn from the data by adjusting the weights and biases of the connections between the neurons.

2.2.1.1 Artificial Neuron

Inspired by biological neurons, the first definition of the artificial neuron dates back to 1943 [13]. Let's consider an input $\mathbf{x} = [\mathbf{x}_0, \dots, \mathbf{x}_n]$, the neuron outputs:

$$\mathbf{y} = \sigma \left(\sum_{i=0}^n \mathbf{w}_i \cdot \mathbf{x}_i + \mathbf{b}_i \right), \quad (2.1)$$

where \mathbf{w}_i and $\mathbf{b} = \sum_{i=0}^n \mathbf{b}_i$ are internal parameters of the neuron, respectively weight and bias. σ is an activation function that can add a non-linearity. Common activation functions and their derivatives are presented in Table 2.2.

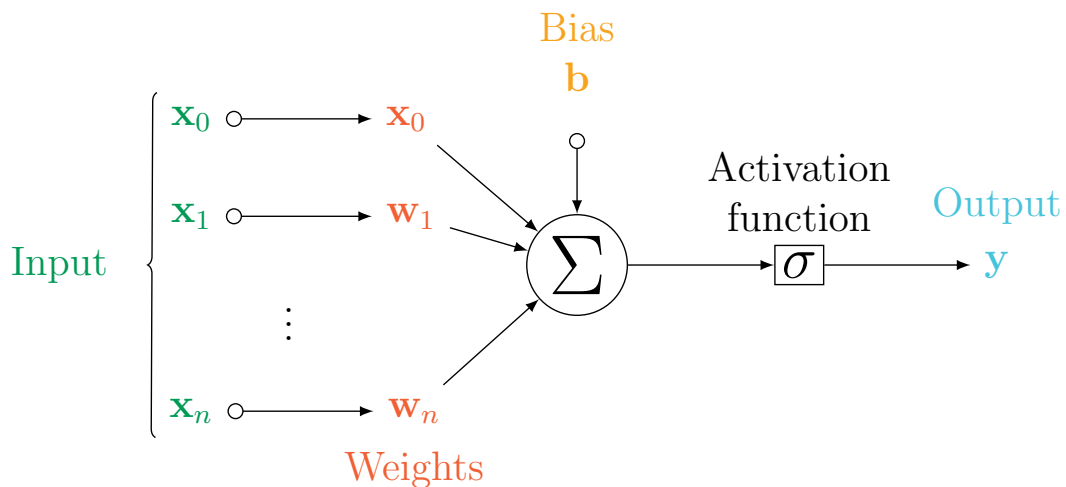


Figure 2.6: Artificial Neuron.

Table 2.2: Common activation functions and their derivatives.

Name	Function	Derivative
Sigmoid	$\sigma(x) = \frac{1}{1 + e^{-x}}$	$\sigma'(x) = \sigma(x)(1 - \sigma(x))$
TanH	$\sigma(x) = \frac{2}{1 + e^{-2x}} - 1$	$\sigma'(x) = 1 - \sigma(x)^2$
ReLU	$\sigma(x) = \begin{cases} 0 & x \leq 0 \\ x & x > 0 \end{cases}$	$\sigma'(x) = \begin{cases} 0 & x \leq 0 \\ 1 & x > 0 \end{cases}$
Leaky ReLU (α)	$\sigma(x) = \begin{cases} \alpha x & x \leq 0 \\ x & x > 0 \end{cases}$	$\sigma'(x) = \begin{cases} \alpha & x \leq 0 \\ 1 & x > 0 \end{cases}$

2.2.1.2 Artificial Neural Networks

Multiple network architectures can be created by connecting various artificial neurons and stacking them using numerous layers, the first of which was proposed in [14]. Inputs are first fed into the first layer of artificial neurons, called the input layer. The output of the input layer then passes through one or more additional layers, called hidden layers, where the computations become increasingly complex. Lastly, the output layer produces the final output of the neural network. Each layer contains some neurons connected to the others in the adjacent layers. If each neuron is linked to all neurons of the previous layer, this is a fully connected layer. The required number of hidden layers and neurons depends on the complexity of the task and the application.

The weights and biases are adjusted in supervised learning based on the error between the predicted output and the ground truth. This error is calculated using a loss function that differs depending on the desired behavior of the network. The gradient of the loss function is computed one layer at a time with respect to the other weights in the network, and this process is conducted in reverse through the entire neural network, hence named the back-propagation. After computation, nodes with a high error will have a smaller weight than those with a lower error.

The activation functions used in the network are also crucial to learn a broader range of functions and represent more complex patterns. For many years, sigmoid and tanh functions were popular in deep learning, but they suffer from the vanishing gradients issue: the gradients of the weights can become very small, which makes learning the network difficult. With the increasing popularity of deep networks, the ReLU activation [15] function gained interest for two main reasons: its simplicity and the fact that its gradient is equal to one or zero. But this has also led to the dying ReLU problem: the neuron can "die" during learning if it constantly produces a negative output. LeakyRelu [16, 17] solves this problem by allowing a small non-zero gradient when the input is less than 0 and becomes the reference activation function.

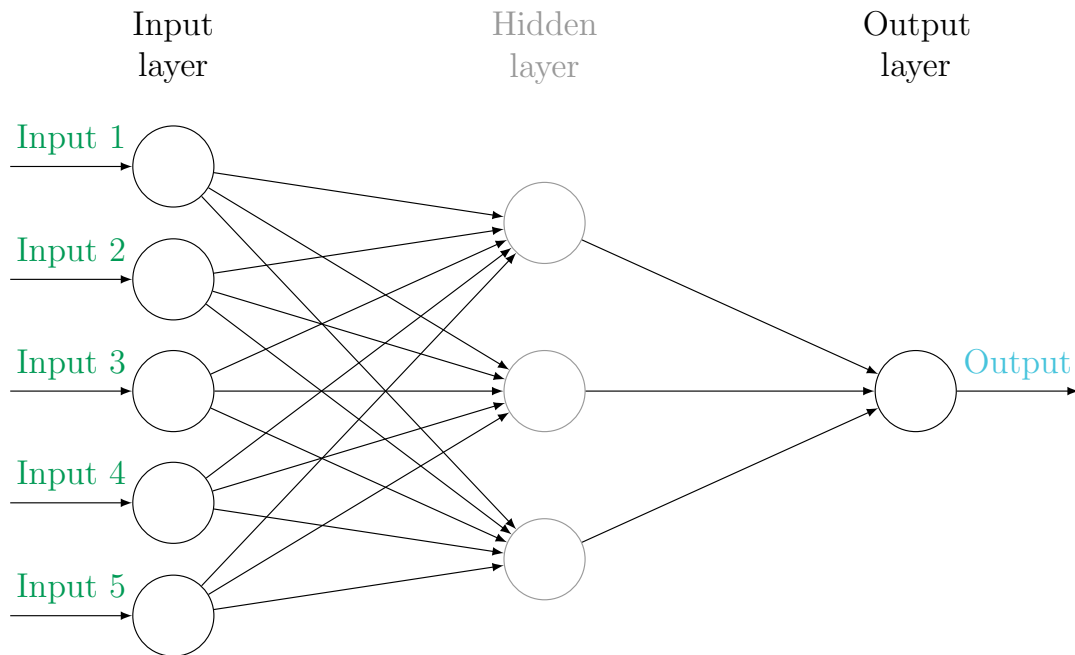


Figure 2.7: Artificial neural network with a single hidden layer. More precisely, it is a fully connected architecture because each neuron is connected to all neurons of the previous layer.

2.2.1.3 Convolutional Neural Networks

Convolutional neural networks have been developed to process images and now dominate most computer vision algorithms. They use a succession of learnable filters, i.e. convolutional layers, to extract features from the input image. These filters are designed to detect specific patterns or features in the input data, such as edges, corners, or textures. Then the output of these layers is often passed through one or more pooling layers (max-pool or more recently strided convolutions), which downsample the data to reduce its dimensionality. Depending on the computer vision tasks, the output is either passed through a fully-connected layer to perform the final classification task or is upsampled (deconvoluted) to reconstruct a new image. Most of the new networks are built using lighter published and proven architectures such as LeNet [18], AlexNet [19], VGG [20], ResNet [21], and MobileNet [22].

The fundamental operation of these networks is convolution. It adds to the value of each image pixel, a combination of neighboring pixels weighted by a filter. One significant advantage of this operation is that the convolutions are spatially equivariant. If the image contains the same features in two different regions, the outputs in these regions will be the same.

Let's consider a convolution filter \mathbf{w} applied on a 2D input features map \mathbf{x} . For each location \mathbf{p}_0 , the convolution output \mathbf{y} is defined by:

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n), \quad (2.2)$$

where \mathcal{R} is a regular grid sampled over the input map \mathbf{x} . This grid defines the filter field size and dilation $\mathcal{R} = a\vec{u} + b\vec{v}$ where (\vec{u}, \vec{v}) is the pixel coordinate system and (a, b) the sampling. For example, a 3×3 regular kernel with dilation 1 is defined by $(a, b) \in (-1, 0, 1)^2$ as illustrated in Figure 2.8.

$$\begin{pmatrix} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} * \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 4 & 3 & 4 & 1 & & \\ 1 & 2 & 4 & 3 & 3 & & \\ 1 & 2 & 3 & 4 & 1 & & \\ 1 & 3 & 3 & 1 & 1 & & \\ 3 & 3 & 1 & 1 & 0 & & \end{pmatrix}$$

Input \mathbf{x}
Filter \mathbf{w}
Output \mathbf{y}

Figure 2.8: Convolution operation of a 3×3 filter applied on a 7×7 input.

2.2.1.4 Transformer Networks

First used for natural language processing tasks, Transformer networks [23] have recently been applied to computer vision tasks. These networks use an attention mechanism that distributes weights over certain input parts. It allows the model to focus better on the most relevant input features.

The attention operation performs three linear transforms on the input $\mathbf{X} = [\mathbf{x}_0, \dots, \mathbf{x}_n]$ in order to produce a sequence of keys $\mathbf{K} = [\mathbf{k}_0, \dots, \mathbf{k}_n]$, values $\mathbf{V} = [\mathbf{v}_0, \dots, \mathbf{v}_n]$ and queries $\mathbf{Q} = [\mathbf{q}_0, \dots, \mathbf{q}_n]$. By performing a dot-product comparison of similarity between the queries and keys, the network can learn to attend to different semantic concepts and features in the input sequence:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax} \left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}} \right) \mathbf{V}, \quad (2.3)$$

where the normalizing factor d_k is the size of the key vector. The attention distribution is calculated between the key and query pairs. Still, the output is a weighted average of the value vectors, allowing the network query to pass on information different from those used to calculate the attention score. By stacking and repeating this operation, the network can learn complex tasks.

For computer vision applications, the image input is split into a sequence of regions and then processed through transformer-based models. One of the main architectures is the Vision Transformer (ViT) [24] and was initially used for image classification as presented in Figure 2.9. In addition, the reader can refer to the following surveys for more details: [25, 26].

Many Transformer networks use convolutional neural networks to be more sampling-efficient [27, 28, 29]. They are either used as an encoder to extract high-level features from the input image, as a decoder to reconstruct the final output, or as a positional encoding to constrain the transformer attention.

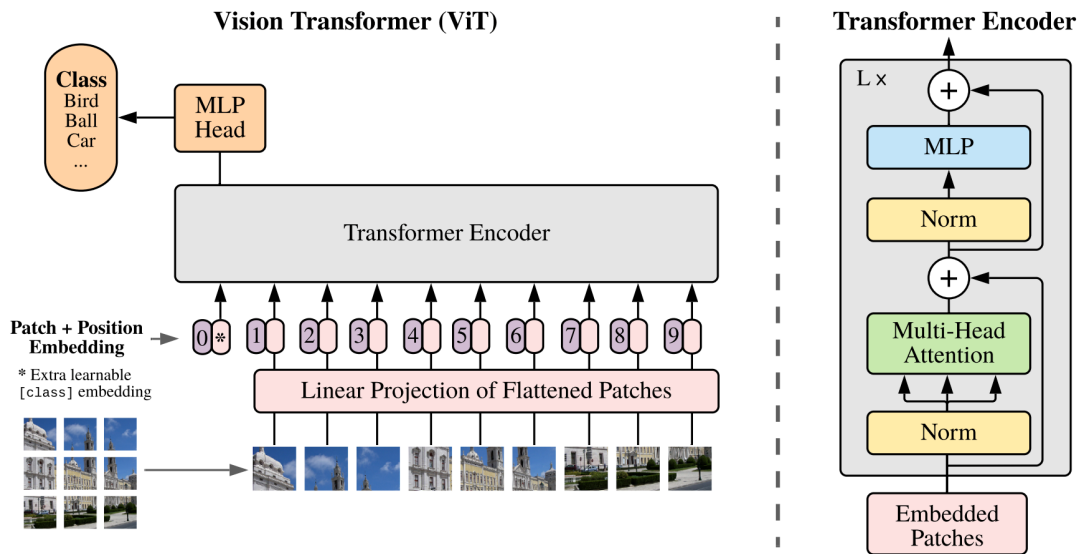


Figure 2.9: The Vision Transformer architecture [24]. The image is split into fixed-size patches with position embeddings and fed into a Transformer encoder. The output is a class estimation.

2.2.2 Common Computer Vision Tasks

Computer vision includes many tasks that aim to provide dense information about the captured scene, allowing pixel-accuracy understanding. The three most commonly used visual modalities are semantic segmentation, depth, and optic flow estimation. Each task carries crucial information and offers a good complementarity for scene understanding. For example, semantic segmentation provides the category and the position of each object/element of the scene; depth indicates their relative distance with respect to the camera; optical flow measures their relative displacement or the observer’s motion.

These three tasks also have particular requirements related to their purpose. They offer a good overview of the diversity of computer vision applications. Image classification, instance segmentation, and saliency are standard computer vision tasks but can be considered a related case of semantic segmentation. Similarly, surface normals prediction is related to depth estimation, and the focus of expansion computation is related to optical flow.

For these reasons, we focused on the three main tasks discussed above: semantic segmentation, depth, and optical flow estimation.

2.2.2.1 Semantic Segmentation

Image segmentation consists in dividing pixels into groups according to specific criteria. In particular, semantics assigns a semantic label to each pixel in an image like ground, trees, or sky, as shown in Figure 2.10. The image’s low-level and high-level features are needed to build a system that determines semantic meaning at the pixel level. Low-level features, such as edges, can be obtained from local intensity variations, while high-level features require a semantic understanding of the image. As a result, the output image provides a holistic and accurate understanding of the camera environment.

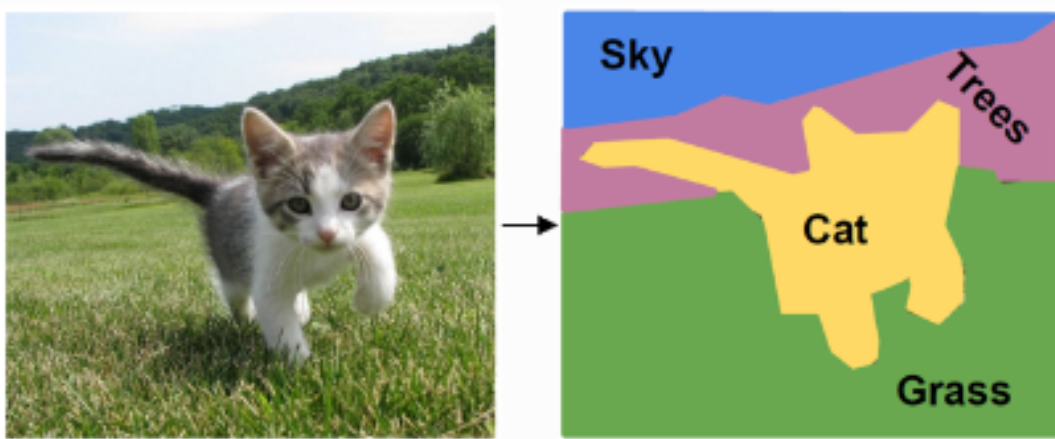


Figure 2.10: An example of semantic segmentation [30].

Early image segmentation algorithms can be divided into three techniques: region detection, edge detection, and graph-based segmentation. Region-based segmentation divides an image into multiple regions based on uniformity criteria such as intensity, color, or distance. This can be done using methods such as clustering [31], region growing [32], and thresholding. On the other hand, edge detection involves defining segments based on a lack of continuity. It often uses the intensity gradient to determine whether a pixel can be classified as an edge [33, 34, 35, 36] (edge points have a high-value gradient). Finally, another approach uses probabilistic graphical models such as Conditional Random Field [37, 38], which models the relationship between pixels and image segmentation.

After the massive success of deep learning on image classification with LeNet-5 [18], it was extended to semantic segmentation. However, this last task required additional spatial information to discriminate pixels correctly. Therefore, the most successful deep learning semantic segmentation architectures usually use a pretrained classification network like VGG [20] and ResNet [21] as the encoder. Then the decoder projects the extracted discriminative features from the encoder onto the pixel space to get a dense classification of every pixel in the input image. Specific decoder architectures were proposed, such as the Pyramid Pooling Module [39] or the Atrous Spatial Pyramid Pooling [40]. One of the first networks used for semantic segmentation is a fully convolutional network [41]. Since then, various more advanced architectures have been proposed, such as UNet [42], YOLO [43], SegNet [44], and DeepLab [40].

In particular, the *ADE20K* dataset [45] is a reference dataset to train and test semantic segmentation solutions. It mixes indoor and outdoor images in 20000 scenes with 150 ground truth semantic classes. The authors proposed a classic encoder-decoder architecture in [46], demonstrating good accuracy and robustness. This network combines a ResNet50 [21] encoder, and a Pyramid Pooling Module [39] decoder.

2.2.2.2 Monocular Depth

The depth estimation problem has been a long-standing problem in computer vision, especially for robot navigation. Initially, this task was solved using LIDAR or stereo cameras. But the cost, weight, and power consumption of these sensors motivated the development of new methods based on monocular cameras. But depth estimation from a single image is an ill-posed problem [47] since the same image could be produced by an infinite amount of real-world scenes as shown in Fig. 2.11.

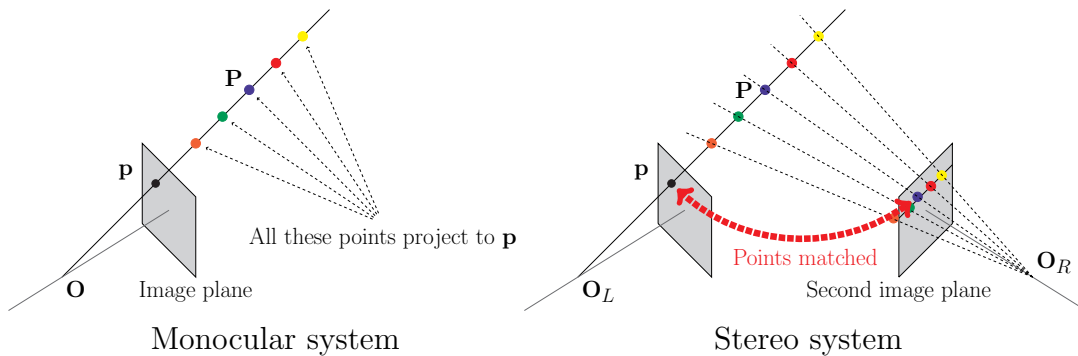


Figure 2.11: Estimating depth from a single image is an ill-posed problem [47] since several real points project onto the same pixel. Using a second camera alleviates this ambiguity by matching points (drawing inspired by [48]).

Most model-based existing approaches aim to reconstruct a stereo vision system and find correspondences between pixels [49, 50, 51]. But they are often limited by their reliance on optical flow or scene assumptions. On the other hand, deep learning brought significant progress in monocular depth estimation thanks to convolutional neural networks. Depth estimation is a dense prediction task in which each pixel in the image is assigned a depth value. It is closely related to semantic segmentation that aims to assign each pixel a class label from several categories.

The first convolutional neural network for monocular depth estimation was published in 2014 [52]. Their solution features a coarse-to-fine scheme and uses supervised training with ground truth depth, thanks to an RGB-D camera. First, the coarse network predicts the scene’s depth at a global level taking advantage of the entire image. Then, this coarse output is combined with the original input image and refined using local information. Following this work, they proposed a multitasking network estimating jointly depth, normals, and semantic labels [53].

Dense depth estimation has also been addressed as a classification task. In [54], the authors propose to transform the depth regression task into an ordinal regression problem. They divided the depth range into a fixed number of bins of predetermined width. Adabins [55] proposes a generalization of this work by computing adaptive bins based on the characteristics of the observed scene and predicting the final depth as a linear combination of the centers of these bins.

To overcome the limitation of annotated datasets, another approach is to use self-supervised learning. Most of these methods seek to mimic a stereo system, either by image reconstruction or video sequences. The most famous network based on image reconstruction is Monodepth [56]. It generates disparity images by training the network with an image reconstruction loss based on left-right consistency. The authors proposed a second version, Monodepth2 [57], which improves the loss function to handle occlusions, adding a multi-scale sampling method to reduce visual artifacts and auto-masking loss to ignore training pixels that violate camera motion assumptions. In parallel, the use of image sequences has also been explored. For example, in [58], the authors use video recordings where the variation in camera position between successive frames is small and can be considered a stereo system.

Recently, vision Transformers are replacing convolutional networks as the backbone for dense prediction tasks, such as monocular depth [27]. Thanks to a global receptive field at every stage and high-resolution feature maps, this solution demonstrated a significant improvement compared to the previous models.

However, all previous models require considerable computational power, usually not available on resource-constrained devices such as mobile robots. Nevertheless, depth is the most commonly used visual modality for obstacle avoidance in navigation, primarily due to its robustness to scene changes. As a result, considerable efforts have been put into reducing and simplifying the network architectures. Supervised methods are particularly well suited for this size reduction. FastDepth [59] presents one of the first solutions optimized for depth estimation in indoor scenes using smartphones. More versatile, MIDAS network [60] is trained on multiple indoor and outdoor perspective datasets and offers accurate and robust monocular depth estimation while being one of the lightest networks.

2.2.2.3 Optical Flow

When an observer is in relative motion with respect to the objects in its environment, a visual field of displacement can be perceived. This field is called optical flow. Computationally, the estimation methods aim to compute the apparent motion of pixels between two frames. It enables autonomous vehicles and robots to acquire temporal cues of the surrounding scenes. The scene flow is generally expressed using color for the motion direction and intensity for its magnitude, as shown in Figure 2.12.

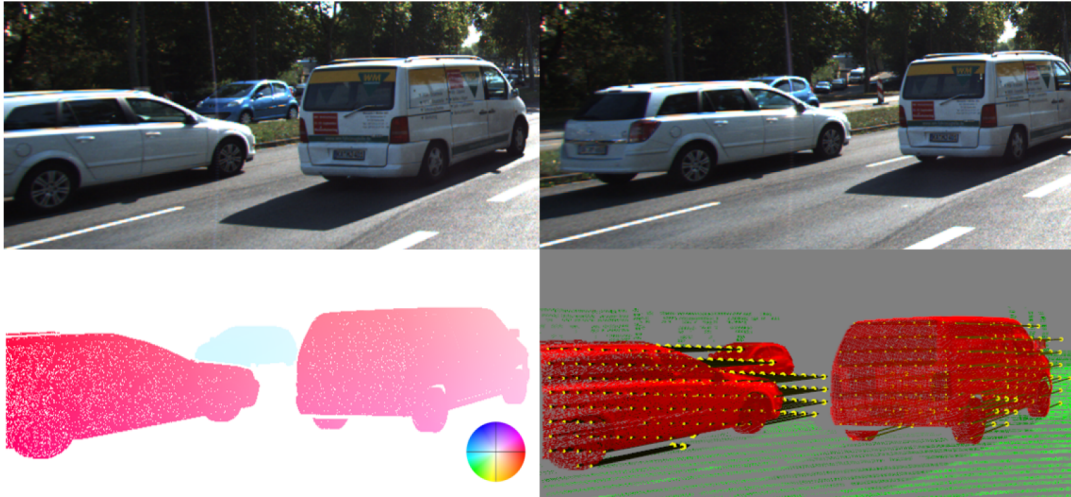


Figure 2.12: Optical flow between two frames [61]. (**Top left**): RGB input at t , (**top right**): RGB input at $t + 1$, (**bottom left**): ground truth optical flow with reference color wheel, (**bottom right**): Flow vectors between pixels.

Many solutions have been proposed to estimate the apparent motion in perspective images. In this section, we present the traditional approach and then the current state-of-the-art methods dominated by deep learning. Let's consider a 3D point in the camera coordinate system $\mathbf{P} = (X, Y, Z) \in \mathbb{R}^3$ projected onto the image plane at the point $\mathbf{p} = (x, y, t)$ and whose light intensity is $I(x, y, t)$. Given Δx and Δy the displacement of this pixel between two image frames and Δt the time variation, the brightness constancy constraint means that the light intensity of the point \mathbf{p} does not change over time. This translates mathematically as:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t). \quad (2.4)$$

We can use a Taylor expansion to develop the previous equation assuming small displacements. If we neglect the second-order terms, the optical flow is then constrained by the following equation:

$$\frac{\partial I}{\partial x} \Delta x + \frac{\partial I}{\partial y} \Delta y + \frac{\partial I}{\partial t} \Delta t = 0. \quad (2.5)$$

Considering the optical flow $\vec{V} = (V_x, V_y) = \left(\frac{\Delta x}{\Delta t}, \frac{\Delta y}{\Delta t}\right)$, the final equation is:

$$\vec{\nabla} I \cdot \vec{V} + \frac{\partial I}{\partial t} = 0. \quad (2.6)$$

However, this equation has two unknowns (V_x, V_y) , so it cannot be directly solved. Several approaches have been proposed to tackle this aperture problem.

The first published methods [62, 63] are based on a variational formulation of the problem. It tackles the flow estimation problem as an energy minimization formulation. These energy functions are based on the brightness constancy equation and spatial smoothness constraints. However, these assumptions are often violated in the presence of large motion, occlusion, and lighting variations.

To better handle large motion, more recent methods adopt a pyramidal processing and warping approach for coarse-to-fine flow estimation [64, 65].

In recent years, significant progress has been made with the publication of two large annotated datasets for optical flow estimation in perspective images: Sintel [66] and KITTI [67]. Thanks to these datasets, learning-based methods could be trained in a supervised manner. One of the most cited networks is FlowNet [68] which achieved state-of-the-art performances in a significantly lower computational time than traditional approaches.

Architectures built upon this legacy [69, 70, 71] have dominated optical flow estimation in perspective images for almost a decade, as shown in Table 2.3. Using a classical encoder-decoder architecture, they are trained using a large dataset of image pairs and ground truth optical flow. The input of the CNN is a pair of successive perspective images, and the output is an estimate of the (V_x, V_y) flow fields. The encoder and decoder are connected to preserve both high-level information of the coarser feature maps and low-level information.

Recently, optical flow estimation was significantly improved. First, RAFT [72] proposed a classically inspired iterative residual optical flow refinement updated by a recurrent unit performing lookups on 4D correlation volumes. Then, this correlation operation was optimized thanks to Transformer networks allowing for global features matching. GMFlow [28], based upon this approach and a self-attention layer for flow propagation, is now one of the best-performing estimation solutions. Despite the new Transformer architecture, this strategy relies heavily on convolutional neural networks to preprocess the input images and extract high-level features.

Table 2.3: Performances of deep learning based optical flow estimation methods. The comparison metric usually used is the End-Point-Error (EPE) [73] for Sintel and Things dataset and the percentage of optical flow outliers (F1) for KITTI dataset. For each method the computational time needed is also presented.

Method	Year	Sintel [66] EPE (\downarrow)	KITTI [67] F1 (%) (\downarrow)	Things [74] EPE (\downarrow)	Time sec (\downarrow)
Horn & Schunk [62, 75]	1981	9.61	28.86	-	>150
DeepFlow [65]	2013	7.21	28.48	-	17
FlowNet [68]	2013	7.76	-	-	1
LiteFlowNet2 [71]	2020	4.69	7.62	-	0.05
RAFT [72]	2020	2.86	4.74	4.25	0.1
GMFlow [28]	2022	2.65	9.67	2.80	0.1

2.3 Omnidirectional Image Processing

As shown in Section 2.2, most state-of-the-art computer vision algorithms use convolutional neural networks. However, these networks rely heavily on their training datasets and are very sensitive to domain shift. Creating an accurate and complete dataset with associated ground truth is labor-intensive and time-consuming. Therefore, most existing datasets consist of perspective images. Omnidirectional sensors capable of extracting ground truth are rare, complex to calibrate, and often prone to reconstruction errors. There are several recent attempts to construct spherical reference datasets. Although training networks on these datasets is possible, extending the application to real cases or outdoor images is not trivial.

In addition, all spherical projections come with significant distortions. In a perspective image, the spatial resolution is generally uniform and defined by two parameters (width and height). We can easily define a region of interest centered on a pixel by creating a rectangle of neighboring pixels. But this classical sampling is not adapted to omnidirectional images because it considers a uniform resolution of the image and does not take into account the distortion of spherical projections. Despite the format (cubic, cylindrical, equirectangular, Mercator, stereographic, etc.), they all come with distortions [76].

In this thesis, we choose the equirectangular projection to represent spherical images. This projection is commonly used for its ease of reading and classic rectangular format. The latitude and longitude of the spherical image are projected in horizontal and vertical coordinates onto a 2D plane. As a result, the equirectangular projection suffers from significant distortions near the polar regions, as shown in Figure 2.13. Because of this non-linearity, objects appear differently depending on their latitudes.

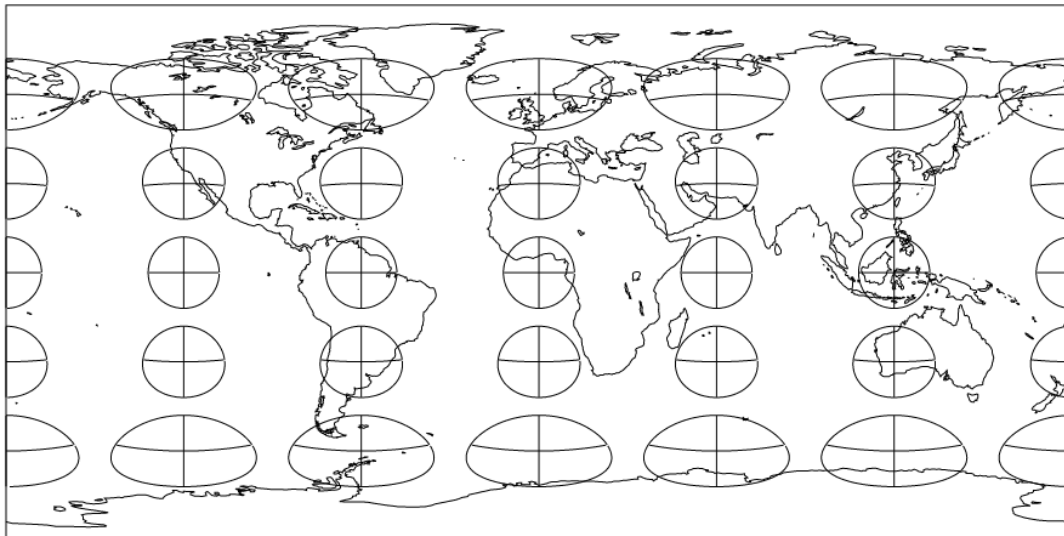


Figure 2.13: Equirectangular projection of the Earth's globe. The Tissot's indicatrix illustrate the amplitude of distortions [76]: a circle whose shape is regular near the equator is significantly distorted near the poles.

Different approaches have been proposed to take these distortions into account, such as training on omnidirectional datasets, spherical latent space, multi-projection fusion, and deformable convolutions.

2.3.1 Training on Omnidirectional Datasets

Most existing datasets are with perspective images and only some with omnidirectional ones. Building an accurate and complete dataset is labor-intensive and time-consuming. Early publications proposed to perform data augmentation on perspective datasets to fit fisheye distortions and use these spherically augmented datasets to train semantic segmentation [77, 78] or optical flow [79] networks.

Several recent attempts have been made to construct spherical reference datasets, such as Matterport3D [80], Stanford-2D3D [81], 3D60 [82], Structure3D [83], Pano3D [84]. Some publications have used these datasets to train a network for a specific omnidirectional task such as image classification [85], semantic segmentation [86], layout estimation [87], monocular depth estimation [82]. But these solutions are limited to indoor office scenes and does not contain ground truth optical flow. Extending the trained applications to outdoor images or different scenes is not trivial. Therefore, developing a new generalized adaptation method from pretrained networks on perspective images is in high demand for omnidirectional applications.

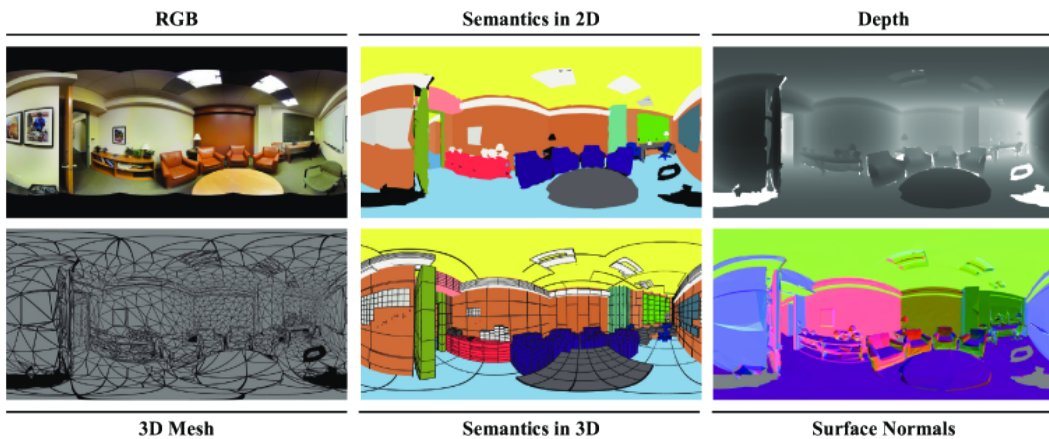


Figure 2.14: Stanford2d3d dataset [81].

2.3.2 Spherical Latent Space

Several works add distortion awareness on the latent space features using a specific mathematical formulation such as the fast Fourier transform [88, 89] or polyhedra [90, 91], as shown in Figure 2.15. However, despite the elegance of these solutions, the adapted network must be trained from scratch with specific training datasets. In addition, the adaptation methods are very computationally demanding. Therefore, it is challenging to implement such strategies on resource-constrained devices for real-time robotic applications.

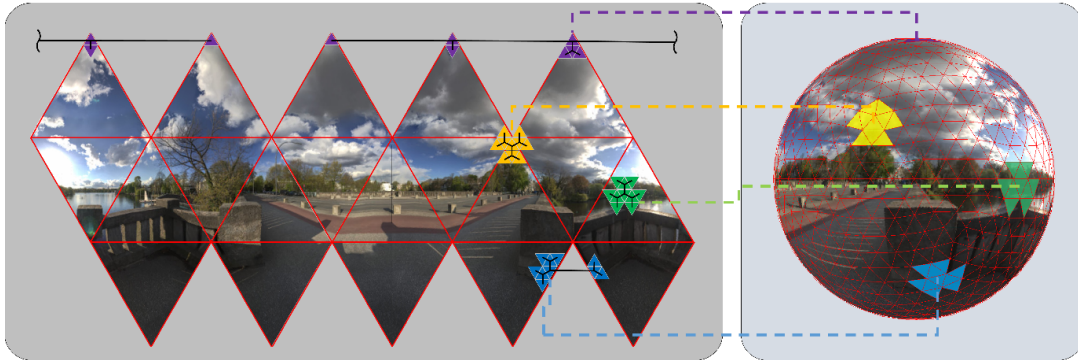


Figure 2.15: The visualization of how the kernel is applied to polyhedra representation [90]. **Yellow kernel** shows the case when the kernel is located at the vertex of the icosahedron. **Blue kernel** shows the case when the kernel is located at the pole.

2.3.3 Multi-Projection Fusion

Fusing multiple scene projections can create a more complete and accurate representation. As a result, multi-projection fusion frameworks combining perspectives, cubemaps, or equirectangular images have been proposed to improve estimation in panoramic images: [92, 93] for semantic segmentation, [94, 95, 96] for monocular depth, and [97, 98] for optical flow. These methods show promising results but require complex and time-consuming training on spherical datasets.

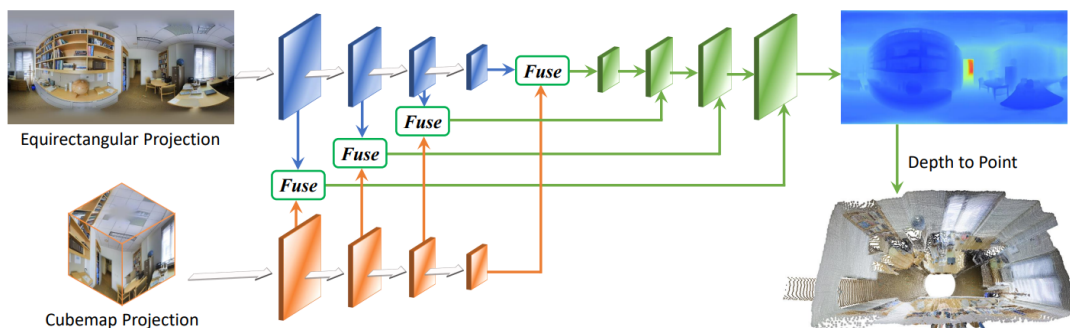


Figure 2.16: UniFuse network: [95]. The model extracts features from equirectangular and cubemap images and merges the contribution to estimate the depth in the observed scene.

2.3.4 Deformable Convolutions

Another approach proposes to replace the uniform perspective convolution operations with distortion-aware convolutions. Technically, the shape of each convolution kernel is modified according to some criteria: its latitude in the case of an equirectangular adaptation. The original deformable convolution was presented by [99] and was initially used to improve perspective image processing. The authors proposed to learn additional offsets in an end-to-end manner to precisely match each region of interest used by the network to the observed scene, as illustrated in Figure 2.17. More recent works have extended this idea by using fixed offsets. For example, in [100, 101, 102, 103], the authors show that depth prior can be used to compute the adaptive kernel statically, leading to better awareness of the geometry.

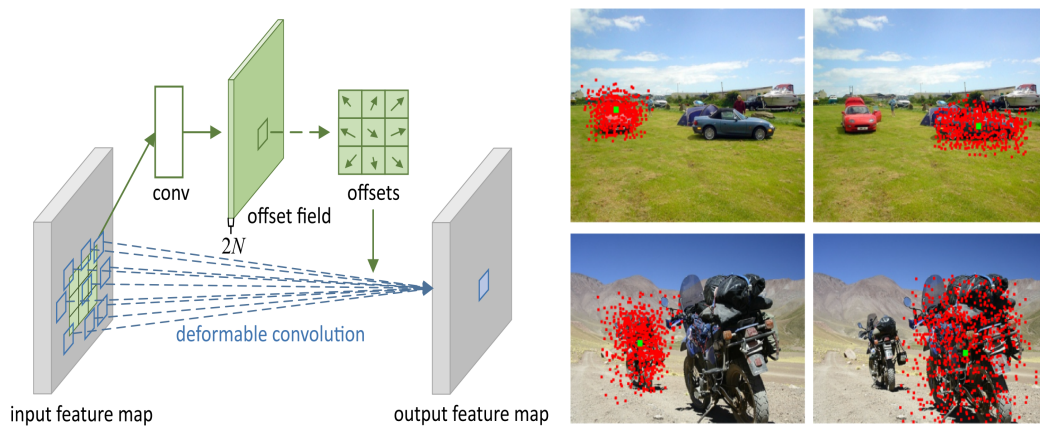


Figure 2.17: Left: Illustration of 3 deformable convolutions [99]. Offsets are learned during training to adapt to various transformations for scale, aspect ratio, and rotation. Right: Each image shows the sampling locations of deformable filters for activation units on a small object and a large one.

In deformable convolutions, the regular grid \mathcal{R} used in classical convolution is augmented with offsets $\Delta\mathbf{p}_n$. The resulting sampling is irregular and the new positions are $\mathbf{p}_n + \Delta\mathbf{p}_n$. Therefore, the Eq. 2.2 becomes:

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n). \quad (2.7)$$

The first use of the deformable convolutions for spherical adaptation was proposed in [104]. First, they roughly increase the size of the kernels by their latitude to improve object detection. Then, in a second work [105], they learn spherical distortions using trainable offsets in an end-to-end manner. Later works [106, 107] reused this learning method for depth estimation.

Finally, the authors in [85] present the distortion-aware convolution strategy for object detection. They use fixed offsets that fit the exact mathematical formulas of equirectangular projection. Although they demonstrate promising results, they still use additional training on specific omnidirectional datasets despite using fixed offsets. Subsequent publications [87] have followed this strategy for layout recovery.

2.4 Proposed Spherical Adaptation Solution

In this first part of the thesis, we propose to demonstrate the generalization of the distortion-aware convolution strategy to any existing convolutional network independently of the computer vision task. Furthermore, unlike previously cited work using this method, we perform **no additional training** on specific omnidirectional datasets as we keep the pretrained weights on perspective images. This allows us to take advantage of the ongoing development of new, more efficient methods in the perspective domain and transfer them to omnidirectional images, all at a **low implementation and computational cost**.

This section presents the equations for computing spherical offsets based on a local perspective projection onto the sphere. Then, we explain our implementation strategy for any convolutional network.

2.4.1 Local Perspective Projection on the Sphere

Inspired by [87], this section presents the mathematics behind our proposed spherical adaptation solution. The usual regular grid \mathcal{R} is modified to fit the equirectangular distortions. To build a kernel of resolution r and angular size α centered in a location $\mathbf{p}_0 = p_{00} = (u_{00}, v_{00})$ in the equirectangular image, the center coordinates are first transformed to spherical system $p_{s,00} = (\phi_{00}, \theta_{00})$ using

$$\phi_{00} = \left(u_{00} - \frac{W}{2}\right) \frac{2\pi}{W} \quad ; \quad \theta_{00} = -\left(v_{00} - \frac{H}{2}\right) \frac{\pi}{H} \quad , \quad (2.8)$$

where W and H are respectively the width and the height of the equirectangular image in pixels. Each point of the kernel is defined by

$$\hat{p}_{spher,ij} = \begin{bmatrix} \hat{x}_{ij} \\ \hat{y}_{ij} \\ \hat{z}_{ij} \end{bmatrix} = \begin{bmatrix} i \\ j \\ d \end{bmatrix} \quad , \quad (2.9)$$

where i and j are integers in the range $[-\frac{r-1}{2}, \frac{r-1}{2}]$ and d is the distance from the center of the sphere to the kernel grid. In order to cover the field of view α , the distance is set to $d = \frac{r}{2 \tan(\frac{\alpha}{2})}$.

The coordinates of these points are computed by normalizing and rotating them to align the kernel center on the sphere. Therefore:

$$p_{spher,ij} = \begin{bmatrix} x_{ij} \\ y_{ij} \\ z_{ij} \end{bmatrix} = R_y(\phi_{00}) R_x(\theta_{00}) \frac{\hat{p}_{spher,ij}}{|\hat{p}_{spher,ij}|} \quad , \quad (2.10)$$

where $R_a(\beta)$ stands for the rotation matrix of an angle β around the a axis.

These coordinates are transformed to latitude and longitude in the spherical domain using

$$\phi_{ij} = \arctan\left(\frac{x_{ij}}{z_{ij}}\right) \quad ; \quad \theta_{ij} = \arcsin(y_{ij}) \quad ; \quad (2.11)$$

and finally back-projected to the original 2D equirectangular image

$$u_{ij} = \left(\frac{\phi_{ij}}{2\pi} + \frac{1}{2}\right) W \quad ; \quad v_{ij} = \left(-\frac{\theta_{ij}}{\pi} + \frac{1}{2}\right) H \quad . \quad (2.12)$$

In Figure 2.18, some example of kernels at different latitude and longitude are presented. The blue point defines the center of the kernel \mathbf{p}_0 . The red points are the positions of the elements $p_{ij} = (u_{ij}, v_{ij})$ in the adapted kernel, defined as previously. The green points are the positions of elements in a standard perspective kernel given by:

$$u_{persp,ij} = u_{00} + ir \quad ; \quad v_{persp,ij} = v_{00} + jr \quad . \quad (2.13)$$

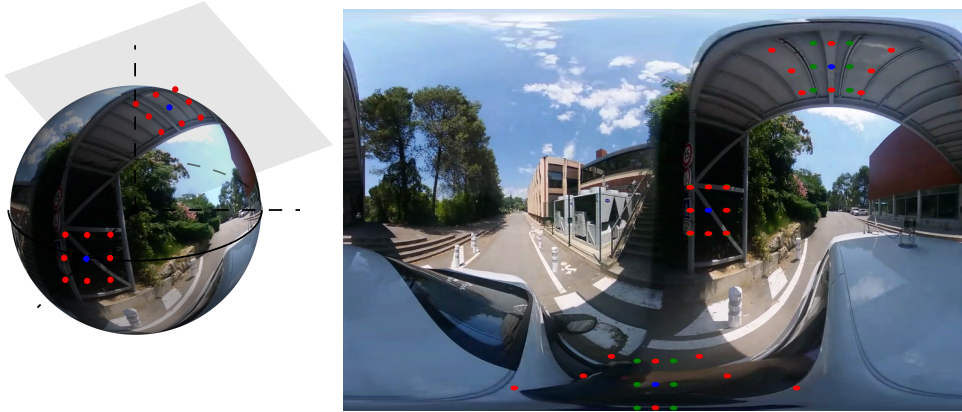


Figure 2.18: The equirectangular image presents significant distortions in the polar regions. Convolution kernel shapes are modified according to their latitude. **Blue**: kernel center, **Green**: perspective kernel, **Red**: adapted equirectangular kernel.

2.4.2 Implementation in any Convolutional Network

The distortion-aware convolution strategy use fixed offsets. As a result, there is no additional parameter learning required. However, in previous works using such strategy [85, 87], they performed a fine-tuning on spherical datasets. But as we already demonstrated in Section 2.3.1, omnidirectional datasets are often limited in terms of visual modalities or diversity of scenes. Therefore, we propose here **to avoid any additional training** by directly reusing pretrained weights on perspective images. Figure 2.19 presents a schema of the general implementation process.

The overall architecture and weights of the network are directly derived from a model trained in a supervised manner using perspective images and ground truth modalities. For all networks considered in this study, we reuse the pretrained weights provided by the authors of the models. This highlights the simplicity of integrating our solution into previously published work and ensures good performance fidelity to the original publication.

In parallel, **we replace the standard convolution layers with new layers handling equirectangular distortions**. In practice, the convolution operations are modified to add fixed offsets to each coordinate of the kernel points. These offsets are calculated using the Eq. 2.12. Moreover, since this equation only requires the dimensions of the input features map and convolution parameters, we can compute them offline. It guarantees **no slowdown during inference**: the computation time of the initial perspective network and the adapted version are identical.

Besides, the proposed solution is **compatible with any kernel’s size, stride, or padding**. Therefore, we can implement our proposed spherical adaptation in any convolutional neural network architecture.

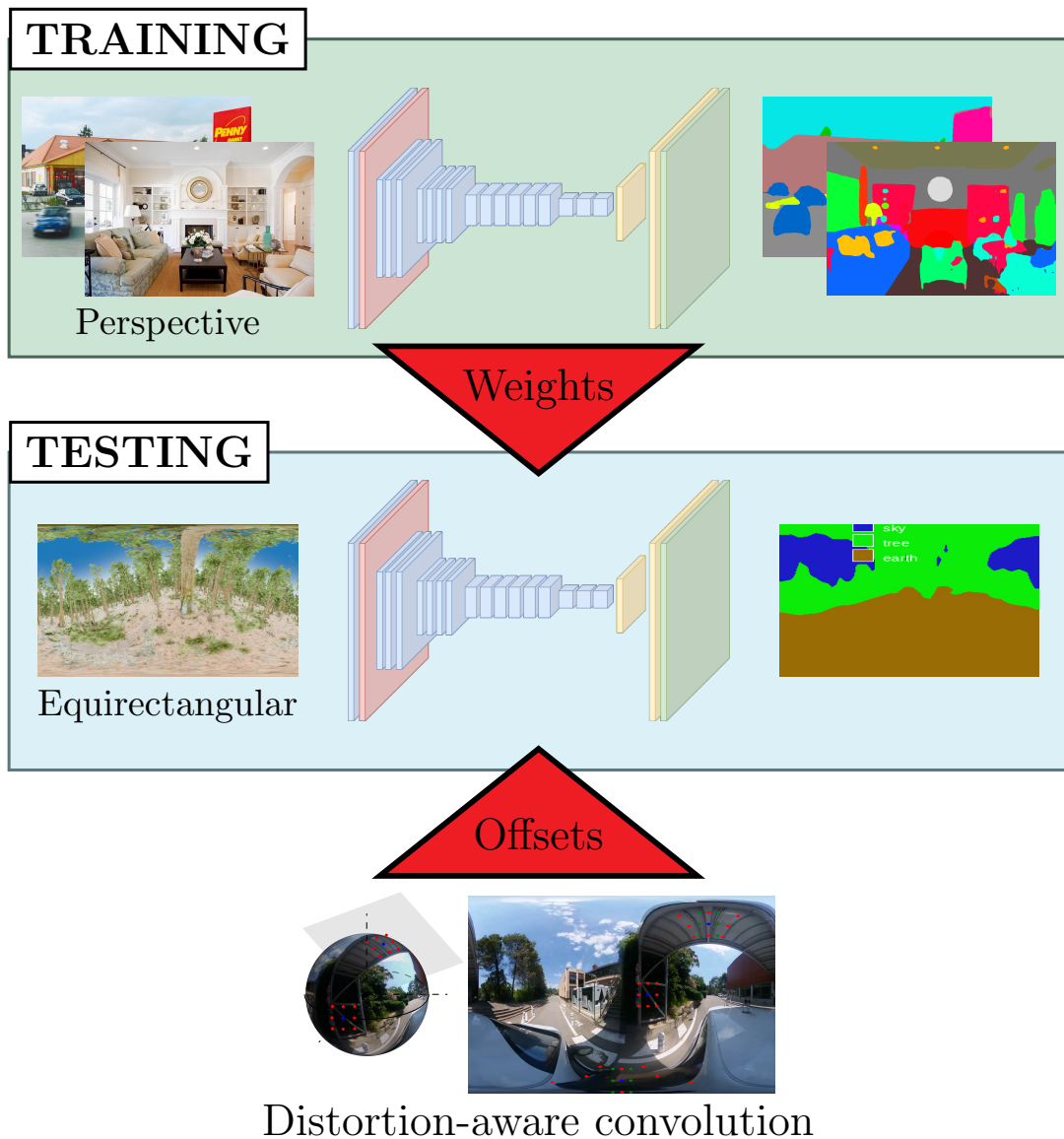


Figure 2.19: One significant advantage of our proposed method is its **ease of implementation** in any existing convolutional network pre-trained with perspective images. Moreover, **no additional training** on omnidirectional datasets is required. At test time, the weights are directly transferred to the same architecture with distortion-aware convolutional filters to process equirectangular images. We compute these spherical offsets offline to **avoid computational slowdowns**. Although this figure illustrates the case of the semantic segmentation task, we apply the same strategy for monocular depth and optical flow estimation.

Chapter 3

Evaluation on Omnidirectional Images

This section presents the generalization of our proposed spherical adaptation to common computer vision tasks. As explained in Section 2.2.2, we mainly focused on semantic segmentation, depth, and optic flow estimation. These tasks provide dense, crucial, and complementary information for pixel-level scene understanding. Moreover, each visual modality has different technical needs, which test the robustness of our approach in a wide variety of contexts. This chapter presents the comparison between the adapted network and its baseline for each modality on several validation datasets. We perform extensive analysis on virtual datasets. In addition, we also provide tests on real omnidirectional images with different backgrounds and lighting.

3.1 Evaluation Datasets

Outdoor scenes are generally more challenging for networks than indoor scenes, mainly due to the diversity of lighting and the limited number of outdoor images in the training datasets. However, the available outdoor omnidirectional datasets are very limited. To our knowledge, no published outdoor dataset aggregates the omnidirectional ground truth for all visual modalities considered. In order to effectively compare the output of the adapted network with its baseline and to accurately measure the contributions of our adaptation, we created our own outdoor datasets.

We use virtual, dense and unstructured photorealistic forest environments to evaluate our proposed spherical adaptation solution. For example, low-altitude images captured between closely spaced tree trunks provide good density and context variations in equirectangular images. In addition, forest scenes are often not used in perspective training datasets, which will further challenge the tested models pretrained on more urban datasets.

Nevertheless, photorealistic rendering engines are often very complex. They usually cannot provide access to all visual modalities, especially when considering omnidirectional content. Therefore, we use different datasets to separately test semantic segmentation and depth on one hand and optical flow on the other.

3.1.1 Semantic Segmentation and Depth Evaluation

RWFOREST Dataset: We create a photorealistic forest environment with complex lighting and dense foliage using the best rendering capabilities of Unreal Engine [108] and forest textures from its marketplace [109]: RWFOREST. For all captured RGB images, we associate a semantic segmentation and a ground truth depth provided by the AIRSIM plugin [10]. For the semantic segmentation, three classes were distinguished: trees, ground, and sky. For depth, we limit the depth values to 100 meters because most state-of-the-art algorithms can only estimate a few dozen meters.

Several versions of our dataset are proposed with different image sizes (*width* \times *height*): 256×256 , 512×256 , and 1024×512 . This allows us to test our convolution adaptation with different input resolutions. Figure 3.1 shows a Sample of the dataset RWFOREST.

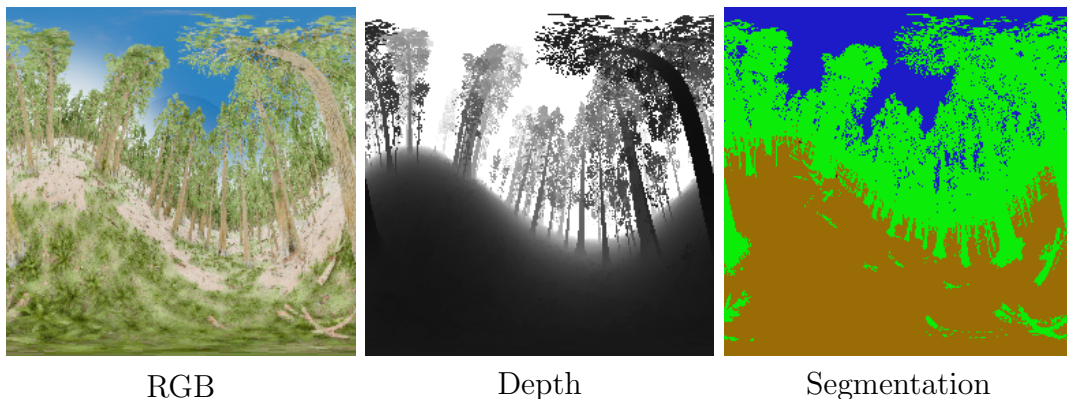


Figure 3.1: RWFOREST dataset. Here is the 256×256 resolution version.

3.1.2 Optical Flow Evaluation

Ground truth optical flow is complex to compute, especially when considering a dense omnidirectional context. In [6], we published OmniFlowNet the first open-source optical flow omnidirectional dataset. Since then, several new datasets have been proposed: Flow360 [110], CityScene, and EquirectFlyingThings [98].

OmniFlowNet Dataset: Inspired by the creation of the Sintel dataset [66], we use Blender [111] to create the OmniFlowNet dataset. This tool is a free and open-source 3D computer graphics software for making animated films, visual effects, 3D printed models, interactive 3D applications, and video games. Three different scenes called CartoonTree, Forest, and LoyPolyModel are generated with free 3D models available online (Figure 3.2). An equirectangular camera, simulated in Blender, moves in these fixed scenes with different orientations given by Euler angles (roll ϕ , pitch θ , yaw ψ). The sets are shown in TABLE 3.1.

Table 3.1: Different camera’s orientation given as Euler Angles (ϕ, θ, ψ) .

Case	CartoonTree	Forest	LowPolyModel
1	$(-\frac{3\pi}{4}, \pi, 0)$	$(-\frac{3\pi}{4}, \pi, -\frac{\pi}{2})$	$(\frac{\pi}{4}, 0, 0)$
2	$(-\frac{\pi}{4}, \pi, 0)$	$(-\frac{\pi}{2}, \pi, -\frac{\pi}{2})$	$(\frac{3\pi}{4}, 0, 0)$
3	$(-\frac{\pi}{2}, \frac{3\pi}{4}, 0)$	$(-\frac{\pi}{2}, \frac{3\pi}{4}, -\frac{\pi}{2})$	$(\frac{\pi}{2}, -\frac{\pi}{4}, 0)$
4	$(-\frac{\pi}{2}, \frac{5\pi}{4}, 0)$	$(-\frac{\pi}{2}, \frac{5\pi}{4}, -\frac{\pi}{2})$	$(\frac{\pi}{2}, \frac{\pi}{4}, 0)$

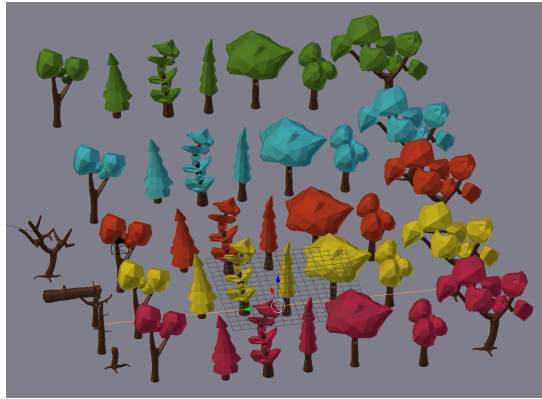


Figure 3.2: 3D models used to generate the Forest dataset.

To extract the ground truth optical flow, we used the *Vector Pass* given by the Blender Cycles Renderer, as presented in [112]. This render pass is usually helpful in producing motion blur by providing the motion of every pixel in the image. Here, the *Vector Pass* returns the pixel displacement in the horizontal and vertical directions perpendicular to the camera axis, the ground truth optical flow. We create 1200 equirectangular RGB images with associated ground truth omnidirectional optical flow. The image resolution is 768×384 to maintain a width-to-height ratio of 2.

Flow360 Dataset: This dataset published in [110] proposes several urban driving scenes during different times of day or weather. This dataset provides the ground truth omnidirectional optical flow associated with RGB image sequences.

CityScene and EquirectFlyingThings (EFT) Dataset: The authors in [98] proposed two new omnidirectional optical flow datasets. The first one CityScene gathers 2000 images taken during urban driving in the virtual environment CARLA [113]. The second one EquirectFlyingThings was inspired by the FlyingThings3D perspective dataset [74] and contains 2000 images.

Figure 3.3 presents an example of images from all above-mentioned cited optical flow datasets.

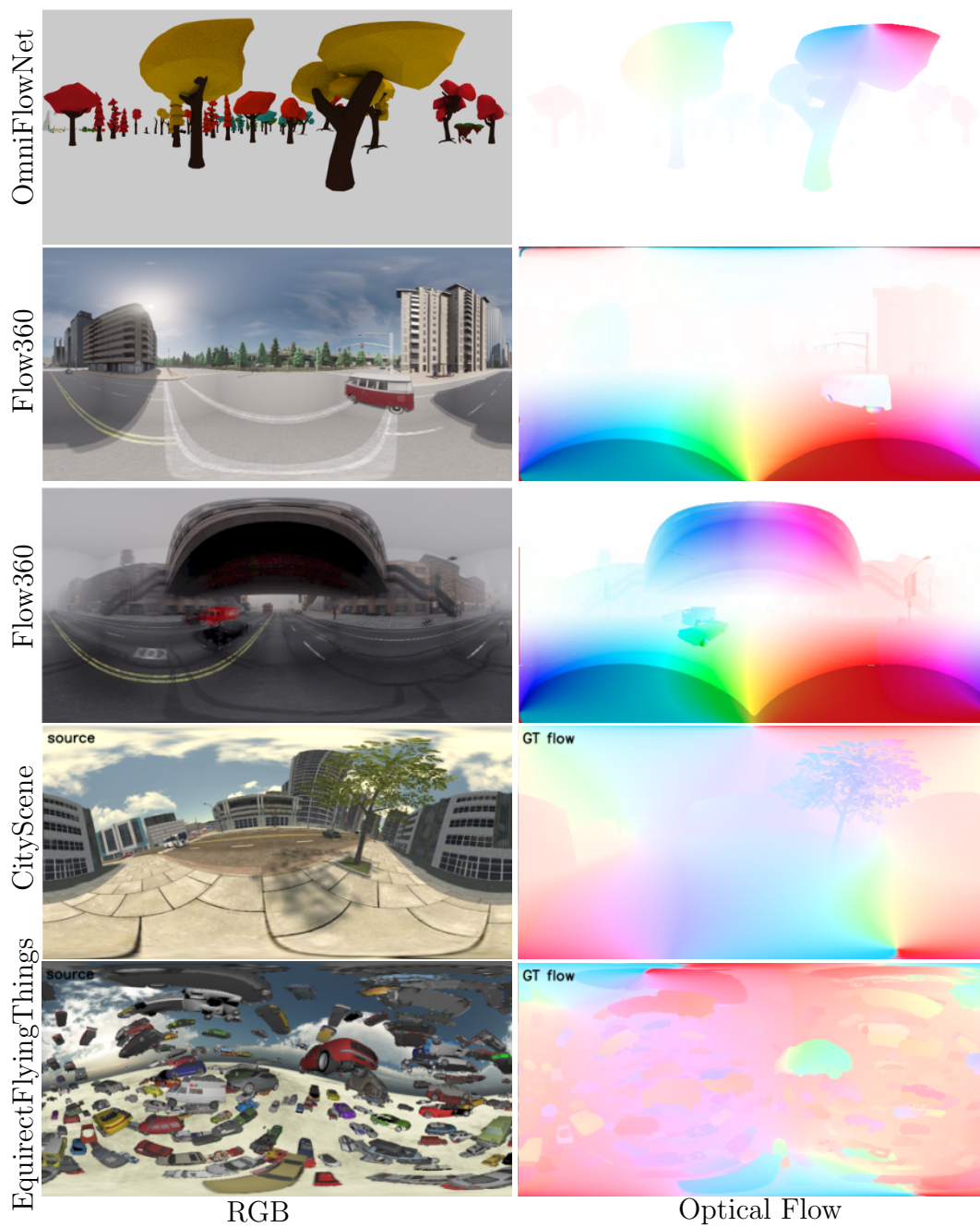


Figure 3.3: OmniFlowNet [6], Flow360 [110], CityScene and EquirectFlyingThings [98] datasets.

3.1.3 Additional Real-World Test Scenarios

To test our adaptation even further, we test it in various real-world scenarios. We capture different outdoor and indoor scenes using a Ricoh Theta Z1, a polydioptric camera that combines two fisheye lenses to reconstruct a full omnidirectional image, as explained in Section 2.1.4. Once acquired, the equirectangular videos are reconstructed from the fisheye inputs using the Ricoh Theta Movie Converter application [114]. Then, the videos are cut into multiple frames using FFmpeg [115]. Since omnidirectional sensors capable of extracting ground truth are rare, complex to calibrate, and often prone to reconstruction errors, we focus on RGB with easily interpretable content. It allows us to validate our approach and to test in real conditions its robustness in images never observed during training.

We study two different contexts. First, we focus on moving objects in front of a fixed camera. For example, in the Ball1 and Ball2 scenes, a ball is thrown in front of an omnidirectional camera or rotates around its north polar region. Second, we also use urban driving images where a camera moves in an essentially rigid background. For example, in the Car1 and Car2 scenes, a car either passes under large infrastructures or near trees. Figure 3.4 presents Samples of these various scenes.

All these situations provide a good variety of content for semantic segmentation, depth, and optical flow estimation.



Figure 3.4: Images captured with an omnidirectional camera during different moving scenes.

3.2 Adapted and Baseline Models Comparison

This section compares the networks using our proposed spherical adaptation with their baseline versions. In all cases, no additional training was performed to fine-tune the networks. Therefore, all of the following input images were never seen during training. First, we perform a quantitative analysis of all visual modalities on the virtual datasets described in the section above. Then, we focus more specifically on some of these results and the previously presented real-world test scenarios.

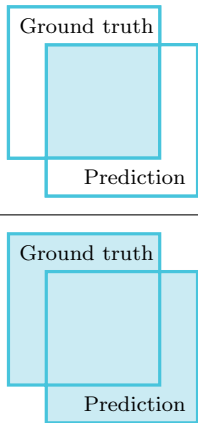
3.2.1 Semantic Segmentation Comparison

First, we analyze the contributions of the distortion-aware convolution strategy to transfer perspective-trained semantic segmentation networks to omnidirectional images. To quantitatively measure the estimation performance, we use two common metrics from [41]. In addition, we propose a third one that provides additional comparison content. Let's consider a dataset with N_i images and a network trained to predict N_c classes, we define the metrics as:

- The Mean Intersection Over Union (MIoU) [41]: indicates the intersection over union between predicted and ground truth pixels, averaged over all the classes:

$$\text{MIoU} = \frac{1}{N_i N_c} \sum_{N_i, N_c} \text{IoU}(\text{image}, \text{class}), \quad (3.1)$$

with for each image and class:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{\text{Diagram 1}}{\text{Diagram 2}}; \quad (3.2)$$


- The Mean Accuracy [41]: indicates the proportion of correctly classified pixels averaged over all the classes.

$$\text{MA} = \frac{1}{N_i} \sum_{N_i} \frac{\text{Correct predictions}}{\text{Total of predictions}} = \frac{1}{N_i} \sum_{N_i} \frac{TP}{TP + FP + FN}, \quad (3.3)$$

with TP the true positives, TN the true negatives, FP the false positives, and FN the false negatives;

- The Averaged Erroneous Class Estimate (AECE) [7]: indicates the number of classes detected by the network but not present in the ground truth image averaged over all runs.

$$\text{AECE} = \frac{1}{N_i} \sum_{N_i} \text{Predicted classes not present in ground truth.} \quad (3.4)$$

Table 3.2 compares the metrics between the spherically adapted network and its baseline version on the RWFOREST dataset. Three different image resolutions are tested to test various convolutional kernel adaptations: 256×256 , 512×256 , and 1024×512 . The network considered here has a classical encoder-decoder architecture using ResNet50 [21] as the encoder and PPM [39] as the decoder, as illustrated in Figure 3.5. We use the network pretrained version on the perspective ADE20K dataset [45].

Table 3.2: Comparison of spherically adapted and baseline semantic segmentation networks on RWFOREST 256×256 , 512×256 , and 1024×512 . Each dataset contains 1000 images.

RWFOREST 256×256	MIoU (\uparrow)	Accuracy (\uparrow)	AECE (\downarrow)
Semantic segmentation baseline	0.677	0.810	2.045
Semantic segmentation adapted	0.688 (+1.525%)	0.828 (+2.282%)	0.337
RWFOREST 512×256	MIoU (\uparrow)	Accuracy (\uparrow)	AECE (\downarrow)
Semantic segmentation baseline	0.504	0.631	3.692
Semantic segmentation adapted	0.564 (+11.980%)	0.639 (+1.332%)	1.577
RWFOREST 1024×512	MIoU (\uparrow)	Accuracy (\uparrow)	AECE (\downarrow)
Semantic segmentation baseline	0.443	0.621	5.845
Semantic segmentation adapted	0.528 (+19.174%)	0.627 (+0.852%)	3.034

For each comparison, the best results are in **bold**.

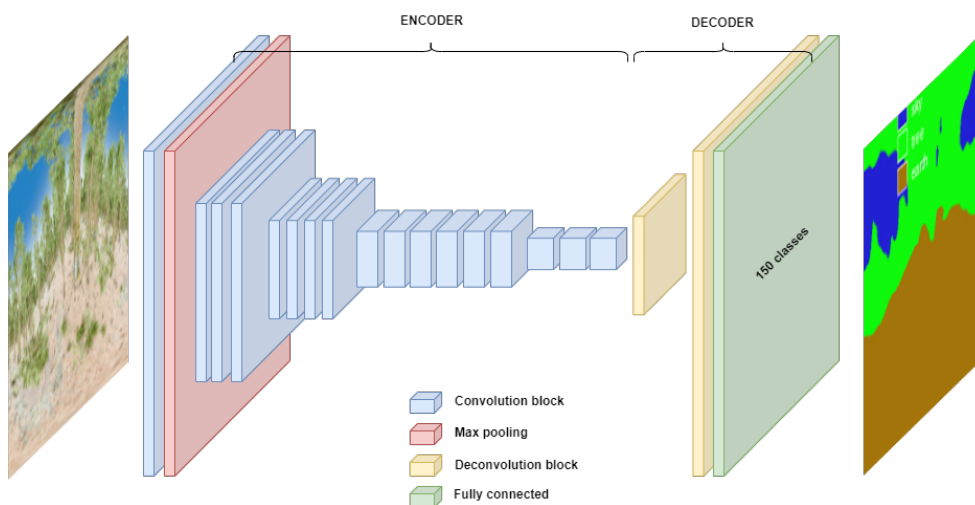


Figure 3.5: Semantic segmentation architecture used in this thesis proposed in [46] (encoder: ResNet50 [21], decoder: Pyramid Pooling Module [39]).

Looking at the metrics, we conclude that distortion-aware convolution improves the semantic segmentation task. The MIoU and accuracy metrics are higher for the spherically adapted model, which are the most significant markers of better segmentation. In addition, the AECE is significantly lower: the adapted network detects fewer erroneous classes.

To better illustrate these results, we provide an in-depth analysis of two Samples of the RWFOREST dataset in Figure 3.6. Two improvements are noticeable and correspond well with the previous conclusions on quantitative comparison: better pattern matching in the polar regions and less erroneous class estimation.

First, the spherical adaptation helps the network to take into account equirectangular distortions. The detection of shapes and patterns is improved in highly distorted regions thanks to a better local coherence of the pixels. This effect is visible in Sample 1, where the adapted network better identifies the tree canopy (upper polar region of the image).

In addition, the adaptation also reduces the number of noisy predictions. Some objects in the equirectangular images are highly distorted, resulting in false class predictions by the network. In Sample 2, the upper polar region of the adapted version is less noisy and contains almost no wrong predictions.

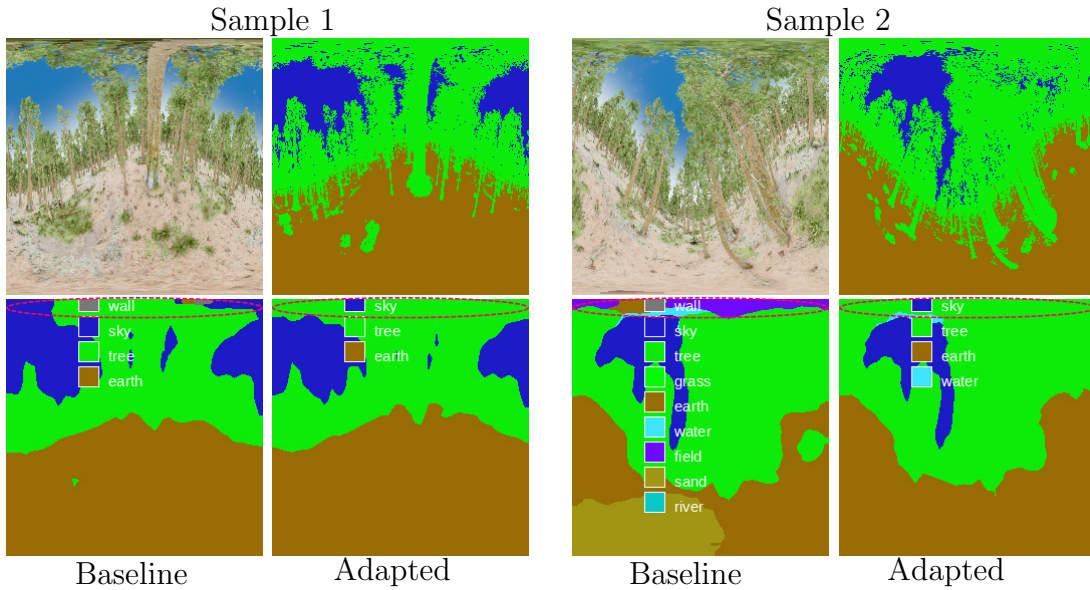


Figure 3.6: Prediction examples in the RWFOREST dataset. The spherical adaptation improves shape detection (tree canopy is better identified) and reduces erroneous class estimation. (**Top left**): RGB input, (**top right**): ground truth segmentation, (**bottom left**): prediction from the baseline network, (**bottom right**): prediction from the adapted network.

We observe the same findings when estimating semantic segmentation in real urban driving scenes. In our proposed example, Figure 3.7, we captured images when the car was passing under trees in order to focus on the tree canopy detection. The semantic segmentation predicted by the adapted network is more accurate than the baseline estimate, with better tree canopy identification and less noisy class predictions. This visually confirms that distortion-aware convolutions improve the semantic segmentation in virtual and real outdoor images.

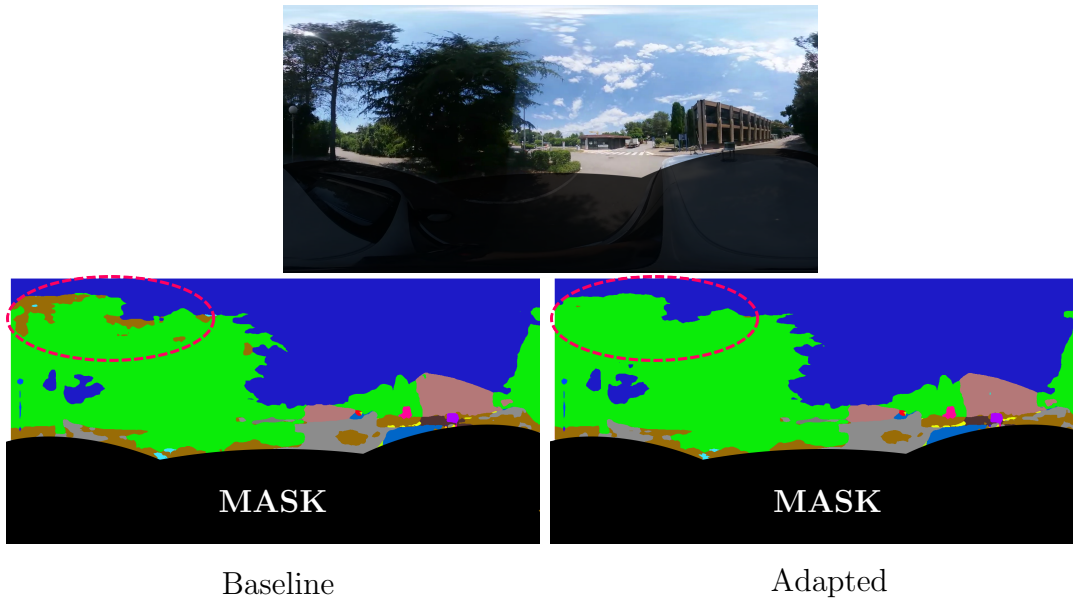


Figure 3.7: Urban driving example (Car2). The adapted network better identifies the tree canopy. A red circle at the top left of the image highlights the area with the most visible differences: the baseline network estimates the earth (in brown) class instead of trees (in green).

A mask is added to the image’s lower part to hide the car’s semantic segmentation estimate. The car’s shape is strongly distorted due to its proximity to the omnidirectional camera. The absence of such images and nearby objects in the training dataset makes the network unable to make a correct prediction. Spherical adaptation improves the quality of semantic segmentation in spherical images but remains limited by the training dataset, as in all supervised methods.

3.2.2 Monocular Depth Comparison

The previous section showed that distortion-aware convolutions improve semantic segmentation. Here, we focus on depth estimation by comparing a spherically adapted depth estimation network and its baseline version. We use three common metrics to evaluate depth estimation algorithms as defined in [116]. Let's consider a dataset with N_i images of N_p pixels, \hat{d}_p the estimated depth for a pixel p , and d_p the ground truth depth. The resulting proposed metrics are:

- The Accuracy under a threshold th :

$$\delta = \max\left(\frac{\hat{d}_p}{d_p}, \frac{d_p}{\hat{d}_p}\right) < th, \quad (3.5)$$

generally three thresholds values are used: $th = 1.25, 1.25^2$, and 1.25^3 ;

- The Absolute Relative Error (AbsRel):

$$\text{AbsRel} = \frac{1}{N_i N_p} \sum_{N_i, N_p} \frac{|d_p - \hat{d}_p|}{d_p}; \quad (3.6)$$

- The linear Root Mean Square Error (RMSE):

$$\text{RMSE} = \frac{1}{N_i} \sum_{N_i} \sqrt{\frac{1}{N_p} \sum_{N_p} (d_p - \hat{d}_p)^2}. \quad (3.7)$$

Similarly to the semantic segmentation evaluation, we compare in Table 3.3 the spherically adapted depth estimation network and its baseline version on the RWFOREST dataset. The network used here is MIDAS, a very lightweight encoder-decoder architecture, illustrated in Figure 3.8.

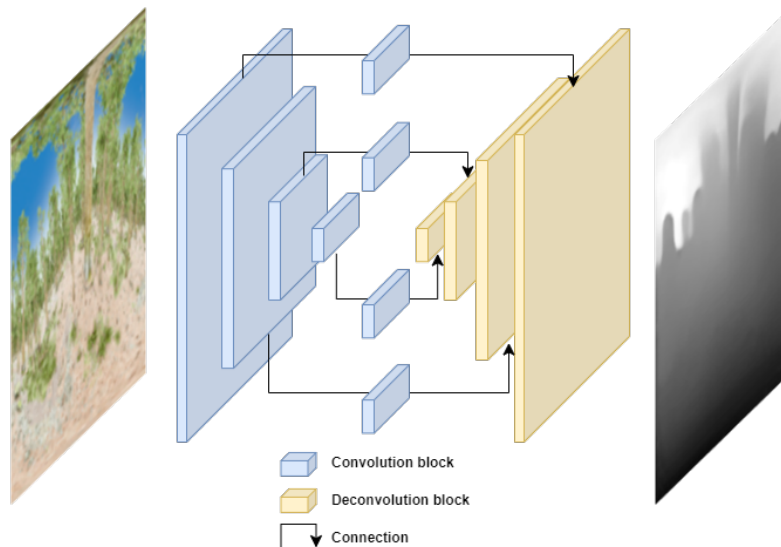


Figure 3.8: MIDAS lightest network [60].

Table 3.3: Comparison of adapted and baseline depth estimation networks on RWFOREST 256×256, 512×256, and 1024×512. Each dataset contains 1000 images.

RWFOREST 256×256	$\delta < 1.25$ (\uparrow)	$\delta < 1.25^2$ (\uparrow)	$\delta < 1.25^3$ (\uparrow)	AbsRel (\downarrow)	RMSE (\downarrow)
Monocular depth baseline	0.251	0.440	0.619	1.198	0.277
Monocular depth adapted	0.26 (+3.6%)	0.454 (+3.2%)	0.630 (+1.8%)	1.154 (-3.7%)	0.275 (-0.7%)
RWFOREST 512×256	$\delta < 1.25$ (\uparrow)	$\delta < 1.25^2$ (\uparrow)	$\delta < 1.25^3$ (\uparrow)	AbsRel (\downarrow)	RMSE (\downarrow)
Monocular depth baseline	0.254	0.421	0.554	1.664	0.305
Monocular depth adapted	0.264 (+3.9%)	0.437 (+3.8%)	0.574 (+3.6%)	1.567 (-5.8%)	0.298 (-2.3%)
RWFOREST 1024×512	$\delta < 1.25$ (\uparrow)	$\delta < 1.25^2$ (\uparrow)	$\delta < 1.25^3$ (\uparrow)	AbsRel (\downarrow)	RMSE (\downarrow)
Monocular depth baseline	0.248	0.413	0.545	1.836	0.316
Monocular depth adapted	0.257 (+3.6%)	0.431 (+4.4%)	0.57 (+4.6%)	1.7 (-7.0%)	0.307 (-2.8%)

For each comparison, the best results are in **bold**.

The adapted method performs better than the baseline version for all considered metrics and image resolutions. The depth’s accuracy is improved in omnidirectional images thanks to distortion-aware convolutions.

Visual analysis of depth estimation is more challenging than in the case of semantic segmentation. Especially, depth prediction images are more difficult to comment on because depth contrasts are less visible to the human eye. As a result, the visual differences seem ambiguous, and it is complex to decide which estimate is better than the other. We illustrated this in Figure 3.9 and Table 3.4.

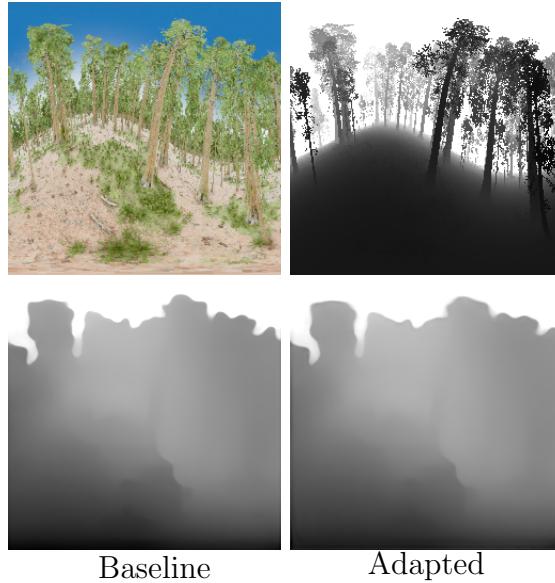


Figure 3.9: Prediction examples in the RWFOREST dataset. The predicted depth images are visually challenging to compare. However, quantitative measurements have shown that the adapted version is numerically better than the baseline. (**Top left**): RGB input, (**top right**): ground truth monocular depth, (**bottom left**): prediction from the baseline network, (**bottom right**): prediction from the adapted network.

Table 3.4: Metrics value for the depth estimations in Figure 3.9.

	$\delta < 1.25$ (\uparrow)	$\delta < 1.25^2$ (\uparrow)	$\delta < 1.25^3$ (\uparrow)	AbsRel (\downarrow)	RMSE (\downarrow)
Baseline	0.183	0.321	0.453	1.374	0.293
Adapted	0.253 (+38.8%)	0.382 (+18.8%)	0.491 (+8.4%)	1.403 (+2.2%)	0.277 (−5.5%)

For each comparison, the best results are in **bold**.

Nevertheless, we can see significant visual improvements in some real outdoor omnidirectional images. Figure 3.10 shows two image acquisitions, the first as the car passes under a bridge and the second as it drives by a large tree. Similarly to the results on semantic segmentation, the detection of patterns is improved in the polar regions, and there is less erroneous depth estimation. Sample 1 shows

that the spherical adaptation improves the depth prediction in the polar regions of the equirectangular images. In the upper left of the image, the bridge depth estimation is more accurate and smoother due to better local pixel coherence.

In addition, Sample 2 shows that the adapted prediction is less sensitive to illumination noise. The image contrast in the top polar region shows significant differences due to the sun configuration. The baseline network interprets these changes as depth differences, while the adapted model is more robust and remains accurate.

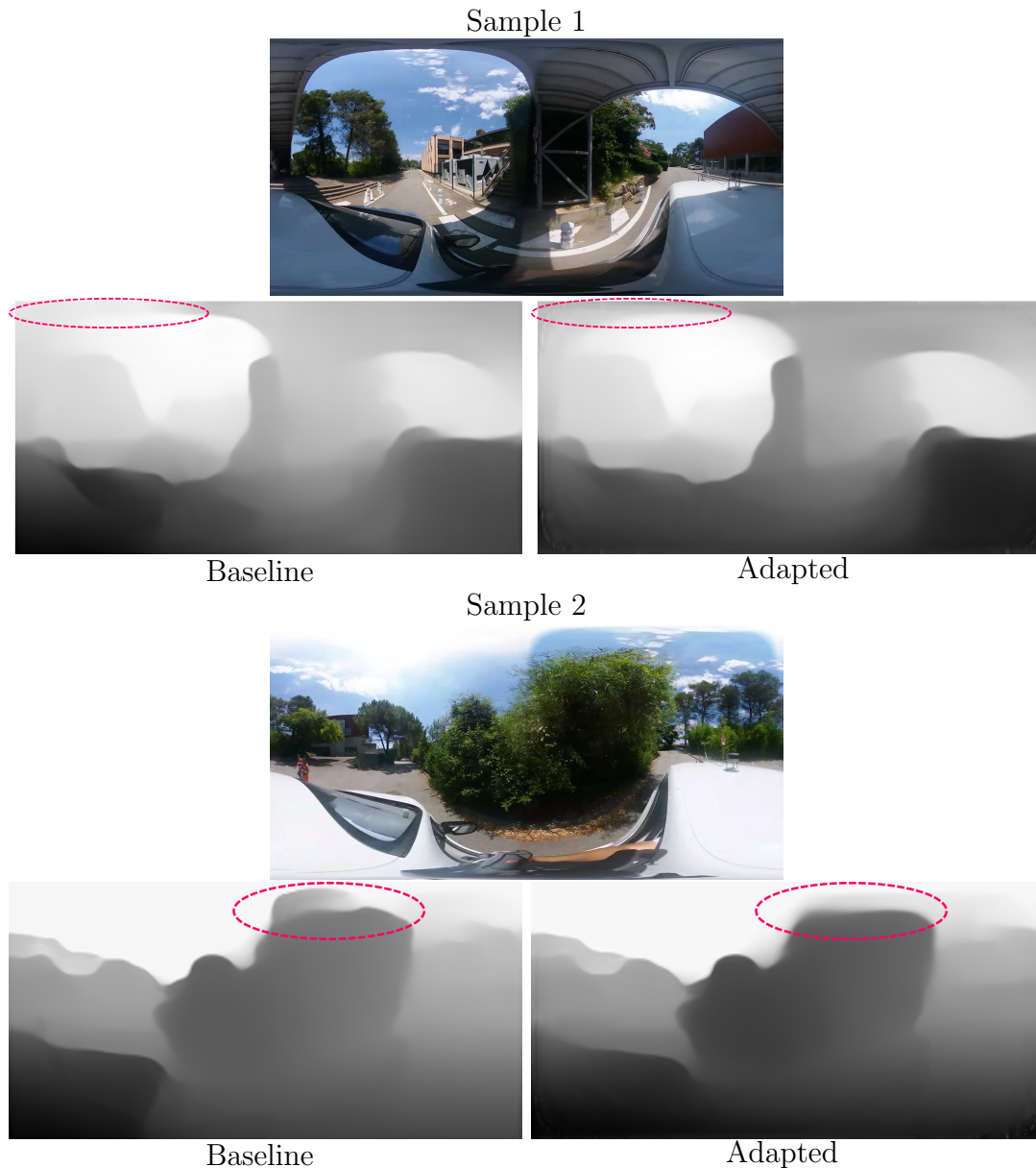


Figure 3.10: Urban driving examples. Red circles at the top of the image highlight the areas with the most visible differences. **Sample 1:** The adapted network better estimates depth in the polar regions of the equirectangular images. **Sample 2:** Less erroneous depth estimation from the adapted network.

3.2.3 Optical Flow Comparison

After having demonstrated in the previous sections the interest in using our spherical adaptation method to transfer perspective semantic segmentation and depth estimation methods to omnidirectional images, we focus here on the optical flow. We use a common metric, the End-Point Error (EPE) [73], to quantify the optical flow estimation accuracy. Let's consider a dataset with N_i images of N_p pixels, (u_f, v_f) the estimated flow, and (u_{fgt}, v_{fgt}) the ground truth:

$$\text{EPE} = \frac{1}{N_i N_p} \sum_{N_i, N_p} \sqrt{(u_{fgt} - u_f)^2 + (v_{fgt} - v_f)^2}. \quad (3.8)$$

We have investigated the advantage of using convolutions for optical flow estimation on two occasions. First, we adapted the LiteFlowNet2 network [71] in a paper published in 2021 [6]. This network presented a standard encoder-decoder architecture and was one of the best-performing networks back then. We specifically used the pretrained version on the perspective Sintel dataset [66]. Then, two years later, in [7], we updated this work using a new and more powerful network: GMFlow [28]. This model uses convolution operations to encode input images into high-level features, which are then processed by Transformers, as shown in Figure 3.11. For this network, we use the version pretrained on Things [74] dataset that replaced Sintel as the new reference dataset.

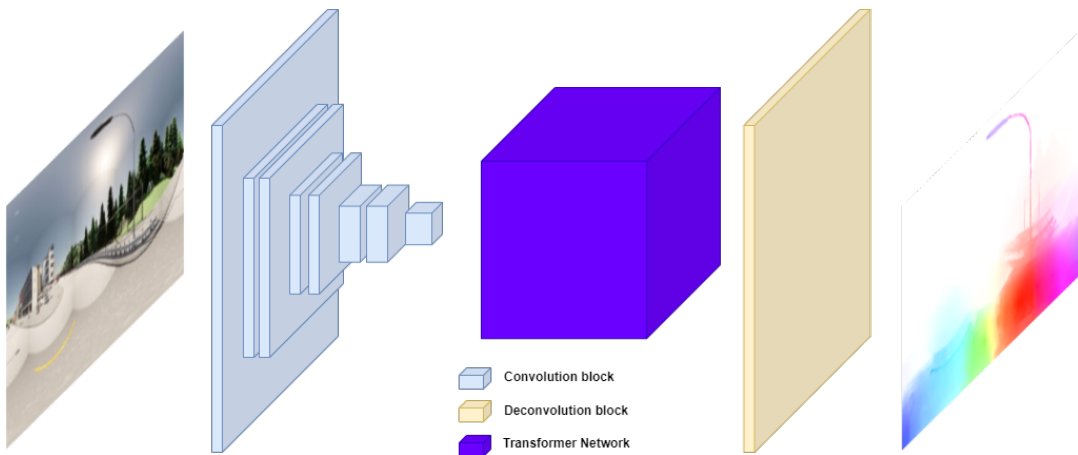


Figure 3.11: GMFlow network [28]. The RAFT [72] convolutional encoder is used to preprocess the image input before the Transformer.

3.2.3.1 First Investigation with LiteFlowNet2 [71]

Table 3.5 presents a quantitative comparison on the OmniFlowNet dataset comparing spherically adapted LiteFlowNet2 and its baseline version. Looking at the metrics, we conclude that the model using distortion-aware convolutions outperforms the baseline. The improved local pixel coherence in the polar regions helps the network better to detect patterns and pixel displacements in these regions.

Table 3.5: Comparison of adapted and baseline LiteFlowNet2 on OmniFlowNet dataset. This dataset gathers 1200 images distributed in three different scenes: CartoonTree, Forest, and LowPolyModel.

	CartoonTree	Forest	LowPolyModel	OmniFlowNet
	EPE (\downarrow)	EPE (\downarrow)	EPE (\downarrow)	EPE (\downarrow)
Optical flow baseline	5.60	10.61	7.66	7.96
Optical flow adapted	4.49 (−19.8%)	9.72 (−8.4%)	7.23 (−5.6%)	7.16 (−10.2%)

For each comparison, the best results are in **bold**.

Figure 3.12 presents the Ball1 and Ball2 motion scenarios. In polar regions, the optical flow estimated by the adapted network is smoother and, all the time, more coherent than the network in perspective, as shown in Sample 1. Whereas the ball motion estimated by the baseline is a shredded mark, the adapted model predicts a complete ball with coherent motion. In Sample 2, the arm moving above the north pole of the camera has a smoother predicted optical flow by the spherical network than the perspective one. The equirectangular convolution helps the network to better understand and calculate motion in highly distorted areas.



Figure 3.12: Ball1 and Ball2 cases. (**Top**) RGB input image , (**bottom left**) optical flow estimation from the baseline network, (**bottom right**) estimation from the adapted network. The adapted network better estimates the optical flow in the top polar region is better. **Sample 1**: The ball is clearly more visible and smoother in that case. **Sample 2**: The arm is smoother.

The south polar region often presents optical flow noise in the spherically adapted estimation. This is related to the artifacts created by the tripod holding the camera. When reconstructing the equirectangular image from the two fish-eye lenses, this highly distorted region certainly brings some noise to the final image. While the baseline network interprets it as white noise, the model with distortion-aware convolution probably detects coherent motion. Figure 3.13 shows that by masking the entire tripod region, the induced noise is reduced.

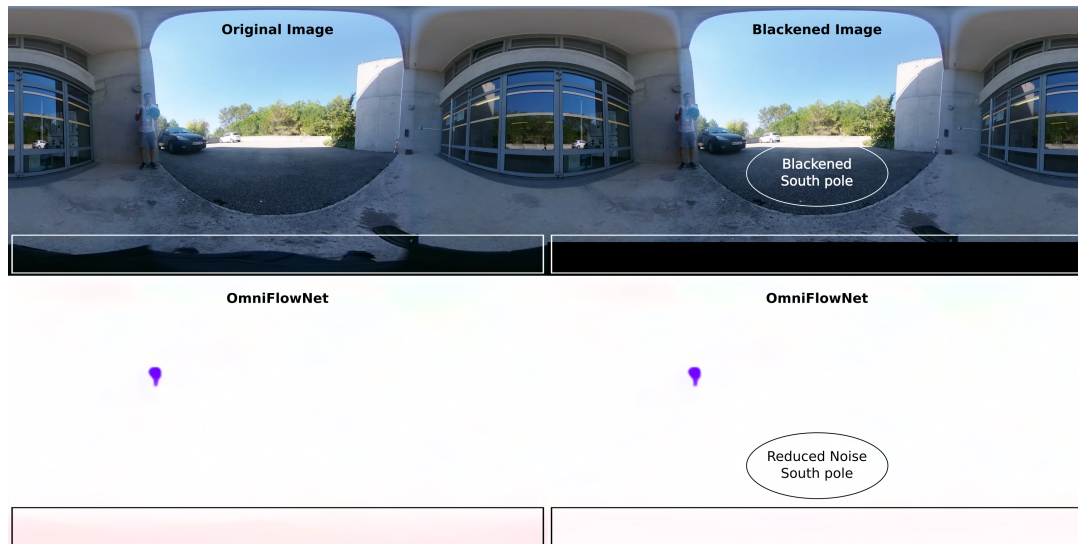


Figure 3.13: Ball 1 case with masked south pole. (**Top left**) original RGB input image, (**top right**) new RGB input image with masked south pole (top-right), (**bottom left**) spherically adapted optical flow estimation of the original image and (**bottom right**) spherically adapted optical flow estimation of the masked image. The optical flow computed with the masked images shows less noise in the south pole. Thus camera tripod and equirectangular reconstruction add noise to the optical flow estimation.

Overall, on several diverse real datasets, the spherically adapted LiteFlowNet2 network outperforms its baseline version in estimating better and smoother optical flow in polar regions and shows the same performance in the equatorial region. Supplementary video results are provided in [117].

3.2.3.2 Second Investigation with GMFlow [28]

Table 3.6 compares the metrics between the spherically adapted network and its baseline version on the four spherical optical flow datasets considered. Similarly to the previous study using LiteFlowNet2 [71], the End-Point Error of the adapted method is smaller than the baseline value for all considered optical flow datasets. The better pixel local coherence improves the pattern detection and the resulting pixel displacement estimation. Nevertheless, the lack of periodicity in the estimated optical flow can explain still high EPE values, especially in the case of the CityScene and EFT datasets. The spherical optical flow is periodic, but the network did not learn this information when learning on perspective datasets. Thus, the estimation of the road flow is still inaccurate. This lack of periodicity remains one of the limitations of this adaptation method for optical flow networks. However, modified convolutions still improve the predictions, especially in the case of single-object flow prediction, as shown in the following qualitative study.

Table 3.6: Comparison of adapted and baseline optical flow networks on OmniFlowNet, Flow360, CityScene, and EquirectFlyingThings (EFT) datasets.

	OmniFlowNet [6] EPE (\downarrow)	Flow360 [110] EPE (\downarrow)	CityScene [98] EPE (\downarrow)	EFT [98] EPE (\downarrow)
Optical flow baseline	5.16	16.15	32.16	42.44
Optical flow adapted	4.96 (-3.93%)	15.95 (-1.27%)	31.36 (-2.08%)	41.83 (-1.43%)

For each comparison, the best results are in **bold**.

Figure 3.14 shows two optical flow estimates from the dataset Flow360. The optical flow enhancements are clearly visible as objects move into the polar regions of the equirectangular image. In both examples, the car passes under a streetlight. Due to the improved local pixel coherence provided by the distortion-aware convolutions, the adapted network is able to track the path of the streetlight in the upper polar region of the image. As a result, the estimated optical flow is close to the ground truth. In parallel, the non-adapted network has difficulty detecting this same streetlight. Consequently, the flow prediction is inaccurate in Sample 1 or even empty in Sample 2.

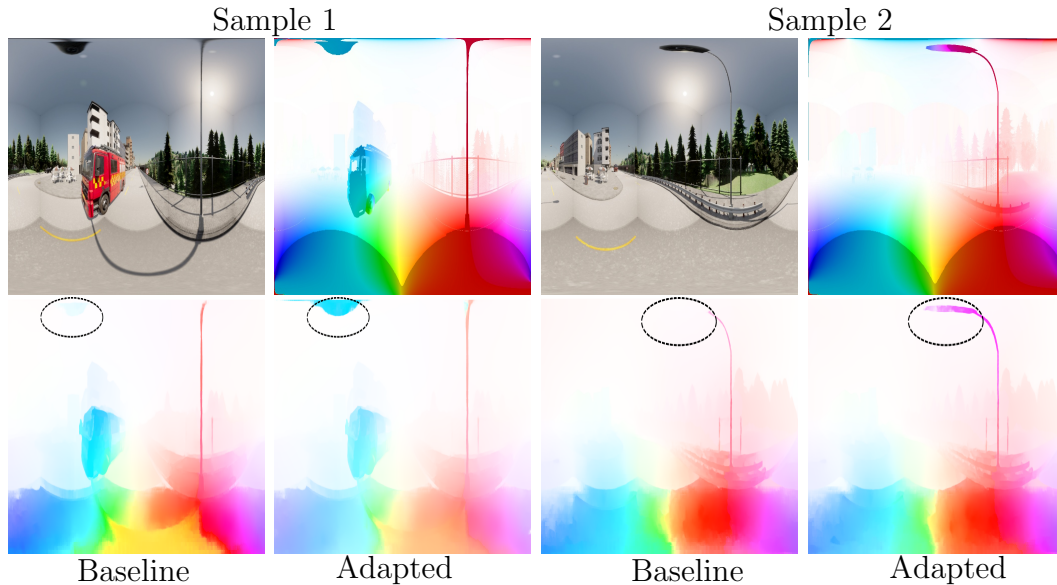


Figure 3.14: Prediction examples in the Flow360 dataset. Spherical adaptation allows better tracking of objects moving in polar regions. As a result, the estimation of the optical flow of the observed lamp post is significantly improved (area highlighted by the red circle). (**Top left**): RGB input, (**top right**): ground truth optical flow, (**bottom left**): prediction from the baseline network, (**bottom right**): prediction from the adapted network.

For optical flow estimation in real images, we focus on the motion of a ball during a throw. Figure 3.15 shows two different image sequences with associated optical flow predictions. Due to better local pixel coherence, the adapted model keeps track of the ball and provides an accurate motion estimate. In contrast, the baseline network loses track of the ball, resulting in a noisy optical flow prediction without an apparent precise motion. This result confirms the improvement in the optical flow estimation in virtual and real images provided by distortion-aware convolutions.

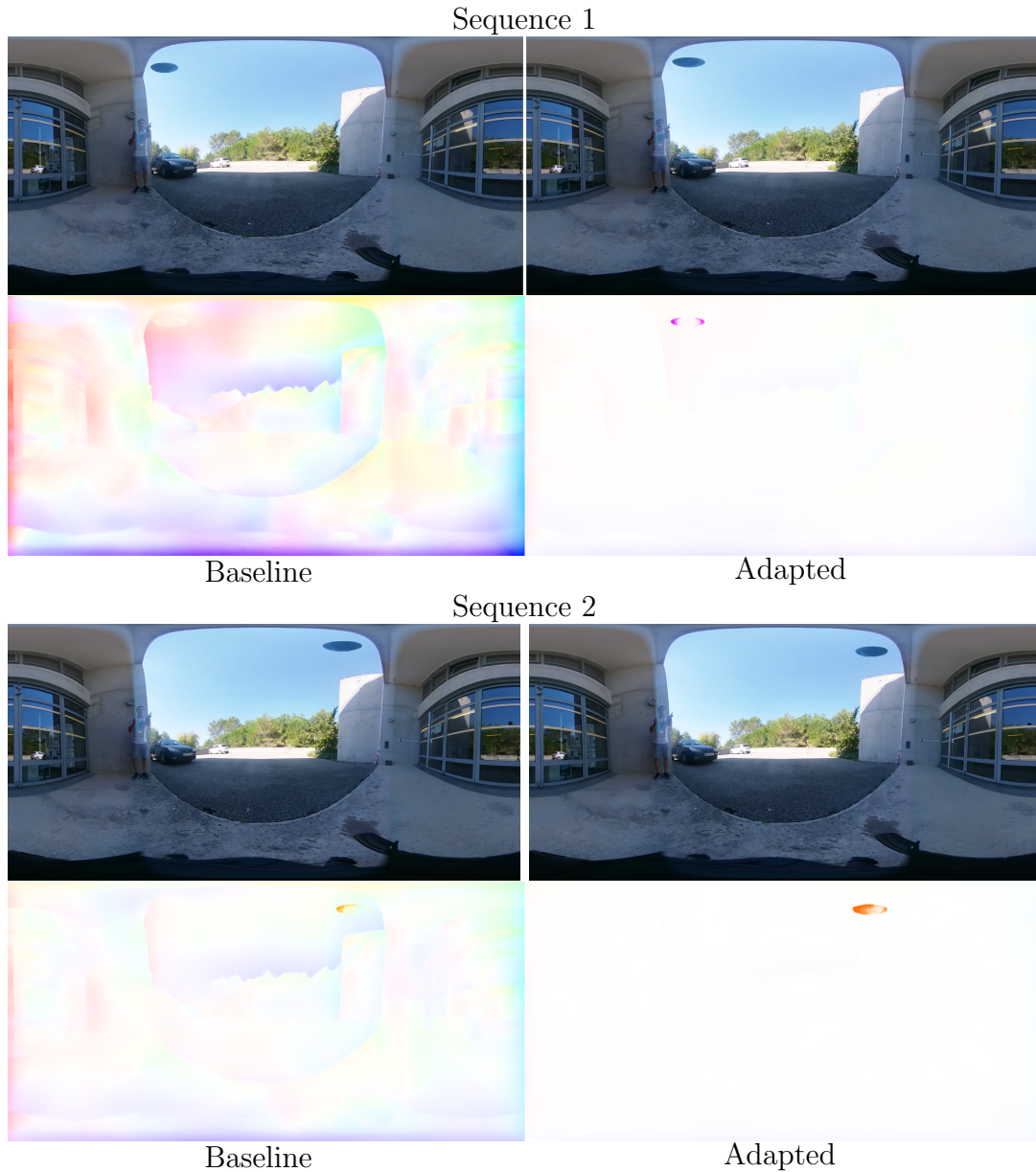


Figure 3.15: Ball1 throw sequences. The adapted network provides correct optical flow estimation, whereas the baseline version loses track of the ball. (**Top left**): RGB input frame at t , (**top right**): RGB input frame at $t + 1$, (**bottom left**): prediction from the baseline network, (**bottom right**): prediction from the adapted network.

3.2.4 Computation Time Comparison

Our proposed adaptation strategy avoids additional training by using fixed offset tables. As mentioned in Section 2.4, these tables can be easily computed offline using convolution parameters and image size. This guarantees no slowdown in code execution, as shown in Table 3.7 below. The computation times of the adapted and baseline networks are very close for all visual modalities. To compute this average, we use the complete datasets, i.e., 1000 images in the case of RWFOREST and 1200 frames in the case of OmniFlowNet and a RTX3060 GPU. Finally, our adaptation method improves estimation performances while keeping the same execution speed, which is a significant advantage.

Table 3.7: Comparison of the running time and model size between spherically adapted and baseline networks. We give the average computation time for processing one omnidirectional RGB image on a RTX3060 GPU. This average is computed using the complete datasets of 1000 images for RWFOREST and 1200 images for OmniFlowNet. Model and offset tables sizes are also provided.

	Computation Time (in ms) (↓)	Model and Offsets Size (in MB)
Semantic segmentation baseline ¹	2.952	206.8
Semantic segmentation adapted ¹	2.963 (+0.4%)	208.0 (+1.2 MB)
Monocular depth baseline ¹	3.802	85.8
Monocular depth adapted ¹	3.797 (−0.1%)	92.0 (+6.2 MB)
Optical flow baseline ²	5.642	18.8
Optical flow adapted ²	5.667 (+0.4%)	19.1 (+0.3 MB)

For each comparison, the best results are in **bold**. ¹ Evaluated on RWFOREST 256×256.

² Evaluated on OmniFlowNet.

Conclusion of Part I

Developing new image processing methods or computer vision applications is always conducted first for perspective images. Spherical image processing comes later and often with significant limitations, as shown by the still limited number of omnidirectional datasets. In this work, we proposed the distortion-aware convolution strategy to easily and quickly adapt any convolutional network pretrained with perspective images to equirectangular content for any computer vision application.

In the previous sections, we tested and proved the generalization of our proposed spherical adaptation solution on three fundamental computer vision tasks: semantic segmentation, monocular depth, and optical flow. We modified a convolutional network for each of these modalities and compared it to its baseline version. In all cases, the spherical adaptation improved performance. Table 3.8 provides a brief review of the previous noteworthy quantitative improvements.

Tested on virtual outdoor images and complex real-world scenarios, the adapted convolution allows us to take into account the significant distortions present in the polar zones of equirectangular images and thus improve the estimation accuracy in these specific areas: the better local coherence of the pixels significantly enhances the quality and smoothness of the estimation and avoids errors.

Although this solution does not compete with networks developed and trained with spherical images, it requires a much lower implementation cost and no additional training. In addition, this allows for rapid transfer to new architectures of any size and shape. All these advantages make this strategy very interesting, especially in the context of image-based navigation of robots with limited computing resources.

Table 3.8: Comparison of adapted and baseline networks on three different visual modalities. The error metric used for semantic segmentation is the complement of the Mean Average of Intersection Over Union, for optical flow is the End-Point Error, and for depth is the Absolute Relative Error.

	Error Metric (\downarrow)
Semantic segmentation baseline ¹	0.323
Semantic segmentation adapted ¹	0.312 (−3.4%)
Monocular depth baseline ¹	1.198
Monocular depth adapted ¹	1.154 (−3.673%)
Optical flow baseline ²	5.16
Optical flow adapted ²	4.96 (−3.93%)

For each comparison, the best results are in **bold**. ¹ Evaluated on RWFOREST 256×256.

² Evaluated on OmniFlowNet.

Part II

Deep Reinforcement Learning Navigation using Omnidirectional Images

Chapter 4

Image-Based Navigation

Autonomous robot navigation refers to the ability of a robot to move through an environment and reach a specific location or follow a particular path without exterior supervision. Over the past few decades, this topic has been widely studied, resulting in many approaches. All these methods first rely on sensors to capture the state of the robot and its environment. Then the autopilot calculates the best action from the sensor outputs and sends it to the low or high-level controller. The frequency of sending these commands depends on the application, the desired speed, and the sensors used. The Global Positioning System (GPS) usually provides information about the robot's state: its position and speed. In addition to GPS, sensors like accelerometers, gyroscopes, and barometers can estimate other robot states, such as attitude.

Robots also require visual perception sensors to capture their environment and detect obstacles. Early methods primarily used LIDAR or stereo cameras for accuracy and reliability. However, with the development of more efficient computer vision techniques thanks to deep learning, the use of monocular cameras is rapidly spreading. Perception is crucial in image-based navigation algorithms: the accuracy, the acquisition speed, the field of view, and the relevance of the captured images have a direct impact on performance. Nevertheless, most existing solutions use cameras with a limited field of view.

In this second thesis part, we propose a drone navigation strategy using 360° field of view images. Omnidirectional cameras can capture the entire environment in a single shot. By comparing our proposed solution to the method using limited field of view sensors, we demonstrate the relevance of using the complete drone's surroundings for navigation.

In this chapter, we first present different approaches for image-based navigation. Then we develop the one chosen in this work: deep reinforcement learning.

4.1 Image-Based Navigation Strategies

Several solutions are available for robot image-based navigation. Two main strategies stand out: classical hierarchical planning and machine learning. The former proposes to combine global trajectory planning with local motion control, while the latter is data-driven and directly links perception and motion commands in an end-to-end pipeline.

4.1.1 Map-Based

Monocular simultaneous localization and mapping (SLAM) is the most famous map-based algorithm [118, 119, 120, 121, 122]. This method creates a map of an unknown environment using a single camera. It involves simultaneously estimating the camera's location (localization) and constructing a map of the surroundings (mapping). Localization is computed using dense information or tracking the movements of features in the image. These features can be points, lines, or other image structures that are unique and easily identifiable in the images. In parallel, mapping is constructed by adding the locations of the features in the camera's field of view to the map. This is done by matching the features in the current image with those in previous images and using the estimated camera pose to determine the relative positions of the features. Monocular SLAM can be implemented using a single inexpensive camera and is relatively robust to changes in lighting conditions, resulting in a reliable navigation solution for robots. Most recent works combine perception using deep learning and map exploration, such as [123]. They perform obstacle detection and localization using a supervised CNN.

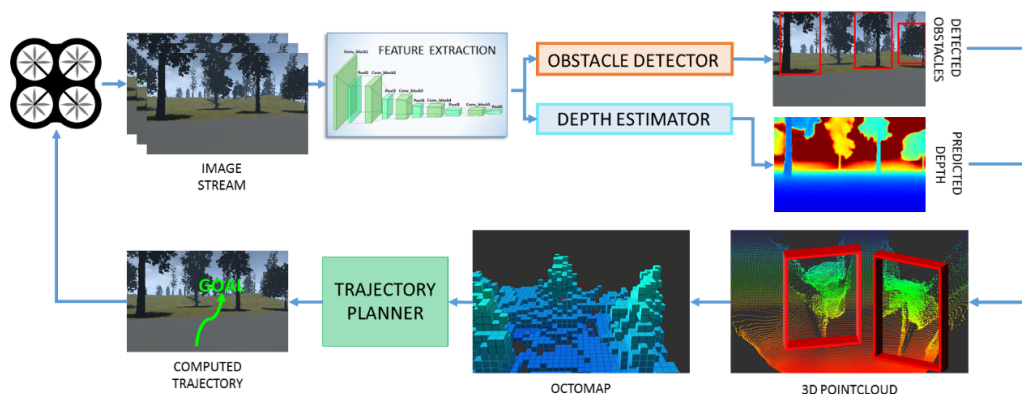


Figure 4.1: Navigation pipeline in [123]. Depth and obstacles detection are merged to build an Octomap.

In [124], the authors proposed a solution for drone navigation in forests. They combine monocular SLAM and trail path detection using deep learning to follow safer trajectories. However, this solution is limited by SLAM accuracy despite promising results in real experiments with a small drone. Besides, restricting the trajectories to forest trail paths is a significant limitation. Monocular SLAM

heavily relies on the sensors' accuracy to measure the environment, which can significantly impact the localization and mapping in dense environments or at high-speed flights [125]. Furthermore, these approaches are often limited by the computational power embedded in small UAVs: updating a 3D map while navigating is challenging and energy-consuming.

4.1.2 Learning-Based

Alternative algorithms based on machine learning techniques have been proposed to overcome the limitations of map-based navigation. Typical methods are imitation learning and deep reinforcement learning.

4.1.2.1 Imitation Learning

In the case of imitation learning [126], the agent learns to reproduce the behavior of an expert from a lot of labeled data. A dataset of actions and observations performed by a reference agent is collected and used to train a machine learning model. This model then aims to reproduce the actions that the reference agent would have performed in a given situation. This strategy provides reasonable control over the policy learned by the agent.

More recent works [127, 128] exploit deep learning to mimic expert behavior for drone navigation in indoor corridors. The dataset of expert demonstrations is built by combining images of the scene and actions to perform. The network is used as a classifier to determine the drone state and following actions, as shown in Figure 4.2.



Figure 4.2: Network outputs from a perspective image of an indoor corridor. [127]

Specific solutions for forest exploration have been proposed. In [129], the authors proposed a forest drone navigation algorithm mimicking human control. As in the case of monocular SLAM, some studies also exploit the presence of forest trails. In [130], trails are predicted using deep learning. Using an annotated dataset built for this occasion, they trained a convolutional neural network to detect and track trails. The network outputs three possible actions for the drone: turn left, go straight, or turn right, as shown in Figure 4.3.

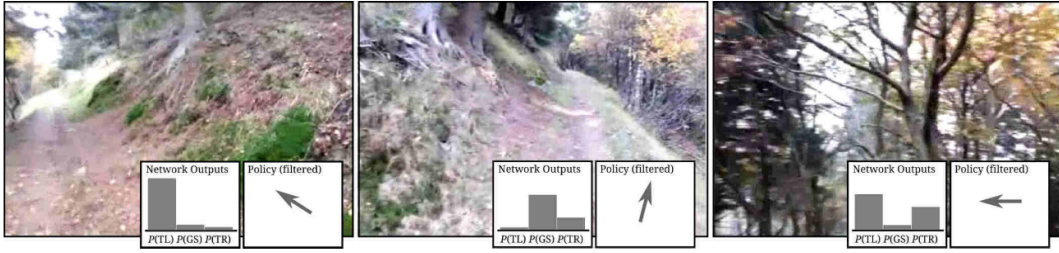


Figure 4.3: Network outputs from a perspective image of a forest trail. The drone commands are: turn left, go straight or turn right. [130]

The most critical limitation of imitation learning is its need for an extensive dataset of expert actions. This dataset must be representative of the range of situations the agent is likely to encounter in order to perform well. Unfortunately, collecting such a detailed data set is non-trivial. Furthermore, these algorithms suffer from generalization capabilities to scenarios not included in the training dataset, particularly critical failures.

4.1.2.2 Deep Reinforcement Learning

In reinforcement learning [131], an agent learns to interact with its environment in order to maximize a reward signal. This agent learns through trial and error, making decisions based on its current state and the rewards or punishments it receives for taking certain actions. This agent must balance exploration and exploitation of the state space to find the optimal policy. Reinforcement learning frameworks have been merged with deep learning to map high-dimensional sensory information and robot motion commands without referencing the ground truth. Deep neural networks allow to process large amounts of data and estimate from it the best following action. As a result, it is possible to train agents to perform complex tasks, such as playing video games or controlling robots[132].

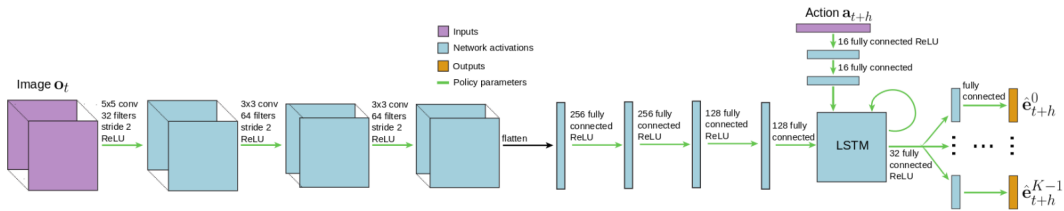


Figure 4.4: Illustration of a deep neural network predictive model. The convolutional encoder takes as input the current RGB image and processes it to form the initial hidden state of a recurrent LSTM unit. This recurrent unit takes as input H actions in a sequential fashion and produces H outputs. These outputs of the recurrent unit are then passed through additional fully connected layers to predict all K events for all H future time steps. These predicted future events, such as position or collision, enable action selection and achieve desirable events. [132]

Doukhi et al. [133] proposed a DRL solution for drone navigation in forests using a small LIDAR instead of visual inputs. Promising results are presented on real experiments with a drone reaching multiple waypoints while avoiding trees.

But this approach also presents some limitations. A very large number of iterations is often required to learn an optimal policy. Moreover, DRL may struggle to learn in environments that are too complex or uncertain, resulting sometimes in instabilities. Finally, the reward function, which plays a major part in good convergence of the agent towards the desired policy, can be very challenging to specify.

4.1.3 Combining Model-Based and Learning-Based

Current state-of-the-art point-goal navigation solutions combine map-based and learning-based approaches [134, 135, 125]. In [125], the authors combine two trajectory planners at different scales to fly a drone at high speed in dense and unstructured environments. Figure 4.5 illustrates the strategy adopted. First, a global planner establishes long-range waypoints using a map-based method. Colored pins represent these global waypoints in the figure. Then, a local planner takes care of the navigation between these waypoints, including obstacle avoidance. The resulting trajectory (green) often differs from the direct trajectory (red) due to the various obstacles. The authors favored imitation learning to design this local planner to ensure the best reliability during real test flights.

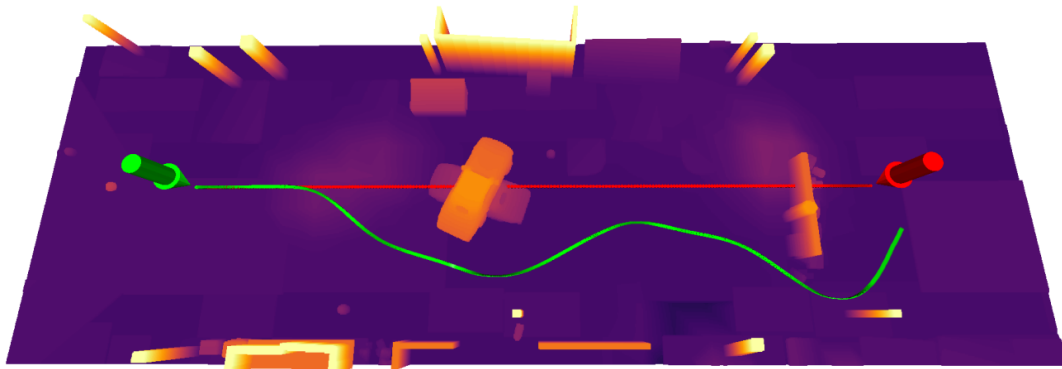


Figure 4.5: Combining global way-points in a reconstructed map and local obstacle avoidance based on learning [125]. Colored pins represent these global waypoints in the figure. The drone trajectory (green) often differs from the direct trajectory (red) due to the various obstacles.

4.1.4 Selected Navigation Method

As shown above, the best image-based navigation methods combine map-based and learning-based approaches. In this thesis, we seek to demonstrate the relevance of omnidirectional perception rather than perspective. Thus, it is expected that the most significant impact will be on obstacle detection. For these reasons, we simplify the problem by focusing only on navigation between global waypoints spaced a few dozen meters apart. We assume that these waypoints have been obtained either by a map-based method or GPS. It is between these waypoints that the drone’s vision is most important, and it is on this part that we focus.

Methods based on imitation learning, although very reliable for real applications, require the use of an expert pilot to create the training dataset. In our case, the pilot must have similar flight capabilities with perspective and omnidirectional images to compare the methods fairly. This step is a significant challenge in addition to the already time-consuming and tedious dataset creation.

Recently, deep reinforcement learning has gained interest with the development of more powerful realistic flight simulators [10, 136, 137]. In addition, with increasingly powerful GPUs, recent publications [132, 138, 4] allow obstacle avoidance by drones to be achieved with only a few hours of training, as opposed to the previous weeks or months. Furthermore, using a virtual environment allows for the exploration of critical situations without the risk of destroying the drone, which significantly increases the robustness of navigation compared to other approaches. Therefore, we favor deep reinforcement learning to perform obstacle avoidance in dense and unstructured environments.

4.2 Reinforcement Learning

The objective of reinforcement learning is to learn a policy that associates states with actions in a reward-based environment, as in Figure 4.6. The agent selects an action based on its current state and policy. This interaction with the environment results in a new state and an additional reward to evaluate the relevance of the applied policy. This sequence of actions ends with a termination state, which usually corresponds to the success or failure of the overall task. Through a succession of trials and errors, the training seeks to maximize the cumulative reward received by the agent. We provide here a brief formal definition of the framework adopted in this thesis, mostly inspired by the reference book on this field [131].

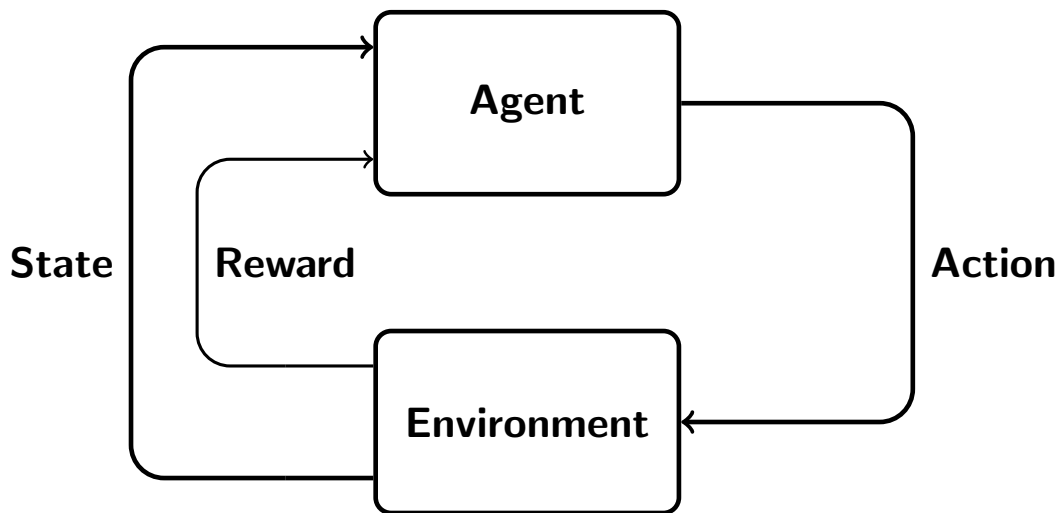


Figure 4.6: Reinforcement learning framework. The agent selects an action based on its current state and policy. This interaction with the environment results in a new state and an additional reward to evaluate the relevance of the applied policy.

4.2.1 Markov Decision Process

Drone point-goal navigation is a decision-making problem with uncertainties, which can be modeled by a Markov decision process (MDP). In this MDP, an agent interacts with the environment by performing actions following a specific policy in a given state. This state is assumed to have the Markov property, i.e. to be only dependent on the previous state. Therefore the transition probability is a function only of the previous state and action. From this same environment, the agent receives a reward, positive or negative, to promote or prevent certain behaviors.

MDP provides a formal definition for a sequential decision making process, defined as a five-tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where:

- \mathcal{S} : a finite set of states;
- \mathcal{A} : a finite set of available actions;
- $P(s, a, s')$: a state transition matrix which indicates the probability that action a in state s at time t will lead to state s' at $t + 1$;
- $R(s, a, s')$: a reward function received by the agent after performing action a at state s and getting to state s' ;
- γ : a discount factor $\in (0, 1]$, representing the difference in importance between short and long term rewards.

4.2.2 Discounted Expected Reward

A policy in reinforcement learning is a function π that associates states with actions $a = \pi(s)$. MDP objectives is to find a policy that maximizes the expected discounted sum of rewards from each state onwards, the return G_t :

$$G_t = \sum_{i=0}^T \gamma^i R(s_{t+i}, a_{t+i}, s_{t+i+1}), \quad (4.1)$$

with γ a discount factor. The choice of this discount depends on the application and the expected behavior of the agent. For example, the closer it is to 0, the more the agent is only concerned with maximizing the reward received at time t , which makes him shortsighted. On the contrary, as it approaches 1, the agent becomes more future-oriented, as future rewards are less discounted. In long-horizon tasks, such as point-goal navigation, the reward must be maximized over the long term (reaching the end goal) rather than being concerned only with immediate performance (avoiding obstacles). Therefore, the discount factor is chosen close to 1.

4.2.3 Algorithmic diversity

There are many reinforcement learning algorithms. However, most have well-defined properties that often depend on the task or the algorithmic capabilities desired.

4.2.3.1 Model-Based or Model-Free

In some cases, the agent can use or learn a model of the environment. This model can predict the next state and reward based on a given action-state pair which will help decide on a course of action by taking into account possible future situations before they are actually experienced. These methods are called model-based, as opposed to more straightforward model-free approaches that are explicitly trial-and-error learners. However, there is no prior model of the

environment in most complex problems, such as point-goal navigation in a dense, unstructured environment.

4.2.3.2 Episodic or Continuous

In some applications, there is a natural notion of a final time step: the agent’s interaction with the environment naturally breaks into subsequences called episodes. Each episode ends with a terminal state in a finite number of T steps, followed by the reset to a starting state. This type of task is called episodic. On the other hand, in some cases, there are no identifiable episodes, and the agent’s interaction with the environment continues continuously without a time limit ($T = \infty$). These tasks are called continuous.

4.2.3.3 Exploration or Exploitation

One of the challenges of reinforcement learning is the trade-off between exploration and exploitation. To maximize the reward it receives, the agent favors actions that have proven effective in the past. It exploits what it already knows. However, these actions can only be found through exploration. The agent must randomly interact with the environment to find new and possibly better actions. A common strategy is the ϵ -greedy approach which consists in picking the best action (greedy action) with a probability of $1 - \epsilon$ or a random action otherwise.

4.2.3.4 On-Policy or Off-Policy

Off-policy algorithms learn from experiences generated by a behavior policy (the policy applied) different from the target policy (the one learned). For example, this policy can be stochastic or sub-optimal, such as ϵ -greedy exploration. In contrast to the on-policy algorithms where the behavior and target policies are identical: the algorithm learns directly from the experiments generated by following the current policy. Using an identical policy in the on-policy setting generally makes these reinforcement learning algorithms less sampling efficient than non-policy ones. Nevertheless, they usually show better algorithmic stability.

4.2.4 Optimal Policy Search

To evaluate how good it is for the agent to be in state s when following a given policy π , the value-function V is defined as the expected return from a given state. Similarly, to evaluate a state-action pair (s, a) when following π , the action-value function Q (also named quality-function) is given by the expected return from a given state-action pair:

$$\begin{aligned} V_\pi(s) &= \mathbb{E}_\pi [G_t \mid s_t = s], \\ Q_\pi(s, a) &= \mathbb{E}_\pi [G_t \mid s_t = s, a_t = a]. \end{aligned} \tag{4.2}$$

Reinforcement learning aims to find an optimal policy π_* which maximizes the expected cumulative reward G_t over the long run. A policy is better than another policy $\pi \geq \pi'$ if the value function of the new policy is better $V_\pi(s) \geq V_{\pi'}(s)$ for all $s \in \mathcal{S}$. If the state-value function is optimal, an optimal policy was used by the agent. Multiple optimal policies are possible and lead all to the same optimal state-value function. The optimal value function V_* and the optimal action-value function Q_* can be defined as followed:

$$\begin{aligned} V_*(s) &= \max_{\pi} V_{\pi}(s) \text{ for all } s \in \mathcal{S}, \\ Q_*(s, a) &= \max_{\pi} Q_{\pi}(s, a) \text{ for all } s \in \mathcal{S}, a \in \mathcal{A}. \end{aligned} \tag{4.3}$$

Usually, to solve simple problems, reinforcement learning uses tabular methods [131]. They are based on the assumption that the functions V and Q can be represented in tabular form, as shown in Tables 4.1 and 4.2.

Table 4.1: V-table

	Value
State s_0	15
State s_1	20
...	...
State s_n	...

Table 4.2: Q-table

	Action a_0	Action a_1	Action a_2	Action a_3
State s_0	-1	12	0	15
State s_1	3	20	4	14
...
State s_n

Three approaches propose to solve tabular methods: dynamic programming, Monte-Carlo, or Temporal Differences. However, these tables become massive when the number of possible action-state pairs increases. Therefore, it is often necessary to use an approximation of these tables, in particular by using deep learning.

4.2.5 Dynamic Programming

Dynamic Programming (DP) refers to a collection of algorithms for solving finite MDPs using a perfect knowledge of the environment (i.e., the states, actions, reward function, and transition function are known). The policy iteration algorithm is an example, which iteratively evaluates and improves the policy until it finds the optimal policy π_* . Since a finite MDP has a finite number of policies, this process converges to an optimal policy and an optimal value function in a finite number of iterations. Nevertheless, DP algorithms are often limited by this assumption of a perfect model and are known to be computationally expensive.


```

1. Initialization
    $V(s) \in \mathbb{R}$  and  $\pi(s) \in \mathcal{A}(s)$  arbitrarily for all  $s \in \mathcal{S}$ 

2. Policy Evaluation
   Repeat
      $\Delta \leftarrow 0$ 
     For each  $s \in \mathcal{S}$ :
        $v \leftarrow V(s)$ 
        $V(s) \leftarrow \sum_{s',r} p(s', r|s, \pi(s)) [r + \gamma V(s')]$ 
        $\Delta \leftarrow \max(\Delta, |v - V(s)|)$ 
   until  $\Delta < \theta$  (a small positive number)

3. Policy Improvement
   policy-stable  $\leftarrow$  true
   For each  $s \in \mathcal{S}$ :
      $a \leftarrow \pi(s)$ 
      $\pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r|s, a) [r + \gamma V(s')]$ 
     If  $a \neq \pi(s)$ , then policy-stable  $\leftarrow$  false
   If policy-stable, then stop and return  $V$  and  $\pi$ ; else go to 2

```

Figure 4.7: Policy iteration [131].

4.2.6 Monte-Carlo

Unlike DP, Monte Carlo algorithms require only experience (i.e., sequences of sample states, actions, and rewards from interactions with an environment). They solve the problem of reinforcement learning by averaging sample values.

```

Initialize:
   $\pi \leftarrow$  policy to be evaluated
   $V \leftarrow$  an arbitrary state-value function
   $Returns(s) \leftarrow$  an empty list, for all  $s \in \mathcal{S}$ 

Repeat forever:
  Generate an episode using  $\pi$ 
  For each state  $s$  appearing in the episode:
     $G \leftarrow$  return following the first occurrence of  $s$ 
    Append  $G$  to  $Returns(s)$ 
     $V(s) \leftarrow \operatorname{average}(Returns(s))$ 

```

Figure 4.8: The first-visit MC method for estimating V_π [131].

4.2.7 Temporal Differences

Temporal Differences (TD) learning algorithms update the values during the episode and do not wait for the end of the episode, like Monte Carlo methods. It mainly improves the computational speed. Two standard TD algorithms are SARSA (on-policy) and Q-learning (off-policy).

The name SARSA comes from the quintuple $(S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1})$ needed for the algorithm. At every step, the agent chose an action a according to its current policy (with sometimes a random action to explore), and gets a reward r and a new state s' . The agent will then chose another action according to its current policy, and update the Q-table with the reward r , and the Q value of new state s' and new action a' . SARSA is on-policy since the new action a' is chosen according to the current policy.

```

Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$ 
Repeat (for each episode):
  Initialize  $S$ 
  Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
  Repeat (for each step of episode):
    Take action  $A$ , observe  $R, S'$ 
    Choose  $A'$  from  $S'$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
     $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)]$ 
     $S \leftarrow S'; A \leftarrow A'$ 
  until  $S$  is terminal

```

Figure 4.9: Sarsa: An on-policy TD control algorithm [131].

Q-learning is an off-policy version of SARSA: the agent does not have to chose a second action a' , the update will be performed with the best possible action ($\max_a Q(s, a)$). As SARSA is on-policy, its convergence properties depend on the policy used, whereas it is not the case for Q-learning, which is independent of the policy being followed.

```

Initialize  $Q(s, a), \forall s \in \mathcal{S}, a \in \mathcal{A}(s)$ , arbitrarily, and  $Q(\text{terminal-state}, \cdot) = 0$ 
Repeat (for each episode):
  Initialize  $S$ 
  Repeat (for each step of episode):
    Choose  $A$  from  $S$  using policy derived from  $Q$  (e.g.,  $\epsilon$ -greedy)
    Take action  $A$ , observe  $R, S'$ 
     $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$ 
     $S \leftarrow S'$ 
  until  $S$  is terminal

```

Figure 4.10: Q-learning: An off-policy TD control algorithm [131].

4.3 Deep Reinforcement Algorithms

Action and state spaces can become massive in complex problems, mainly when the agent’s state contains large arrays such as images. Therefore, the size needed to store action-state pairs and the time required to explore all possible combinations become problematic. Deep neural networks can be used as function approximators to overcome these restrictions. As shown in Section 2.2, they can approximate nonlinear functions and extract relevant features from any input. With deep learning, reinforcement learning is now able to generalize to unseen states, eliminate the need to maintain a lookup table in memory, and create complex mappings between actions and states. In addition, network architectures are typically much smaller than those applied to computer vision, partly because the need for precision decreases in favor of reduced learning time.

4.3.1 Value-Based Approaches

Deep Q-learning (DQN) [139] is one of the most famous DRL algorithms. As a value-based algorithm, its basic premise is that knowledge of the Q-function, defined in the equation 4.2, is sufficient to get to the optimal policy. It uses a deep neural network to represent and approximate this Q-function. Furthermore, experience replay [140] is added to improve the sample efficiency and stability: past experiments are stored in a buffer and sampled to update the network weights. The DQN is well known for demonstrating extraordinary abilities to play many video games. However, its performance remains limited and new algorithms have gradually replaced it.

4.3.2 Policy Gradient Methods

The size of the Q-function grows rapidly with the number of possible actions, which becomes very problematic in the case of a continuous action space. Policy gradient methods directly optimize the policy to maximize the expected cumulative reward. Value-based methods, such as Q-learning, estimate the optimal value function and derive a policy. In contrast, policy gradient methods, such as REINFORCE [141], directly learn the policy by updating the parameters of a policy function (classically, the weights and bias of a neural network) using gradient descent. In the case of a discrete action space, the output of the policy network will be a vector of probabilities over all possible actions. In the case of a continuous action space, the classical output of the policy network is the mean (and potentially the variance) of a Gaussian. Policy gradient methods generally have good convergence properties but are quite slow.

4.3.3 Actor-Critic Strategies

Actor-Critic combines Q-learning and policy gradient using two neural networks, one for the policy (actor) and one for the value function (critic), to maximize the expected cumulative reward. The actor (policy gradient based) decides what action to take, while the critic (value-based) evaluates how good that action is and how to adjust it. Both networks improve over time. Actor-critic methods can handle high-dimensional action spaces and have good stability and generalization capabilities. However, they can be complex to implement and require careful tuning of hyperparameters. We tested two actor-critic methods: TD3 an off-policy algorithm and PPO an on-policy solution.

4.3.3.1 TD3

Twin Delayed DDPG (TD3) [142] is a variation of the Deep Deterministic Policy Gradient (DDPG) algorithm [143]. DDPG is based on the actor-critic strategy for deterministic policies and uses experience replay to improve learning. DDPG achieves execution speeds 20 times faster than DQN on the same video games. However, the algorithm is very sensitive to hyperparameters making fine-tuning difficult. TD3 aims to solve some stability and convergence problems. It uses two critical networks to estimate the value function, a target policy smoothing technique that adds noise to the target policy during the learning process, and a delayed policy update mechanism that updates the policy less frequently than the critical networks. It has already been proposed for obstacle avoidance by drones in [138].

4.3.3.2 PPO

Proximal Policy Optimization (PPO) [144] is a simplification of the Trust Region Policy Optimization (TRPO) algorithm [145]. TRPO uses the actor-critic strategy to iteratively improve the policy in a trust region, which ensures that the policy update does not deviate too far from the previous policy. PPO adds a clipped surrogate objective function to TRPO, which further constrains the size of the policy updates to avoid instability and multiple passes over the training data using mini-batches. In the end, PPO is a simplified version of TRPO that is easier to implement while exhibiting identical performance. It has performed well on various drone flight problems [146, 147, 148, 149].

In the next chapter, we present the adopted navigation solution based on the deep reinforcement learning framework presented here.

Chapter 5

Navigation Framework

Our goal is to propose a point-goal navigation strategy using omnidirectional images and compare it to its perspective reference. Furthermore, we chose deep reinforcement learning to perform obstacle avoidance as explained in Section 4.1.4. The following presents the training and test flight environment, our navigation solution, and the different navigation performances.

5.1 Flight Environment

We chose the forest context to test our navigation solution. Forests are dense and unstructured environments, offering a great diversity for testing obstacle avoidance. Therefore, we first use a schematic forest to compare the omnidirectional and perspective methods. Then, we test the most promising solutions in a photorealistic forest.

5.1.1 RDMAP: Simplified Training and Testing Environment

Unreal Engine [108] was chosen as rendering software for its wide range of available scene complexity, from very simplified scenes to photo-realistic environments. Connected to Airsim [10], 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 build a simplified forest environment named RDMAP. This 200×200 meters terrain consists of many vertical cylinders randomly placed schematizing a dense forest of tree trunks. Fig. 5.1 shows an overview of this simplified environment. AirSim directly provides RGB images and the associated ground truth semantic segmentation and depth.

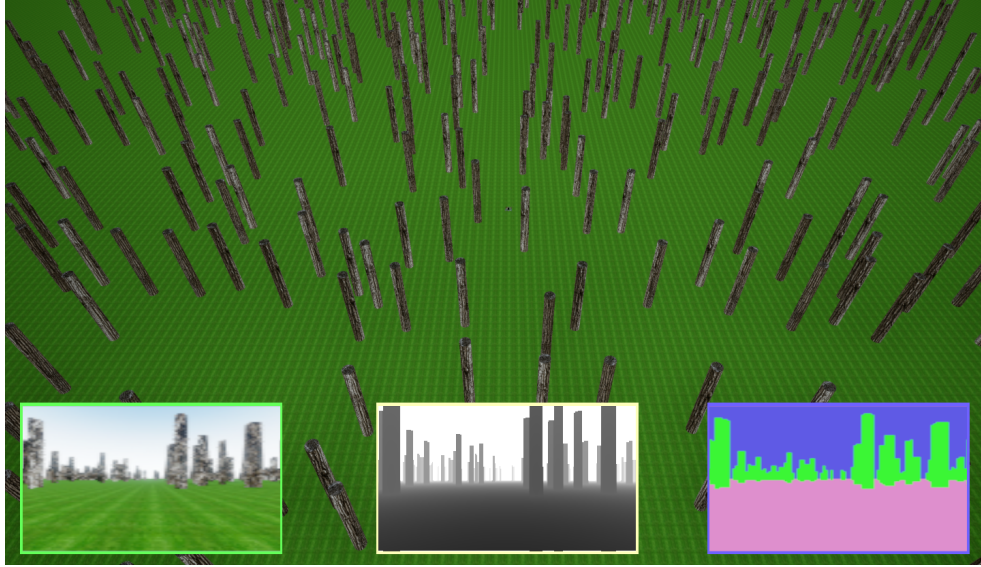


Figure 5.1: Overview of the RDMAP environment.

5.1.2 RDFOREST: Photorealistic Testing Environment

To reduce the gap with reality, we create the RDFOREST forest environment. Using the best rendering capabilities of Unreal Engine and forest textures from its marketplace [109], we build a photorealistic forest with complex lighting. Unlike the simplified trunks in RDMAP, the trees here have different sizes, branches, and dense foliage. As a result, RDFOREST is a challenging environment where the captured images are much more complex to analyze and translate into actions.



Figure 5.2: Overview of the RDFOREST environment.

5.2 Proposed Framework

As presented in Section 4.2.1, reinforcement learning follows a specific framework illustrated in Figure 5.3. At each time-step $t_k = k\Delta t$, where Δt is the control sampling time, 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. We define in this section the action and state spaces and the reward that we considered in this thesis.

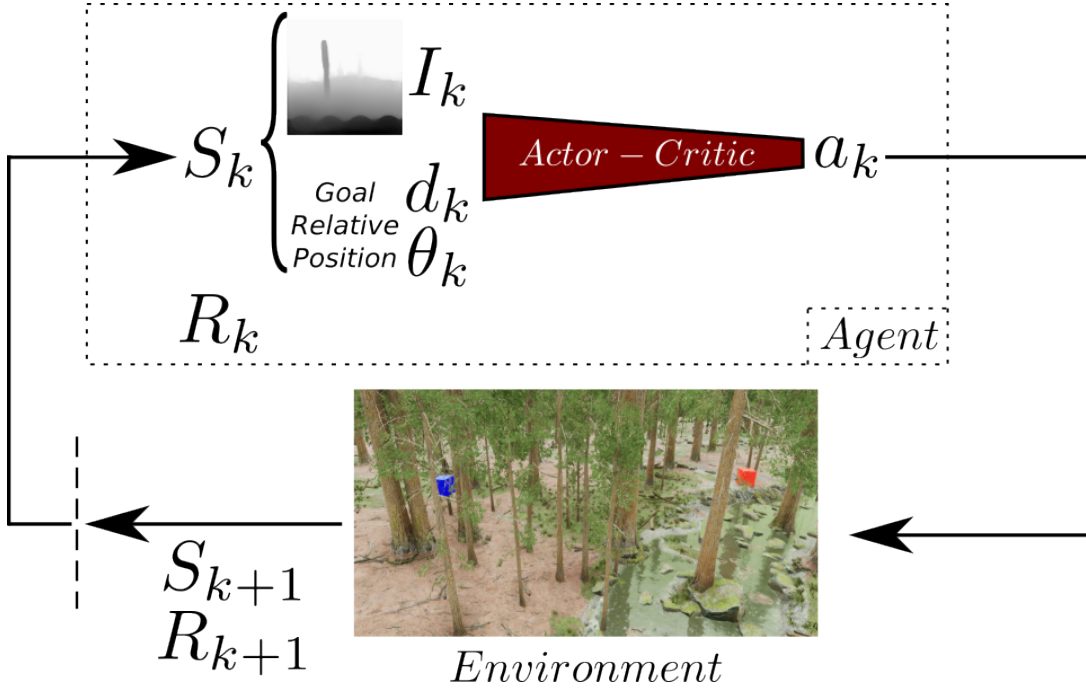


Figure 5.3: General DRL framework. The drone state gathers information about the goal relative position (d_k, θ_k) and a visual capture of the agent’s environment I_k as defined in Section 5.2.2.

5.2.1 Action Space

In our considered point-goal navigation problem, the agent must fly between trees without collision. Due to the mostly vertical structure of trees, we choose to perform obstacle avoidance in an iso-altitude plane. Therefore, the drone keeps a constant altitude during flight while the controller focuses on rotational movements. As a result, the agent’s action space is $\mathcal{A} \subset [-\pi, \pi]$, which corresponds to a desired yaw angle among N_b directions for the episodic case or infinite directions for continuous case:

$$a = \begin{cases} \left(\frac{2i}{N_b-1} - 1\right) \pi & i \in \{0, \dots, N_b - 1\} & \text{episodic case.} \\ i\pi & i \in [-1, 1] & \text{continuous case.} \end{cases} \quad (5.1)$$

5.2.2 Drone State

In reinforcement learning, the agent uses its current state to determine the best following action. This state must be relevant and complete enough to provide sufficient information to make an appropriate 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. In practice, we consider a drone at position $P_k = (x_k, y_k, z_k)$ with a yaw angle ψ_k heading towards a fixed goal at position $P^* = (x^*, y^*, z^*)$. At each time-step t_k , we define the distance d_k and the angle θ_k to goal:

$$\begin{aligned} d_k &= \|P_k - P^*\|_2, \\ \theta_k &= \arctan2(y^* - y_k, x^* - x_k) - \psi_k. \end{aligned} \quad (5.2)$$

The drone captures its surroundings with its perception sensor and transforms it into an image noted I_k . Several types of modalities are tested in this study and are described in the next Section 5.2.3. The resulting drone state at time-step t_k is defined by:

$$S_k = [d_k, \theta_k, I_k]. \quad (5.3)$$

5.2.3 Visual Modalities used as Input

Perception is crucial to achieve obstacle avoidance. This thesis compares two visual modalities for input image I_k : RGB and depth. For depth, we study both ground truth depth and depth estimated by deep learning. The AirSim simulator directly provides RGB images and the associated absolute depth. We perform an additional cropping in the [0..5] meter range to remain representative of the capabilities of a small onboard LIDAR.

In parallel, we can use specialized spherical networks or standard perspective solutions with spherical adaptations to estimate depth from equirectangular RGB images. In the first part of this thesis, we have already proposed a detailed overview of such techniques and highlighted the following statement: most omnidirectional methods require significant computational power and specific training on spherical datasets, which makes them less suitable for drone navigation. Therefore, we choose the same lightweight network as in the first part of this thesis: MIDAS [60]. This convolutional neural network is one of the lightest and most accurate perspective depth estimation networks published, with performances already proven in several mobile and drone applications [150, 151]. We directly use its pre-trained version `midas_v21_small` to be representative of an embedded drone solution. First, we use the baseline version of the MIDAS network. We will apply our spherical adaptation proposed in the first part of this thesis later on.

Two different fields of view are used for all these modalities: a limited field (90°) and an omnidirectional one (360°). To make a fair comparison, we keep the same encoder between the different study cases. The image resolution I_k is independent of modality or FOV and fixed at 100×100 pixels.

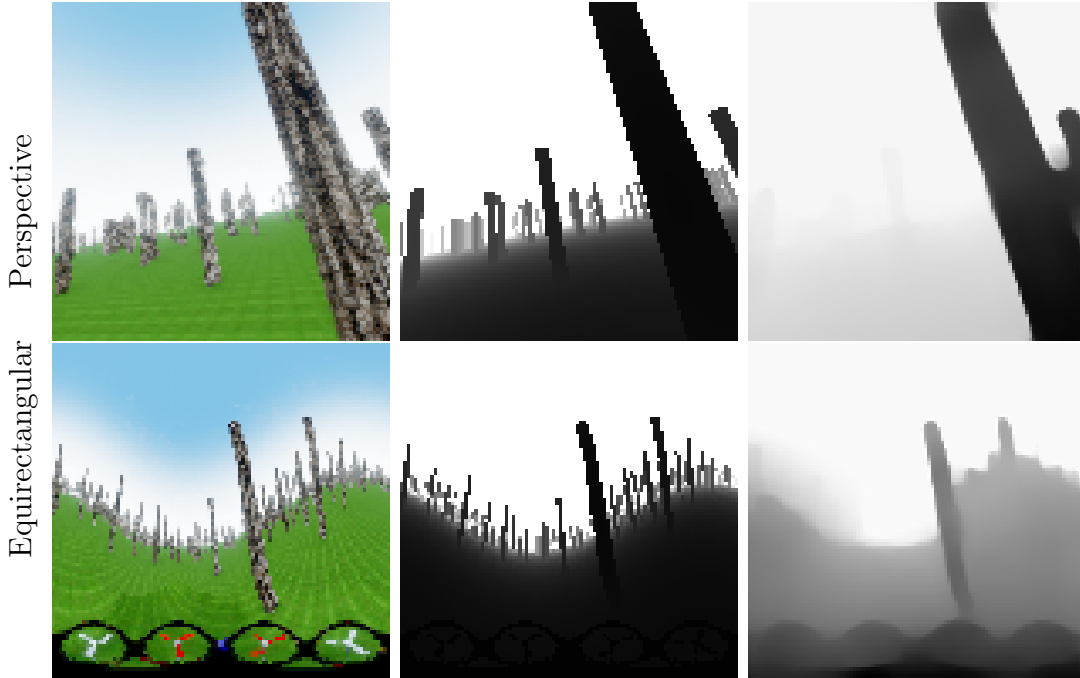


Figure 5.4: Different visual modalities considered for drone navigation: (From left to right: RGB, ground truth depth, and estimated depth with baseline MIDAS network [60]).

5.2.4 Actor-Critic Network

The Actor-Critic network architecture selected is based on contributions that have already proven effective for drone navigation [152, 125, 138]. First, the image I_k is preprocessed using a succession of convolutions and fully connected networks. The resulting 32-dimensional vector is then combined with the goal information (d_k, θ_k) to determine the next action a_k using another fully connected network. The global pipeline is shown in Figure 5.5.

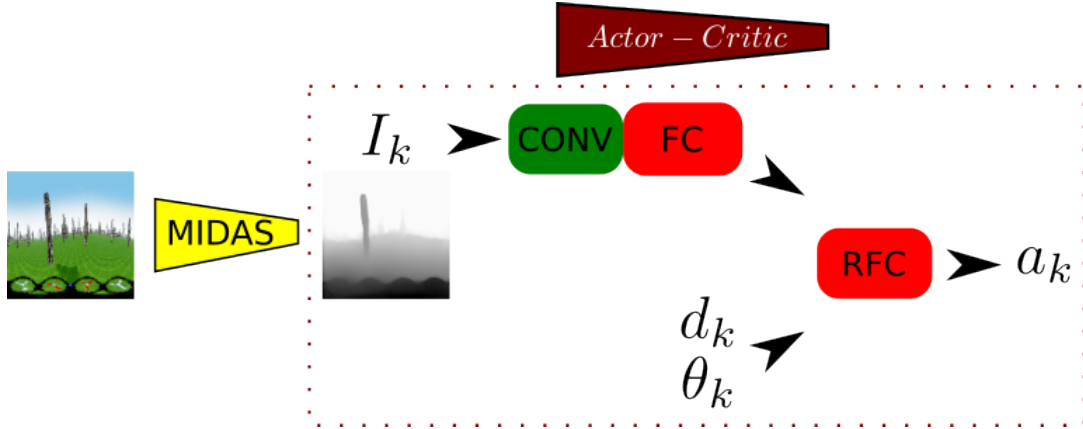


Figure 5.5: The visual observation I_k is encoded into a 32-dimensional vector using convolutional operations (CNN) and a fully connected network (FC). Then, combining this output vector and the goal information (d_k, θ_k) , another fully connected network (RFC) predicts the agent’s next action a_k . The specific case of estimated depth as input is presented in this example. The MIDAS network is not used when the navigation focuses on RGB images, ground truth depth or semantic segmentation.

The precise architecture of the layers is presented in Table 5.1.

Table 5.1: Network architecture for the actor-critic pipeline. Total: 60814 parameters.

Layer	Type	Nb parameters
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, θ_k)		
RFC1	FC(34,64)	2172
RFC2	FC(64,128)	8320
RFC3	FC(128,128)	16512
RFC4	FC(128,37)	4773

5.2.5 Reward

The reward function design is critical to ensure proper convergence of the agent to the expected behavior. Since many attempts are often needed to reach a suitable candidate, we adapt previously published reward functions for the specific case of drone point-goal navigation. We mainly based our final solution on the proposal of [153]. However, we perform a slight modification to their proposal. In our case, the drone does not always have access to absolute depth. Therefore, we remove the dependency on safe flight distance to keep the same reward function for all visual modalities.

Our final reward function depends on the goal relative position (d_k, θ_k) to push the agent towards this goal, a penalty term (-0.02) to penalize too-long trajectories, and a terminal reward R_{end} that promotes or punishes the agent depending on the episode end state. The goal is considered as reached when the distance between the drone and this goal is less than a minimal distance d_{min} . Moreover, the drone is penalized when it does not reach the goal in less than t_{max} time steps or moves too far away at a distance greater than d_{max} . At each time step t_k , the resulting function is given by:

Push towards the goal

$$R_k = \boxed{-0.1d_k - 0.05\|\theta_k\|} + \boxed{-0.02} + \boxed{R_{end}} \quad (5.4)$$

Penalization **Terminal Reward**

- 5 if goal reached ($d_k < d_{min}$);
- -5 if collision;
- -2 if stuck or away ($t_k > t_{max}$ or $d_k > d_{max}$);

In the next chapter, we finally present the evaluation of the proposed navigation framework in dense and unstructured environments, such as forests.

Chapter 6

Navigation Evaluation

This chapter presents the final navigation evaluation using the deep reinforcement learning framework defined in the previous chapter. First, we define the metrics evaluating the different trajectories produced by our proposed navigation models. Then, we describe our training schedule and the valid hyperparameters for each simulation. Finally, we present the evaluation of the different solutions considered. This evaluation consists of four steps:

- Selection of a deep reinforcement learning solver;
- Evaluation and comparison of navigation based on omnidirectional or perspective images;
- Improvement of the proposed navigation solution by applying a spherical adaptation based on the distortion-aware convolution strategy presented in the thesis first part;
- Evaluation of the most promising models on a more photorealistic forest environment without additional training.

6.1 Metrics

We use two standard metrics to evaluate navigation solutions. Considering N_t the total number of paths to be evaluated and S_i the Boolean indicator of the success of episode i ($S_i = 1$ if successful, 0 otherwise), we define:

- the Success Rate (SR) to directly assess the drone’s abilities to reach its goal ($d_k < d_{min}$):

$$\text{SR} = \frac{1}{N_t} \sum_{i=1}^{N_t} S_i \quad ; \quad (6.1)$$

- the Success weighted by Path Length (SPL), as defined in [154]:

$$\text{SPL} = \frac{1}{N_t} \sum_{i=1}^{N_t} S_i \frac{\ell_i}{p_i} \quad , \quad (6.2)$$

where p_i is the length of the drone’s trajectory and ℓ_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 1. Besides, failed tests are strongly penalized by the boolean value S_i .

6.2 Training Schedule and Hyperparameters

We use the simplified forest environment RDMAP to train all proposed navigation solutions using different visual modalities and fields of view. Each training takes 100k time steps and uses the same schedule and parameters for a fair comparison. Table 6.1 presents the hyperparameters used to tune the deep reinforcement learning algorithm.

Table 6.1: Simulation hyperparameters.

Hyperparameter	Value
Learning rate	0.0003
Number of steps	2048
Batch size	64
Number of epochs	10
Gamma	0.99
Δ_t	200 ms
N_b	37 actions
d_{min}	1 meter
d_{max}	100 meters
t_{max}	200 time-steps

During training, the distance between the drone’s initial position and goal is always 20 meters. Then, during inference, we also test longer distances (40 and 60 meters) to challenge our navigation solution. The training is performed on Nvidia Tesla-V100 graphics cards and lasts about 8 to 10 hours. Inference on 600 drone trajectories takes between 120 and 180 minutes.

6.3 Reinforcement Learning Solver selection

Before performing an intense navigation comparison between visual modalities and field of view, we have to select a deep reinforcement learning algorithm. We use two point-goal navigation test to perform this initial selection: one with and another without obstacles. We select three different algorithms as presented in Section 4.3: two off-policy strategies with different levels of algorithmic complexity: DQN and TD3, and an on-policy algorithm: PPO. Note that the feature extraction network architecture described in Section 5.2.4 was not yet complete for these initial tests. We initially used the solution proposed in [138]. We then refined this model using complementary works [125, 152].

6.3.1 Test with No Obstacle

First, we focus on a simple point-goal navigation task with no obstacles. Since visual capture is not helpful, we do not use the image I_k of the drone state to predict the best following action. The drone initially starts in the center of the map and must reach its target on a 20-meter circle. Figure 6.1 shows the trajectories obtained with the three different solvers. We observe that the solution using DQN presents jerky trajectories with abrupt direction changes like a bang-bang controller. On the contrary, TD3 and PPO show smoother results. These results are unsurprising, as DQN is one of the oldest deep reinforcement learning algorithms and has been gradually supplanted by more efficient and robust methods such as TD3 and PPO. Therefore, we keep only these two solvers in the next test phase.

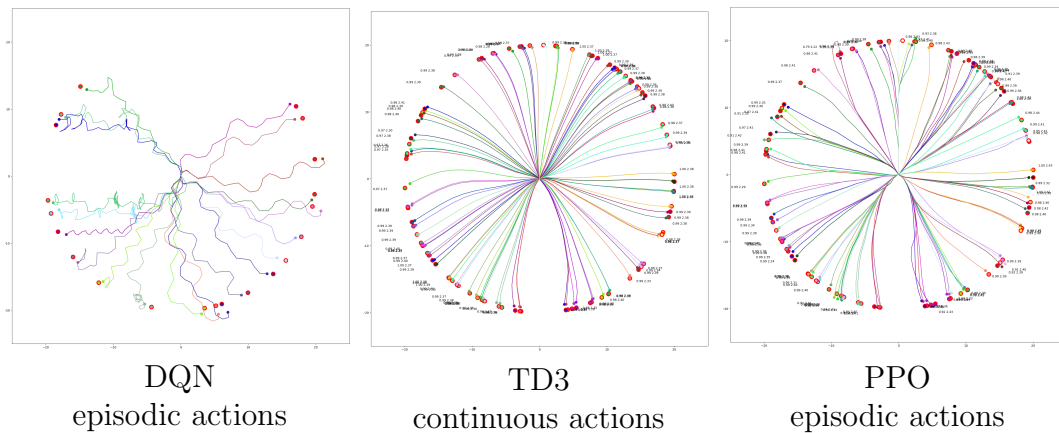


Figure 6.1: The drone starts at the center and must reach its target on a 20-meter circle without obstacles. DQN solution is jerky, whereas TD3 and PPO are smoother.

6.3.2 Test with Obstacles

We add obstacles to the previous point-goal navigation task. The obstacles are simple vertical cylinders such as the ones used in RDMAP. Contrary to the previous test case, we use omnidirectional RGB image I_k as input to predict the best following action. Table 6.2 presents the navigation performances obtained with the two remaining solvers.

Table 6.2: Comparing Success Rate (SR in %) with TD3 and PPO as solver. Each evaluation is performed on 100 runs.

RUN	SR of TD3 (%) (\uparrow)	SR of PPO (%) (\uparrow)
20 meters goal	50	80
40 meters goal	35	74

Despite the obstacles, the navigation solution using PPO remains exceptionally stable and avoids most collisions: the success rate is 80% for 20 meters trajectories. On the contrary, the model using TD3 is unstable and fails several times. TD3 is more sample-efficient thanks to its off-policy characteristic but much less stable and robust than PPO in our specific case. A long fine-tuning phase would be necessary to cope with its higher sensitivity and instability. On the contrary, out-of-the-box PPO is slightly slower but much more promising.

Thus, given these first results and the good robustness of PPO in complex situations (SR of 74% for trajectories twice as long as those experienced during training), we chose this algorithm as solver. As explained earlier, we further improved these results by refining the architecture of the feature extraction network. Numerous convolution parameter and layer changes led to a wide range of navigation performance. We use the architecture with the best results in what follows: the one described in Table 5.1.

6.4 Omnidirectional versus Perspective Navigation

We now compare the performances of our proposed solution using omnidirectional or perspective images. To do so, we train and test each model in the RDMAP environment. The same training schedule and hyperparameters are used regardless of the input visual modality or field of view. We test each solution on the same 600 point-to-goal navigation trajectories. However, the goal distance is not only 20 meters as in training, but we also investigate more distant goals at 40 meters and 60 meters. It puts to the test the robustness of our navigation algorithm in situations not seen during training. Finally, we compute the survival rate (SR) and the success weighted by path length (SPL), defined in Section 6.1, for all tested trajectories. As described in Section 5.2.3, we compare two different fields of view for all visual modalities considered: a limited one (90°) and an omnidirectional one (360°). Table 6.3 compares the navigation based on omnidirectional models and the perspective references.

Table 6.3: Comparing 90° and 360° modalities. Each evaluation is performed on 600 runs and 3 goal-distances (20, 40 and 60 meters). Ground Truth (GT), Estimated Depth (ED).

RUN	SR (%) (↑)	SPL (%) (↑)
90° FOV RGB	68.3	52.8
360° FOV RGB	85.7	62.6
90° FOV GT Depth	76.0	54.0
360° FOV GT Depth	88.8	68.6
90° FOV ED	69.7	57.9
360° FOV ED	86.0	67.0

Looking at the metrics, the omnidirectional case shows significantly better performance than the perspective case. 360° FOV images optimize collision avoidance and navigation tasks. First, the larger FOV improves navigation and obstacle detection tasks. Obstacles invisible in a narrow FOV are now detected in omnidirectional images, reducing the number of collisions (higher SR). In addition, the drone has a better understanding of its environment, which allows for better trajectory optimization (higher SPL). These results demonstrate the great potential of large FOV sensors for point navigation, regardless of the type of visual information captured.

As expected, ground truth depth is the best-performing visual modality. In the second place, estimated depth has a slight advantage over RGB images and shows promising performance, especially when using 360° FOV images. As a result, despite its lightweight architecture, the prediction accuracy of MIDAS is sufficient to build a more reliable navigation solution than a solution based on RGB alone.

Below we present two examples of trajectory comparisons using ground truth depth as a visual modality.

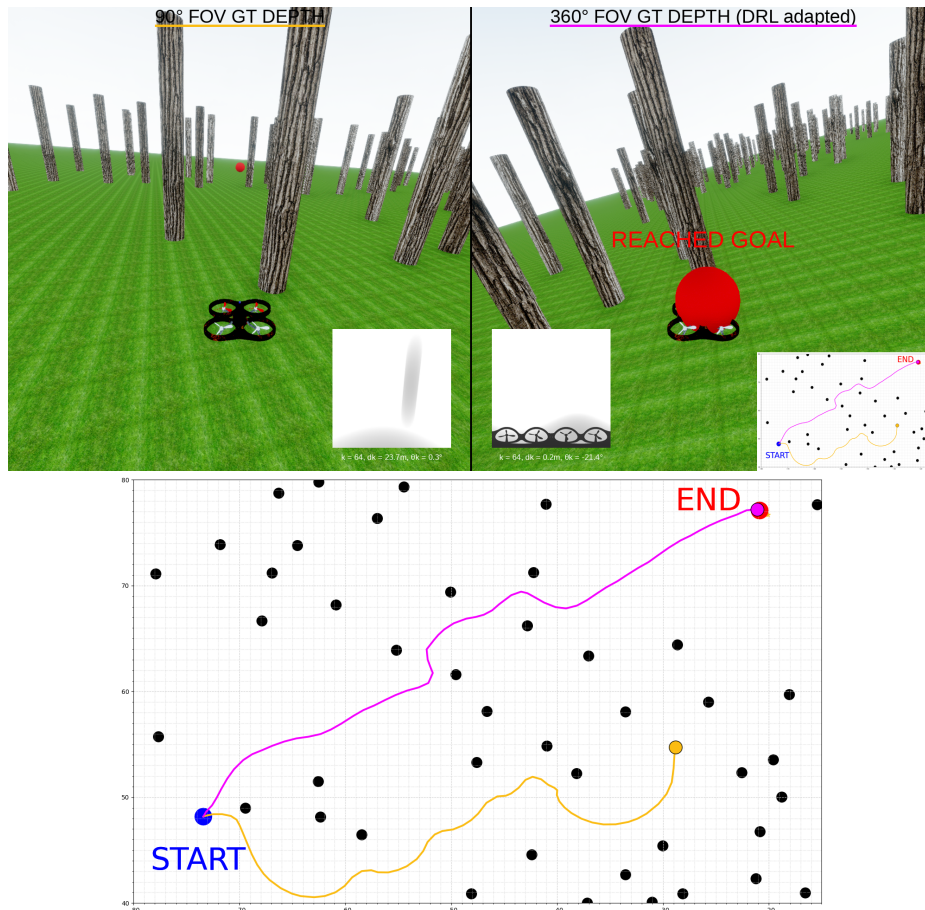


Figure 6.2: Trajectory sample 1 for an identical goal at 60 meters on the RDMAP environment using 90° FOV (in orange on the left) and 360° FOV (in purple on the right) ground truth depth as input visual modality. [Link to the video.](#)

Figure 6.2 illustrates these findings by comparing the trajectories of the perspective model version (orange on the left) and the omnidirectional one (purple on the right) for the same starting point and goal. The distance to the goal is 60m, three times longer than the one seen during training. The solution using a 360° FOV reaches the destination in only 12 seconds (64 time-steps), while the limited FOV version reaches it in 18 seconds (91 time-steps). The omnidirectional images allow for a better understanding of the scene and a better localization of obstacles and the drone. It translates into more direct trajectories to the goal while keeping a reasonable safety margin to avoid collisions.

Figure 6.3 shows trajectories for another pair of start and end points. In this scenario, the omnidirectional model reaches the goal in 13 seconds (66 time-steps), approximately the same time as in the previous example. In contrary, the perspective solution gets stuck between several obstacles and, due to its limited vision, cannot cross and eventually crashes after 38 seconds (190 time-steps).

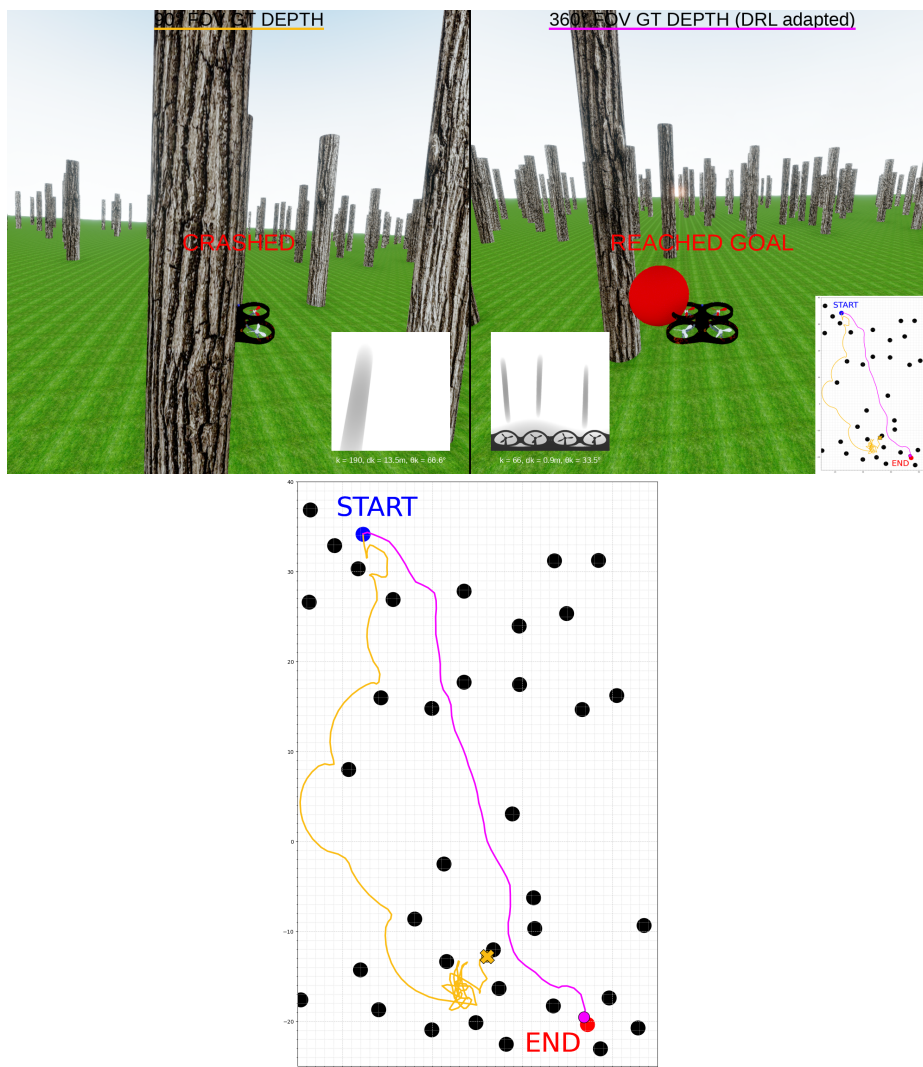


Figure 6.3: Trajectory sample 2 for an identical goal at 60 meters on the RDMAP environment using 90° FOV (in orange on the left) and 360° FOV (in purple on the right) ground truth depth as input visual modality. [Link to the video.](#)

6.5 Distortion-Aware Convolutions for DRL

The proposed navigation solution shows promising results using 360° FOV observations as inputs. However, as shown in Section 2.3, all spherical projections present some distortions. In particular, equirectangular images show significant distortions near the polar regions.

Therefore, we reuse the proposed spherical adaptation described in Section 2.3 and apply it on two different networks. First, the convolution layers of the actor-critic network used in the deep reinforcement learning algorithm are modified. Second, for models using estimated depth, we adapt the MIDAS network to improve depth prediction in equirectangular images without additional training, as we did in the first thesis part. Figure 6.4 shows different distortion-aware kernel shapes in function of their position in the equirectangular image.

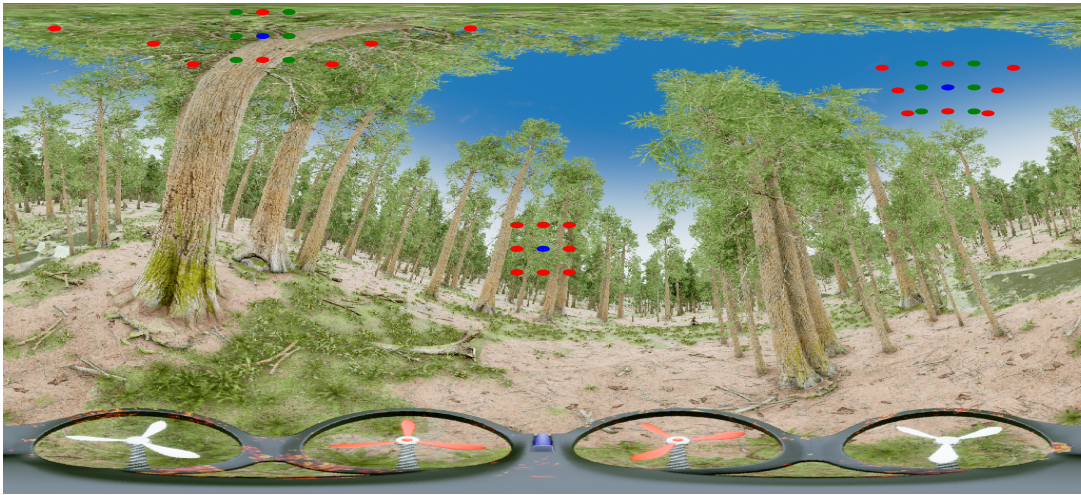


Figure 6.4: Example of kernels with different latitude and longitude. In blue is the center of the kernel, in green the perspective kernel and in red the adapted equirectangular one. The wider distortions are near the poles.

6.5.1 Actor-Critic Network Adaptation

We modify the four convolutional layers of the proposed actor-critic network architecture, previously presented in Figure 5.5. Prior to training, the offsets tables are computed based on the resolution of the observation used and the different parameters of each adapted convolutional layer. Figure 6.5 shows how this additional plugin is applied on the network architecture. The proposed navigation solution is trained in the RDMAP environment in a process similar to that presented in Section 6.2.

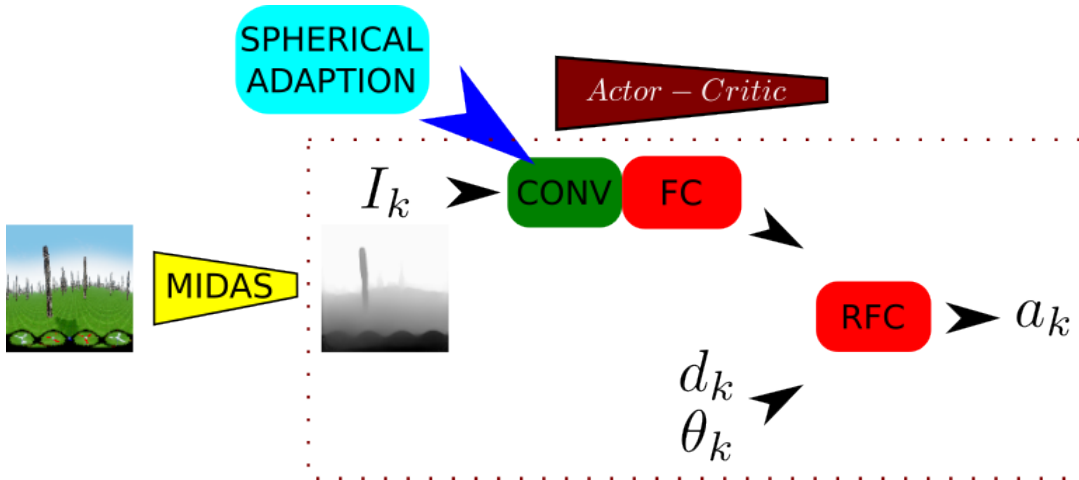


Figure 6.5: Pre-computed spherical adapted offsets are added to the four convolution layers of the Actor-Critic network during training and testing of the navigation solution.

The spherically adapted actor critic model (DRL adapted) is tested in the RDMAP environment and compared to its baseline from Section 6.4. Table 6.4 shows the performances of 360° FOV navigation based on ground truth or estimated depth.

Table 6.4: Performances in RDMAP.

RUN	SR (%) (\uparrow)	SPL (%) (\uparrow)
360° FOV GT depth (baseline)	88.8	68.6
360° FOV GT depth (DRL adapted)	94.8	78.3
360° FOV ED (baseline)	86.0	69.0
360° FOV ED (DRL adapted)	89.5	73.3

For each modality, the distortion-aware solutions show highly better performance. Maintaining local pixel coherence during convolutions helps the actor-critic network to better detect obstacles in the I_k input image. As a result, the drone trajectories are better optimized, with fewer collisions and faster paths. In particular, it allows the spherically adapted ED solution to outperform the non-adapted GT depth based solution, especially in trajectory optimization.

6.5.2 MIDAS Network Adaptation

We reuse the spherically adapted version of MIDAS network presented in Section 3.2.2. We already demonstrated better depth estimation results on various datasets and transfer it to the navigation application. The actor-critic network spherical adaptation remains active as it has shown excellent results in the previous section. Figure 6.6 shows the new global pipeline.

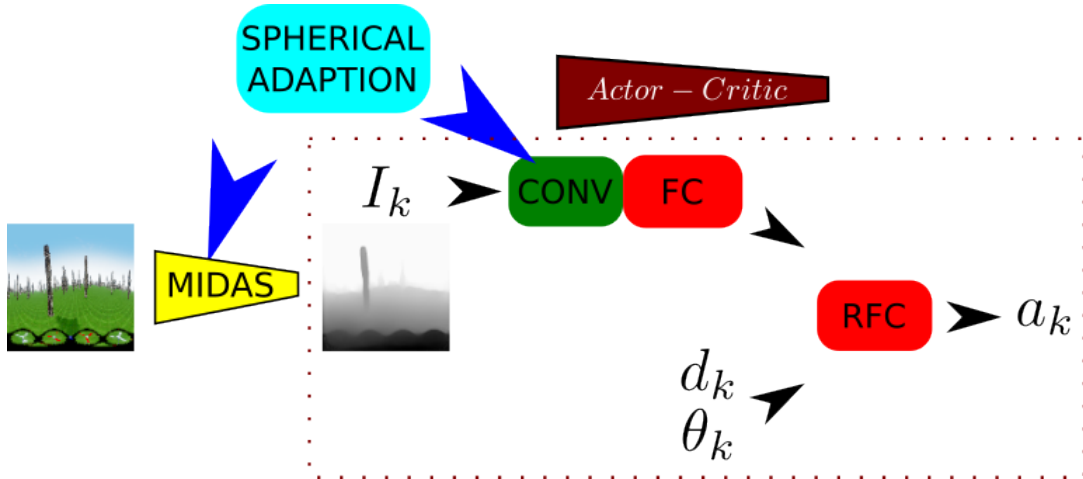


Figure 6.6: The convolutions of the MIDAS decoder are modified without additional network training to take into account spherical distortions for depth prediction. Therefore, the final proposed DRL pipeline has two spherical adaptations: one for the MIDAS network and one for the Actor-Critic network.

The navigation solution using spherically adapted MIDAS network is trained and tested in the RDMAP environment similarly to previous runs. Table 6.5 shows the evaluation metrics. As expected, the more accurate depth estimation the obstacle detection and more generally the navigation performances. The success rate is higher thanks to the MIDAS spherical adaptation. In parallel, the global scene understanding is improved and results in faster trajectories (higher SPL). Therefore, using a larger FOV and spherical adaptations allows the final version of our solution based on estimated depth to go from an initial success rate of 70% to a final version close to 90%. Moreover, the drone trajectories also gain noticeably in speed, as we can see in the following example.

Table 6.5: Performances in RDMAP.

RUN	SR (%) (\uparrow)	SPL (%) (\uparrow)
360° FOV ED (DRL & MIDAS adapted)	89.8	74.5

Figure 6.7 illustrates these results by comparing the trajectories of the perspective model version (orange on the left) and the omnidirectional version (purple on the right) for the same starting point and goal. Similarly, the distance to the goal is 60m. The solution using 360° FOV estimated depth reaches the destination in 17 seconds (81 time-steps). Unsurprisingly, this time is longer than the previous example with ground truth depth in Figure 6.2, but the drone still keeps a reasonable safety margin to avoid collisions during its flight. In contrast, the limited FOV version crashed into an obstacle after 9 seconds (45 time-steps) because it did not detect the tree trunk earlier and was not fast enough to avoid it. These results again encourage using omnidirectional visual inputs for obstacle avoidance by drones.

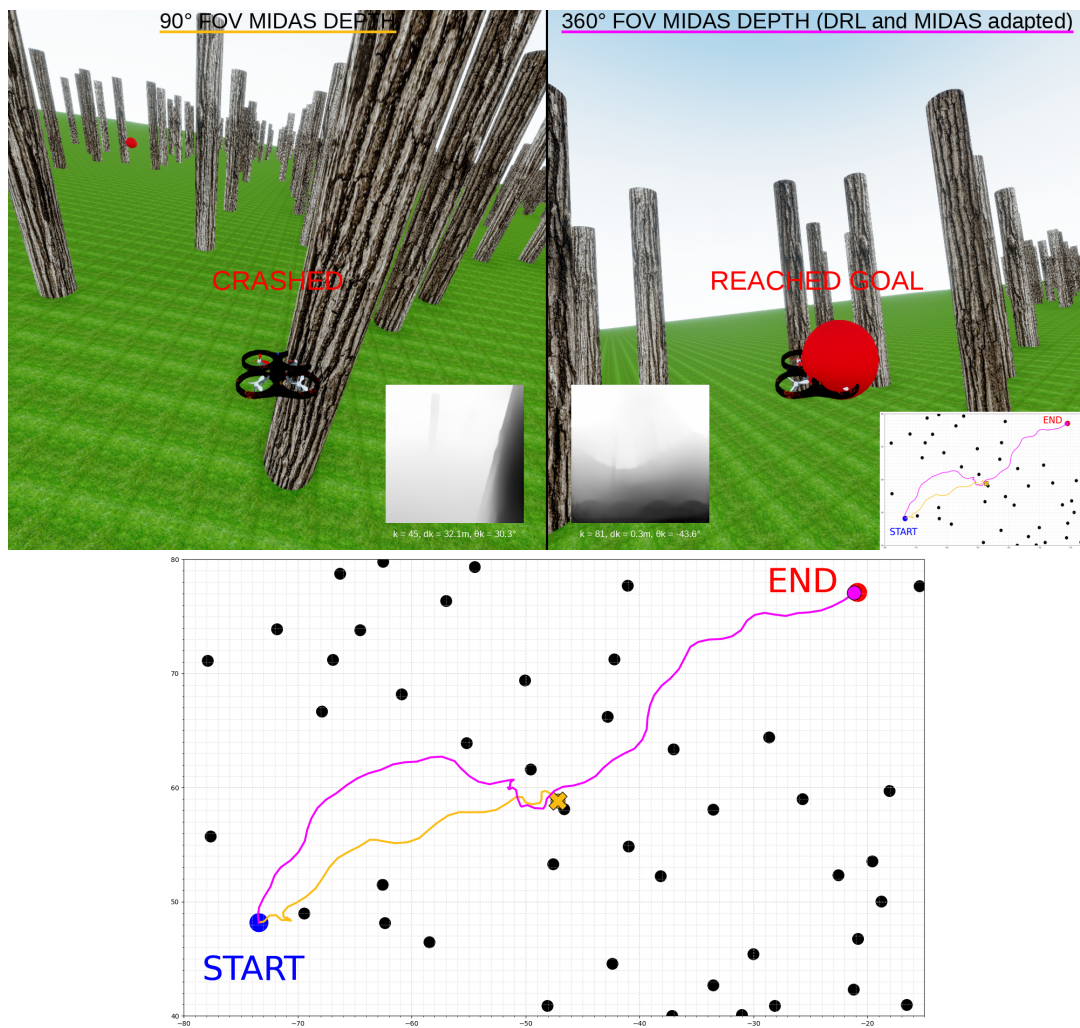


Figure 6.7: Trajectory sample 1 for an identical goal at 60 meters on the RDMAP environment using 90° FOV (in orange on the left) and 360° FOV (in purple on the right) estimated depth as input visual modality. The 360° FOV solution on the right use distortion-aware convolutions in the actor-critic and MIDAS networks. [Link to the video.](#)

Figure 6.8 shows trajectories for the same start and end points as ground truth depth trajectory sample 2. In this scenario, both models reach the goal. The spherically adapted omnidirectional estimated depth model goes faster in 13 seconds (66 time-steps) than the perspective version in 19 seconds (85 time-steps).

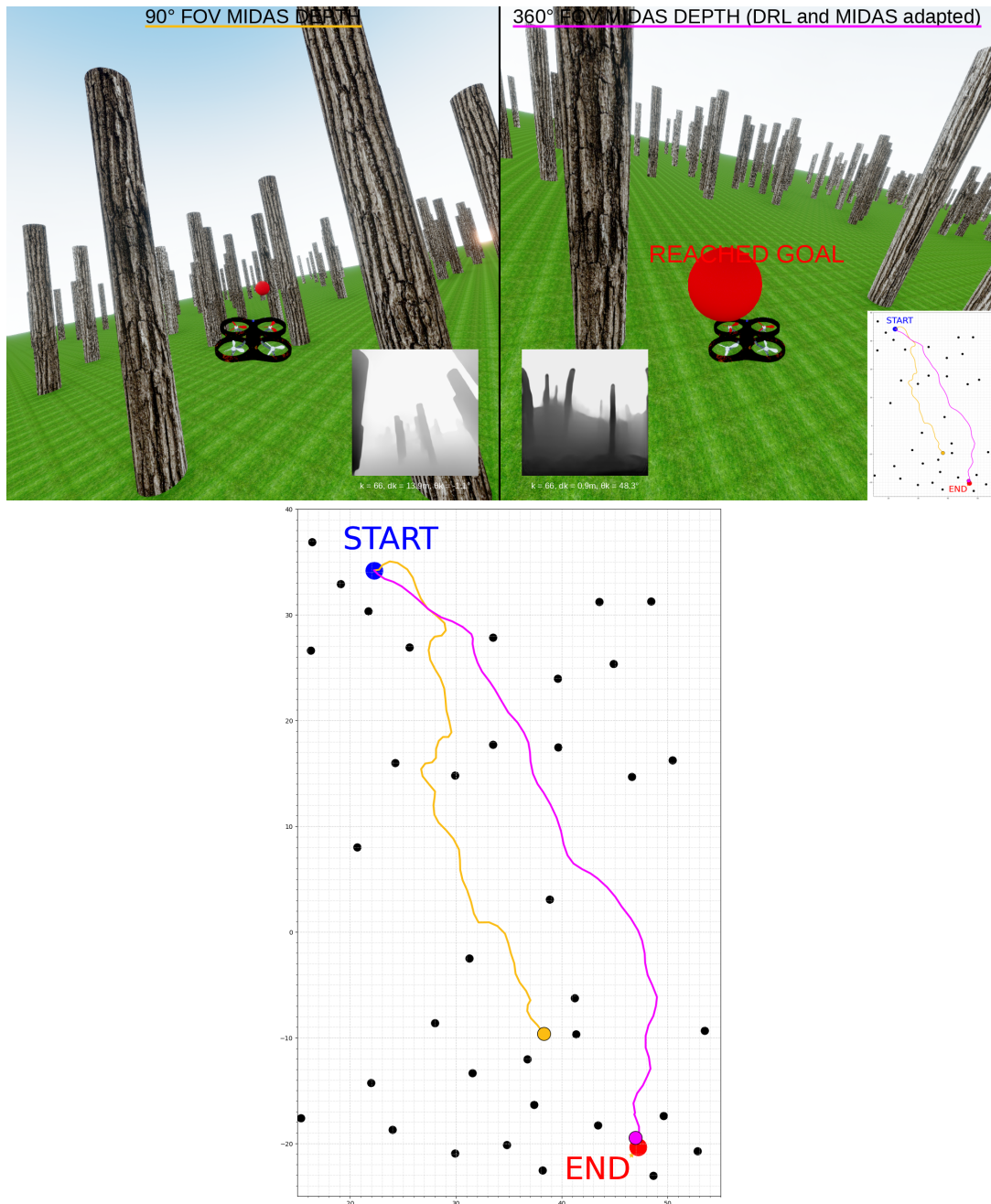


Figure 6.8: Trajectory sample 2 for an identical goal at 60 meters on the RDMAP environment using 90° FOV (in orange on the left) and 360° FOV (in purple on the right) estimated depth as input visual modality. The 360° FOV solution on the right use distortion-aware convolutions in the actor-critic and MIDAS networks. [Link to the video.](#)

6.6 Generalization to a Photorealistic Forest

To test the robustness of our most promising navigation solutions in different scenarios, we test them directly in a more complex photorealistic environment without additional training. We propose to use the RДФOREST forest environment presented in Section 5.1.2. Unlike the simplified trunks in RDMAP, the trees here have different sizes, branches, and dense foliage. As a result, RДФOREST is a challenging environment where the captured images I_k are much more complex to analyze and translate into actions. Figure 6.9 shows some observations from this environment.



Figure 6.9: RДФOREST: (left to right: RGB, ground truth depth, and estimated depth with baseline MIDAS network [60]).

The promising pre-trained models from the previous sections were directly tested on 600 objectives of 20 meters, 40 meters, and 60 meters. The Table 6.6 presents the result of the best performing solutions from the previous sections: the spherically DRL adapted ground truth and the spherically DRL & MIDAS adapted estimated depth.

Table 6.6: Performances in RДФOREST.

RUN	SR (%) (\uparrow)	SPL (%) (\uparrow)
90° FOV GT depth	81.0	69.1
360° FOV GT depth (DRL adapted)	89.7	76.9
360° FOV ED (DRL & MIDAS adapted)	84.7	62.1

Although a challenging test using much more complex images than those used for training, our proposed solution still performs very well. The DRL & MIDAS adapted 360° FOV estimated depth reaches almost 85% success. This proves the

robustness of our proposed solution to domain change despite the increase in observational complexity.

The MIDAS network also proves its stability in predicting consistent depth estimation in various environments without additional training. Despite its small size, it is reliable enough to build a DRL navigation solution with performance close to ground truth modalities. Our proposed spherical adaptation further improves navigation performance without additional training, computational slowdown, or implementation of complex code. Figure 6.10 presents an example of trajectory in the RDFOREST environment performed using 360° FOV estimated depth with spherical adaptation in DRL and MIDAS networks. The distance to the goal is 60 meters and the drone reach it in 19 seconds (91 time-steps). These results again encourage using omnidirectional visual inputs for obstacle avoidance by drones and demonstrate good robustness of our proposed navigation solution to domain shift.

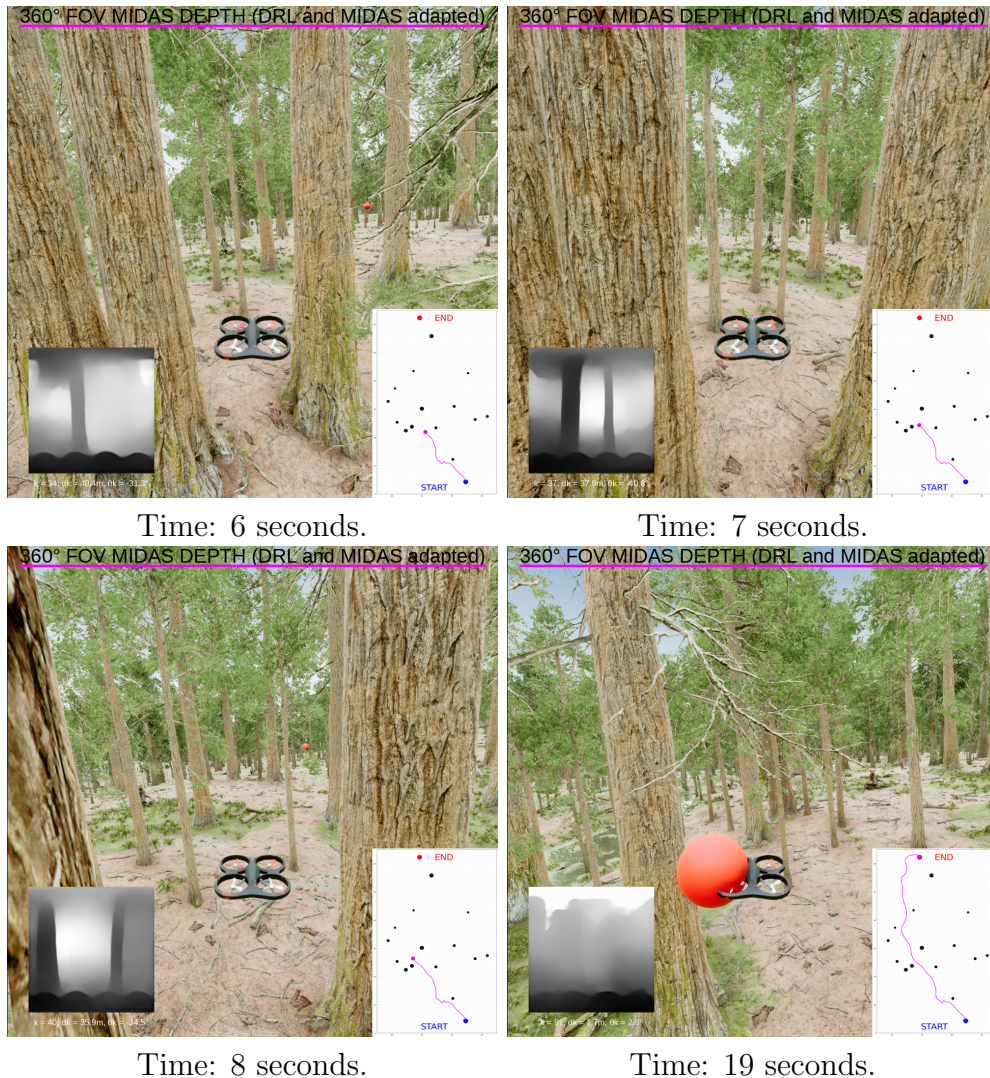


Figure 6.10: Trajectory sample for an 60 meters goal on the RDFOREST environment using 360° FOV estimated depth as input visual modality with spherical adaptation in DRL and MIDAS networks. [Link to the video.](#)

Conclusion of Part II

Perception is crucial for image-based navigation algorithms. Therefore, widening the field of view greatly improves performance. This is what we have demonstrated in the second part of this thesis by focusing on drone point-goal navigation in dense and unstructured environments such as forests.

For this purpose, two distinct fields of view were used as input to the same deep reinforcement learning-based navigation method: a 90° limited FOV and a 360° equirectangular one. In all cases tested, the omnidirectional vision detects more obstacles, significantly improving collision avoidance and success rate. Moreover, the 360° understanding of its environment helps the drone to evolve more easily and quickly, resulting in faster trajectories toward its goal while maintaining a reasonable safety distance during most of the flight.

In parallel, we tested several visual modalities as input to our navigation method. Unsurprisingly, ground truth depth is the modality with the best performance. Indeed, the absolute depth information allows the drone to know the precise position of all the elements in the scene at any time. Moreover, this modality is robust to any change in the environment. Nevertheless, we have shown promising navigation performances using an ultra-light convolutional network like MIDAS to estimate depth from monocular RGB images. Despite the significant spherical distortions, this network trained on perspective images provides reliable and accurate depth prediction, sufficient to outperform a solution using only RGB.

Furthermore, we used the part I findings and applied the distortion-aware convolutions strategy to consider the distortions in equirectangular images. We used this spherical adaptation on the deep reinforcement learning actor-critic network and the MIDAS depth estimation network. In each case, we have shown that a better local coherence between pixels during convolution operations significantly improves performances.

Finally, even when tested in a new photorealistic environment, our navigation solutions demonstrated high robustness to domain shift.

The full video examples are available at [155].

Chapter 7

Conclusion

7.1 Summary of Contributions

In this work, we have proposed a method for omnidirectional image processing using distortion-sensitive convolutions for vision and robotics applications. This approach improves the local coherence of pixels during convolution operations at the heart of most computer vision algorithms. This strategy can be easily adapted to **any pre-trained convolutional network with perspective images without additional training**. It provides significant **time savings** for developing applications using omnidirectional images while maintaining a **low implementation cost** and **improving estimation accuracy**. It also keeps up with new architectures regularly proposed in deep learning for perspective images.

In part I, we have demonstrated the generalization of our approach to three fundamental computer vision tasks: semantic segmentation, monocular depth and optical flow. Tested on virtual outdoor images and complex real-world scenarios, **the adapted spherical model consistently outperformed the baseline version while maintaining close runtimes and memory sizes**. Even when tested on images whose distribution is far from the one used during training, our strategy has shown **good robustness to domain shift**.

In part II, we extended the demonstration to robotics, particularly aerial navigation in dense and unstructured environments. Adapting a state-of-the-art algorithm to omnidirectional images showed that **distortion-aware convolutions can improve point-goal navigation performance**. We have also shown that it is possible to spherically adapt a depth estimation method initially reserved for perspective images and to build a more reliable navigation method than the one based only on RGB images while maintaining identical runtimes.

In parallel, we have shown that using omnidirectional vision to navigate significantly increases the success rate compared to a method with a limited field of view. A broader view improves obstacle detection, which significantly reduces the number of collisions. In addition, the robot better perceives its surroundings and thus better optimizes its trajectory. **Navigation based on omnidirectional images is safer and faster**. Our solution shows robustness even when tested in a more complex environment than the one used during training.

7.2 Perspectives

The contributions made in this thesis and parallel work have opened up many avenues and directions for future research. Therefore, we now propose some possible perspectives.

7.2.1 Distortion-Aware Transformers

Computer vision algorithms are now in a phase of hesitation and exploration between Convolutional Neural Networks and Transformers. CNNs are well known for their sliding receptive field that can pay attention to local pixels. The Transformers, from another perspective, can better model global correlation while requiring a higher computational cost. Each of these methods has advantages and disadvantages. As a result, it still needs to be made clear which model can consistently lead to better performance. Nevertheless, Transformers will undoubtedly continue to be studied and used.

It would be interesting to extend our spherical adaptation strategy to Transformers. This adaptation could be done similarly to our proposition for CNNs and would extend Transformers, initially dedicated to perspective images, to omnidirectional applications. For CNNs, we modified the convolution kernels' shape, regular for perspective images, to take into account the local distortions of equirectangular images. Similarly, the Transformers split the input image into several regular patches during pre-processing. This splitting operation could be adapted to exactly match the omnidirectional distortions. Some authors [156, 157] have already contributed to panoramic processing, as shown in Figure 7.1. However, in these works, the offsets used for slicing are learned during training. It would be interesting to directly use the formulas presented in Section 2.4.1 and avoid any additional learning on an omnidirectional dataset.

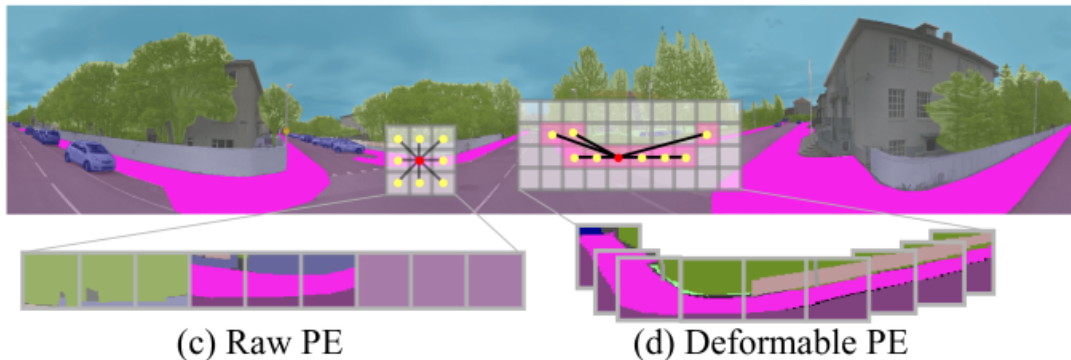


Figure 7.1: (c) Standard Patch Embeddings. (d) Deformable Patch Embedding for panoramas. [156].

7.2.2 Solve the Periodicity Issue

As presented in this thesis, distortion-aware convolutions allow the transfer of networks trained on perspective images to omnidirectional applications. However, since no additional training is performed on spherical images, the models do not incorporate the fact that equirectangular images are periodic. For example, in optical flow estimation (Section 3.2.3), the network cannot correctly estimate displacements on the edges: it does not know that these edges are mathematically connected. It would be interesting to find a solution to incorporate this periodicity in networks trained only with perspective images.

A first very rough approach would be to use several projections of the same equirectangular image by performing rotations to modify the position of the elements in the image center and edges. The work of [110, 98] presents some promising propositions. As shown in Figure 7.2, flow prediction is improved by combining the contributions of an image and its 180° rotation along the vertical axis. Since the runtimes of most estimation networks are very small, it would be possible to combine additional projections (rotations) to improve the prediction further. However, this would only enhance the estimation without incorporating the periodicity of spherical images.

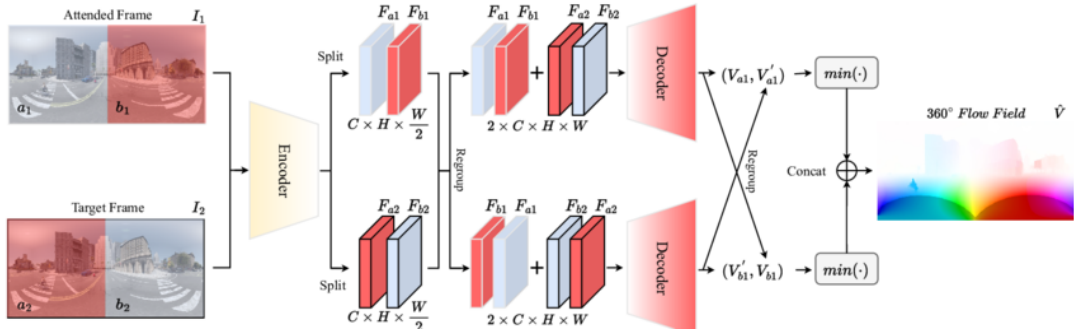


Figure 7.2: Cyclic Flow Estimation. Partitioned feature maps are extracted from the encoder of the attended frame and the target frame. According to the cyclicity of the left and right boundaries of a panoramic image, the features extracted via the encoder, are regrouped into two feature pairs and sent to the decoder to obtain the complementary optical flow field. The 360° flow can finally be obtained via \min operations [110].

7.2.3 Omnidirectional Image generation

Image generation methods are popular nowadays thanks to the stable diffusion models [158]. From any textual input, these methods can create matching images. Moreover, with the recent addition of add-ons like [159], it is possible to further control the output by adding higher-level information such as contour, semantic segmentation, or depth as presented in Figure 7.3.

However, most of this work is in the context of perspective images. Therefore, studying the generation of equirectangular images using these models could be very interesting. In particular, distortion-aware convolutions could help the transfer from perspective to omnidirectional scenes.



Figure 7.3: Controlling Stable Diffusion with ADE20K [58] segmentation map [159]. On the left is the control semantic segmentation control input used to create the different landscape images on the right.

Bibliography

- [1] D. Marr. *Vision: A Computational Investigation into the Human Representation and Processing of Visual Information* Freeman. 4576th ed. Vol. 218. 1982, 398 p. (Cit. on p. 23).
- [2] Y. LeCun, Y. Bengio, and G. Hinton. “Deep Learning”. In: *Nature* 521.7553 (2015), pp. 436–444 (cit. on pp. 23, 34).
- [3] F. Pearson. *Map Projections : Theory and Applications*. CRC Press Boca Raton, Fla, 1990, 372 p. (Cit. on p. 24).
- [4] Y. Song, K. Shi, R. Penicka, and D. Scaramuzza. “Learning Perception-Aware Agile Flight in Cluttered Environments”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. London, United Kingdom: IEEE, May 2023, pp. 1–7 (cit. on pp. 25, 83).
- [5] A. Rituerto, L. Puig, and J. Guerrero. “Comparison of omnidirectional and conventional monocular systems for visual SLAM”. In: *Proceedings of the Workshop on Omnidirectional Vision, Camera Networks and Non-classical Cameras (OMNIVIS)*. Zaragoza, Spain: IEEE, June 2010, pp. 1–8 (cit. on p. 25).
- [6] C.-O. Artizzu, H. Zhang, G. Allibert, and C. Demonceaux. “OmniFlowNet: a Perspective Neural Network Adaptation for Optical Flow Estimation in Omnidirectional Images”. In: *Proceedings of the International Conference on Pattern Recognition (ICPR)*. Milan, Italy: IEEE, Jan. 2021, pp. 2657–2662 (cit. on pp. 26, 53, 55, 65, 70).
- [7] C.-O. Artizzu, G. Allibert, and C. Demonceaux. “OMNI-CONV: Generalization of the Omnidirectional Distortion-Aware Convolutions”. In: *Journal of Imaging* 9.2 (2023), pp. 1–16 (cit. on pp. 26, 27, 58, 65).
- [8] C.-O. Artizzu, G. Allibert, and C. Demonceaux. “OMNI-DRL: Learning to Fly in Forests with Omnidirectional Images”. In: *Proceedings of the Symposium on Robot Control (SYROCO)*. Matsumoto, Japan: IFAC, Oct. 2022, pp. 1–6 (cit. on pp. 26, 27).
- [9] C.-O. Artizzu, G. Allibert, and C. Demonceaux. “Deep Reinforcement Learning with Omnidirectional Images: application to UAV Navigation in Forests”. In: *Proceedings of the International Conference on Control, Automation, Robotics and Vision (ICARCV)*. Singapore, Singapore: IEEE, Dec. 2022, pp. 1–6 (cit. on pp. 26, 27).

-
- [10] S. Shah, D. Dey, C. Lovett, and A. Kapoor. “AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles”. In: *arXiv abs/1705.05065* (2018), pp. 621–635 (cit. on pp. 27, 53, 83, 92).
- [11] F. Aziz, O. Labbani-Igbida, A. Radgui, and A. Tamtaoui. “Generic Spatial-Color Metric for Scale-Space Processing of Catadioptric Images”. In: *Computer Vision and Image Understanding* 176-177 (Nov. 2018), pp. 54–69 (cit. on p. 32).
- [12] S. Yogamani, C. Hughes, J. Horgan, G. Sistu, S. Chennupati, M. Uricar, et al. “WoodScape: A Multi-Task, Multi-Camera Fisheye Dataset for Autonomous Driving”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Seoul, Korea: IEEE, Oct. 2019, pp. 9307–9317 (cit. on p. 32).
- [13] W. S. McCulloch and W. Pitts. “A Logical Calculus of the Ideas Immanent in Nervous Activity”. In: *The Bulletin of Mathematical Biophysics* 5 (4 Dec. 1943), pp. 115–133 (cit. on p. 34).
- [14] F. Rosenblatt. “The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.” In: *Psychological Review* 65 (6 1958), pp. 386–408 (cit. on p. 35).
- [15] K. Fukushima. “Cognitron: A Self-Organizing Multilayered Neural Network”. In: *Biological Cybernetics* 20 (3-4 1975), pp. 121–136 (cit. on p. 35).
- [16] A. L. Maas, A. Y. Hannun, A. Y. Ng, et al. “Rectifier Nonlinearities Improve Neural Network Acoustic Models”. In: *Proceedings of the International Conference on Machine Learning (ICML)*. Vol. 28. Atlanta, USA: JMLR, June 2013, pp. 1–6 (cit. on p. 35).
- [17] B. Xu, N. Wang, T. Chen, and M. Li. “Empirical Evaluation of Rectified Activations in Convolutional Network”. In: *arXiv abs/1505.00853* (2015), pp. 1–5 (cit. on p. 35).
- [18] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. “Gradient-Based Learning Applied to Document Recognition”. In: *Proceedings of the Institute of Electrical and Electronics Engineers (IEEE)* 86 (11 1998), pp. 2278–2324 (cit. on pp. 36, 39).
- [19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “ImageNet classification with deep convolutional neural networks”. In: *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*. Vol. 25. Lake Tahoe, USA, 2012, pp. 1097–1105 (cit. on p. 36).
- [20] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. San Diego, USA, 2015, pp. 1–14 (cit. on pp. 36, 39).
- [21] K. He, X. Zhang, S. Ren, and J. Sun. “Deep Residual Learning for Image Recognition”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, USA: IEEE, June 2016, pp. 770–778 (cit. on pp. 36, 39, 40, 58).
-

- [22] A. Howard, M. Sandler, B. Chen, W. Wang, L.-C. Chen, M. Tan, et al. “Searching for MobileNetV3”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Seoul, Korea: IEEE, Oct. 2019, pp. 1314–1324 (cit. on p. 36).
- [23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al. “Attention is All You Need”. In: *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*. Long Beach, USA, 2017, pp. 6000–6010 (cit. on p. 37).
- [24] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, et al. “An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. Vienna, Austria, May 2021, pp. 1–22 (cit. on p. 38).
- [25] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, et al. “A Survey on Vision Transformer”. In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 45 (1 Jan. 2022), pp. 87–110 (cit. on p. 38).
- [26] S. Khan, M. Naseer, M. Hayat, S. W. Zamir, F. S. Khan, and M. Shah. “Transformers in Vision: A Survey”. In: *ACM Computing Surveys* 54 (10 Jan. 2022), pp. 1–41 (cit. on p. 38).
- [27] R. Ranftl, A. Bochkovskiy, and V. Koltun. “Vision Transformers for Dense Prediction”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Montreal, Canada: IEEE, Oct. 2021, pp. 12159–12168 (cit. on pp. 38, 41).
- [28] H. Xu, J. Zhang, J. Cai, H. Rezatofighi, and D. Tao. “GMFlow: Learning Optical Flow via Global Matching”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, USA: IEEE, June 2022, pp. 8121–8130 (cit. on pp. 38, 43, 65, 69).
- [29] Z. WU, Z. Zhou, G. Allibert, C. Stolz, C. Demonceaux, and C. Ma. “Transformer fusion for indoor rgb-d semantic segmentation”. In: *Available at SSRN 4251286* (2022) (cit. on p. 38).
- [30] F.-F. Li, J. Johnson, and S. Yeung. *An Introduction to Deep Neural Networks for Computer Vision*. http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture11.pdf. 2017 (cit. on p. 39).
- [31] L. Najman and M. Schmitt. “Watershed of a Continuous Function”. In: *Signal Processing* 38 (1 July 1994), pp. 99–112 (cit. on p. 39).
- [32] N. Ikonomatakis, K. Plataniotis, M. Zervakis, and A. Venetsanopoulos. “Region growing and region merging image segmentation”. In: *Proceedings of the International Conference on Digital Signal Processing (DSP)*. Santorini, Greece: IEEE, July 1997, pp. 299–302 (cit. on p. 39).
- [33] M. Thompson, R. O. Duda, and P. E. Hart. “Pattern Classification and Scene Analysis”. In: *A Wiley-Interscience publication* 7 (4 1974), 370 p. (Cit. on p. 39).

-
- [34] J. Canny. “A Computational Approach to Edge Detection”. In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 8 (6 Nov. 1986), pp. 679–698 (cit. on p. 39).
- [35] M. Kass, A. Witkin, and D. Terzopoulos. “Snakes: Active contour models”. In: *International Journal of Computer Vision (IJCV)* 1 (4 Jan. 1988), pp. 321–331 (cit. on p. 39).
- [36] Z. Wu, D. P. Paudel, D.-P. Fan, J. Wang, S. Wang, C. Demonceaux, et al. “Source-free depth for object pop-out”. In: *IEEE ICCV*. 2023 (cit. on p. 39).
- [37] J. D. Lafferty, A. McCallum, and F. C. N. Pereira. “Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data”. In: *Proceedings of the International Conference on Machine Learning (ICML)*. 1. San Francisco, USA: JMLR, 2001, pp. 282–289 (cit. on p. 39).
- [38] P. Krähenbühl and V. Koltun. “Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials”. In: *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*. Vol. 24. Granada, Spain, 2011, pp. 1–9 (cit. on p. 39).
- [39] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. “Pyramid Scene Parsing Network”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, Hawaii: IEEE, July 2017, pp. 1–10 (cit. on pp. 39, 40, 58).
- [40] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 40 (4 Apr. 2018), pp. 834–848 (cit. on p. 39).
- [41] J. Long, E. Shelhamer, and T. Darrell. “Fully Convolutional Networks for Semantic Segmentation”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Boston, USA: IEEE, June 2015, pp. 3431–3440 (cit. on pp. 39, 57).
- [42] O. Ronneberger, P. Fischer, and T. Brox. “U-Net: Convolutional Networks for Biomedical Image Segmentation”. In: *Computer Science* 9351 (2015), pp. 234–241 (cit. on p. 39).
- [43] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. In: Las Vegas, USA: IEEE, June 2016, pp. 779–788 (cit. on p. 39).
- [44] V. Badrinarayanan, A. Kendall, and R. Cipolla. “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation”. In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 39 (12 Dec. 2017), pp. 2481–2495 (cit. on p. 39).
-

- [45] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. “Scene Parsing through ADE20K Dataset”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, Hawaii: IEEE, July 2017, pp. 5122–5130 (cit. on pp. 40, 58).
- [46] B. Zhou, H. Zhao, X. Puig, T. Xiao, S. Fidler, A. Barriuso, et al. “Semantic Understanding of Scenes Through the ADE20K Dataset”. In: *International Journal of Computer Vision (IJCV)* 127 (3 Mar. 2019), pp. 302–321 (cit. on pp. 40, 58).
- [47] P. Sinha and E. Adelson. “Recovering Reflectance and Illumination in a World of Painted Polyhedra”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Berlin, Germany: IEEE, May 1993, pp. 156–163 (cit. on p. 40).
- [48] S. Fidler. *CSC420: Intro to Image Understanding*. http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf. 2023 (cit. on p. 40).
- [49] D. Scharstein and R. Szeliski. “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms”. In: *International Journal of Computer Vision (IJCV)* 47 (2002), pp. 7–42 (cit. on p. 40).
- [50] M. Liu, M. Salzmann, and X. He. “Discrete-Continuous Depth Estimation from a Single Image”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Columbus, USA: IEEE, June 2014, pp. 716–723 (cit. on p. 40).
- [51] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun. “Dense Monocular Depth Estimation in Complex Dynamic Scenes”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, USA: IEEE, June 2016, pp. 4058–4066 (cit. on p. 40).
- [52] D. Eigen, C. Puhrsch, and R. Fergus. “Depth map prediction from a single image using a multi-scale deep network”. In: *Advances in Neural Information Processing Systems (NIPS)* 3 (January 2014), pp. 2366–2374 (cit. on p. 40).
- [53] D. Eigen and R. Fergus. “Predicting Depth, Surface Normals and Semantic Labels with a Common Multi-scale Convolutional Architecture”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Santiago, Chile: IEEE, Dec. 2015, pp. 2650–2658 (cit. on p. 40).
- [54] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. “Deep Ordinal Regression Network for Monocular Depth Estimation”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City, USA: IEEE, June 2018, pp. 2002–2011 (cit. on p. 41).
- [55] S. F. Bhat, I. Alhashim, and P. Wonka. “AdaBins: Depth Estimation Using Adaptive Bins”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, USA: IEEE, June 2021, pp. 4008–4017 (cit. on p. 41).

-
- [56] C. Godard, O. M. Aodha, and G. J. Brostow. “Unsupervised Monocular Depth Estimation with Left-Right Consistency”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, Hawai: IEEE, July 2017, pp. 6602–6611 (cit. on p. 41).
- [57] C. Godard, O. M. Aodha, M. Firman, and G. Brostow. “Digging Into Self-Supervised Monocular Depth Estimation”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Seoul, Korea: IEEE, Oct. 2019, pp. 3827–3837 (cit. on p. 41).
- [58] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. “Unsupervised Learning of Depth and Ego-Motion from Video”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, Hawai: IEEE, July 2017, pp. 6612–6619 (cit. on pp. 41, 116).
- [59] D. Wofk, F. Ma, T.-J. Yang, S. Karaman, and V. Sze. “FastDepth: Fast Monocular Depth Estimation on Embedded Systems”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. Montreal, Canada: IEEE, May 2019, pp. 6101–6108 (cit. on p. 41).
- [60] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. “Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-Shot Cross-Dataset Transfer”. In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 44 (3 Mar. 2022), pp. 1623–1637 (cit. on pp. 41, 61, 95, 96, 110).
- [61] R. Schuster, C. Bailer, O. Wasenmüller, and D. Stricker. “Combining Stereo Disparity and Optical Flow for Basic Scene Flow”. In: *Proceedings of the Commercial Vehicle Technology (CVT)*. Kaiserslautern, Germany: Springer, 2018, pp. 90–101 (cit. on p. 42).
- [62] B. K. P. Horn and B. G. Schunck. “Determining Optical Flow”. In: *Artificial Intelligence (AIJ)* 17 (3 Sept. 1981), pp. 185–203 (cit. on pp. 42, 43).
- [63] B. D. Lucas and T. Kanade. “An Iterative Image Registration Technique with an Application to Stereo Vision”. In: *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*. Vancouver, Canada: AAAI Press, Aug. 1981, pp. 674–679 (cit. on p. 42).
- [64] T. Brox and J. Malik. “Large Displacement Optical Flow: Descriptor Matching in Variational Motion Estimation”. In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 33 (3 Mar. 2011), pp. 500–513 (cit. on p. 43).
- [65] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. “DeepFlow: Large Displacement Optical Flow with Deep Matching”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Sydney, Australia: IEEE, Dec. 2013, pp. 1385–1392 (cit. on p. 43).

- [66] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black. “A Naturalistic Open Source Movie for Optical Flow Evaluation”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Vol. 7577. Firenze, Italy: Springer, 2012, pp. 611–625 (cit. on pp. 43, 53, 65).
- [67] A. Geiger, P. Lenz, and R. Urtasun. “Are we Ready for Autonomous Driving? the KITTI Vision Benchmark Suite”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Providence, USA: IEEE, June 2012, pp. 3354–3361 (cit. on p. 43).
- [68] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, et al. “FlowNet: Learning Optical Flow with Convolutional Networks”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Santiago, Chile: IEEE, Dec. 2015, pp. 2758–2766 (cit. on p. 43).
- [69] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. “FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Honolulu, Hawaii: IEEE, July 2017, pp. 1647–1655 (cit. on p. 43).
- [70] T.-W. Hui, X. Tang, and C. C. Loy. “LiteFlowNet: A Lightweight Convolutional Neural Network for Optical Flow Estimation”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City, USA: IEEE, June 2018, pp. 8981–8989 (cit. on p. 43).
- [71] T.-W. Hui, X. Tang, and C. C. Loy. “A Lightweight Optical Flow CNN —Revisiting Data Fidelity and Regularization”. In: *Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* 43 (8 Aug. 2021), pp. 2555–2569 (cit. on pp. 43, 65, 69).
- [72] Z. Teed and J. Deng. “Raft: Recurrent All-Pairs Field Transforms for Optical Flow”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Online: Springer, Aug. 2020, pp. 402–419 (cit. on pp. 43, 65).
- [73] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. “A Database and Evaluation Methodology for Optical Flow”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Vol. 92. Barcelona, Spain: IEEE, Mar. 2011, pp. 1–31 (cit. on pp. 43, 65).
- [74] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, et al. “A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Las Vegas, USA: IEEE, June 2016, pp. 4040–4048 (cit. on pp. 43, 54, 65).
- [75] D. Sun, S. Roth, and M. J. Black. “A Quantitative Analysis of Current Practices in Optical Flow Estimation and the Principles Behind Them”. In: *International Journal of Computer Vision (IJCV)* 106 (2 Jan. 2014), pp. 115–137 (cit. on p. 43).

-
- [76] D. M. Goldberg and J. R. Gott. “Flexion and Skewness in Map Projections of the Earth”. In: *Cartographica: The International Journal for Geographic Information and Geovisualization* 42 (4 Dec. 2007), pp. 297–318 (cit. on p. 44).
- [77] L. Deng, M. Yang, Y. Qian, C. Wang, and B. Wang. “CNN based Semantic Segmentation for Urban Traffic Scenes using Fisheye Camera”. In: *Proceedings of the Intelligent Vehicles Symposium (IV)*. Redondo Beach, USA: IEEE, June 2017, pp. 231–236 (cit. on p. 45).
- [78] A. Saez, L. M. Bergasa, E. Romeral, E. Lopez, R. Barea, and R. Sanz. “CNN-based Fisheye Image Real-Time Semantic Segmentation”. In: *Proceedings of the Intelligent Vehicles Symposium (IV)*. Changshu, China: IEEE, June 2018, pp. 1039–1044 (cit. on p. 45).
- [79] K. Bhandari, Z. Zong, and Y. Yan. “Revisiting Optical Flow Estimation in 360 Videos”. In: Taichung, Taiwan: IEEE, July 2021, pp. 8196–8203 (cit. on p. 45).
- [80] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niebner, M. Savva, et al. “Matterport3D: Learning from RGB-D Data in Indoor Environments”. In: *Proceedings of the International Conference on 3D Vision (3DV)*. Qingdao, China: IEEE, Oct. 2017, pp. 667–676 (cit. on p. 45).
- [81] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese. “Joint 2D-3D-Semantic Data for Indoor Scene Understanding”. In: *arXiv* abs/1702.01105 (Feb. 2017), pp. 1–9 (cit. on p. 45).
- [82] N. Zioulis, A. Karakottas, D. Zarpalas, and P. Daras. “OmniDepth: Dense Depth Estimation for Indoors Spherical Panoramas”. In: *Proceedings of the European Conference of Computer Vision (ECCV)*. Munich, Germany: Springer, Sept. 2018, pp. 453–471 (cit. on p. 45).
- [83] J. Zheng, J. Zhang, J. Li, R. Tang, S. Gao, and Z. Zhou. “Structured3D: A Large Photo-Realistic Dataset for Structured 3D Modeling”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Online: Springer, Aug. 2020, pp. 519–535 (cit. on p. 45).
- [84] G. Albanis, N. Zioulis, P. Drakoulis, V. Gkitsas, V. Sterzentsenko, F. Alvarez, et al. “Pano3D: A Holistic Benchmark and a Solid Baseline for 360° Depth Estimation”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Nashville, USA: IEEE, June 2021, pp. 3722–3732 (cit. on p. 45).
- [85] B. Coors, A. P. Condurache, and A. Geiger. “SphereNet: Learning Spherical Representations for Detection and Classification in Omnidirectional Images”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Munich, Germany: Springer, Sept. 2018, pp. 525–541 (cit. on pp. 45, 47, 50).
-

- [86] S.-H. Chou, C. Sun, W.-Y. Chang, W.-T. Hsu, M. Sun, and J. Fu. “360-Indoor: Towards Learning Real-World Objects in 360° Indoor Equirectangular Images”. In: *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*. Snowmass Village, USA: IEEE, Mar. 2020, pp. 834–842 (cit. on p. 45).
- [87] C. Fernandez-Labrador, J. M. Facil, A. Perez-Yus, C. Demonceaux, J. Civera, and J. J. Guerrero. “Corners for layout: End-to-end layout recovery from 360 images”. In: *IEEE Robotics and Automation Letters* 5 (2 Apr. 2020), pp. 1255–1262 (cit. on pp. 45, 47, 48, 50).
- [88] T. S. Cohen, M. Geiger, J. Koehler, and M. Welling. “Spherical CNNs”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. Vancouver, Canada, Apr. 2018, pp. 1–15 (cit. on p. 46).
- [89] C. M. Jiang, J. Huang, K. Kashinath, Prabhat, P. Marcus, and M. Niessner. “Spherical CNNs on Unstructured Grids”. In: *Proceedings of the International Conference on Learning Representations (ICLR)*. New Orleans, USA, May 2019, pp. 1–16 (cit. on p. 46).
- [90] Y. Lee, J. Jeong, J. Yun, W. Cho, and K.-J. Yoon. “SpherePHD: Applying CNNs on a Spherical PolyHeDron Representation of 360° Images”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, USA: IEEE, June 2019, pp. 9173–9181 (cit. on p. 46).
- [91] C. Zhang, S. Liwicki, W. Smith, and R. Cipolla. “Orientation-Aware Semantic Segmentation on Icosahedron Spheres”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Seoul, Korea: IEEE, Oct. 2019, pp. 3532–3540 (cit. on p. 46).
- [92] L. Deng, M. Yang, H. Li, T. Li, B. Hu, and C. Wang. “Restricted Deformable Convolution-Based Road Scene Semantic Segmentation Using Surround View Cameras”. In: *Transactions on Intelligent Transportation Systems (TPAMI)* 21 (10 Oct. 2020), pp. 4350–4362 (cit. on p. 46).
- [93] A. R. Sekkat, Y. Dupuis, V. R. Kumar, H. Rashed, S. Yogamani, P. Vasseur, et al. “SynWoodScape: Synthetic Surround-View Fisheye Camera Dataset for Autonomous Driving”. In: *IEEE Robotics and Automation Letters* 7 (3 July 2022), pp. 8502–8509 (cit. on p. 46).
- [94] F.-E. Wang, Y.-H. Yeh, M. Sun, W.-C. Chiu, and Y.-H. Tsai. “BiFuse: Monocular 360 Depth Estimation via Bi-Projection Fusion”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Online: IEEE, June 2020, pp. 459–468 (cit. on p. 46).
- [95] H. Jiang, Z. Sheng, S. Zhu, Z. Dong, and R. Huang. “UniFuse: Unidirectional Fusion for 360° Panorama Depth Estimation”. In: *IEEE Robotics and Automation Letters* 6 (2021), pp. 1519–1526 (cit. on p. 46).

-
- [96] Q. Feng, H. P. H. Shum, and S. Morishima. “360 Depth Estimation in the Wild - the Depth360 Dataset and the SegFuse Network”. In: *Proceedings of the Conference on Virtual Reality and 3D User Interfaces (VR)*. Christchurch, New Zealand: IEEE, Mar. 2022, pp. 664–673 (cit. on p. 46).
- [97] M. Yuan and C. Richardt. “360° Optical Flow using Tangent Images”. In: *Proceedings of the British Machine Vision Conference (BMVC)*. London, United Kingdom: BMVA, Nov. 2021, pp. 1–20 (cit. on p. 46).
- [98] Y. Li, C. Barnes, K. Huang, and F.-L. Zhang. “Deep 360° Optical Flow Estimation Based on Multi-Projection Fusion”. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. Tel Aviv, Israel: Springer, Oct. 2022, pp. 336–352 (cit. on pp. 46, 53, 54, 55, 70, 115).
- [99] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, et al. “Deformable Convolutional Networks”. In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Venice, Italy: IEEE, Oct. 2017, pp. 764–773 (cit. on p. 47).
- [100] Z. Wu, G. Allibert, C. Stolz, and C. Demonceaux. “Depth-Adapted CNN for RGB-D Cameras”. In: *Proceedings of the Asian Conference on Computer Vision (ACCV)*. Kyoto, Japan: Springer, Nov. 2020, pp. 388–404 (cit. on p. 47).
- [101] Z. Wu, G. Allibert, C. Stolz, C. Ma, and C. Demonceaux. “Modality-Guided Subnetwork for Salient Object Detection”. In: *Proceedings of the International Conference on 3D Vision (3DV)*. London, United Kingdom: IEEE, Dec. 2021, pp. 515–524 (cit. on p. 47).
- [102] Z. Wu, G. Allibert, C. Stolz, C. Ma, and C. Demonceaux. “Depth-Adapted CNNs for RGB-D Semantic Segmentation”. In: *arXiv preprint/2206.03939* (2022), pp. 1–12 (cit. on p. 47).
- [103] Z. Wu. “Depth attention for scene understanding”. PhD thesis. Université Bourgogne Franche-Comté, 2022 (cit. on p. 47).
- [104] Y.-C. Su and K. Grauman. “Learning Spherical Convolution for Fast Features from 360° Imagery”. In: *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*. Long Beach, USA, Dec. 2017, pp. 529–539 (cit. on p. 47).
- [105] Y.-C. Su and K. Grauman. “Kernel Transformer Networks for Compact Spherical Convolution”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. Long Beach, USA: IEEE, June 2019, pp. 9434–9443 (cit. on p. 47).
- [106] X. Cheng, P. Wang, Y. Zhou, C. Guan, and R. Yang. “Omnidirectional Depth Extension Networks”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. Paris, France: IEEE, May 2020, pp. 589–595 (cit. on p. 47).
- [107] H.-X. Chen, K. Li, Z. Fu, M. Liu, Z. Chen, and Y. Guo. “Distortion-Aware Monocular Depth Estimation for Omnidirectional Images”. In: *Signal Processing Letters* 28 (2021), pp. 334–338 (cit. on p. 47).
-

- [108] E. Games. *Unreal Engine*. <https://www.unrealengine.com/>. 2020 (cit. on pp. 53, 92).
- [109] M. U. GmbH. *MW Redwood Tree Forest Biome*. <https://www.unrealengine.com/marketplace/en-US/product/redwood-forest-collection>. 2022 (cit. on pp. 53, 93).
- [110] H. Shi, Y. Zhou, K. Yang, X. Yin, Z. Wang, Y. Ye, et al. “PanoFlow: Learning 360° Optical Flow for Surrounding Temporal Understanding”. In: *Transactions on Intelligent Transportation Systems* (Feb. 2022), pp. 1–15 (cit. on pp. 53, 54, 55, 70, 115).
- [111] B. O. Community. “Blender”. In: (2013) (cit. on p. 53).
- [112] A. Ranjan, D. T. Hoffmann, D. Tzionas, S. Tang, J. Romero, and M. J. Black. “Learning Multi-human Optical Flow”. In: *International Journal of Computer Vision* 128 (4 Apr. 2020), pp. 873–890 (cit. on p. 54).
- [113] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. “CARLA: An Open Urban Driving Simulator”. In: *Proceedings of the Annual Conference on Robot Learning (CoRL)*. Vol. 78. Mountain View, USA: PMLR.org, Nov. 2017, pp. 1–16 (cit. on p. 54).
- [114] R. company. *Ricoh Theta Movie Converter application*. <https://support.theta360.com/en/download/movieconverter/>. 2018 (cit. on p. 56).
- [115] FFmpeg. *FFmpeg*. <https://ffmpeg.org/download.html>. 2020 (cit. on p. 56).
- [116] C. Cadena, Y. Latif, and I. D. Reid. “Measuring the Performance of Single Image Depth Estimation Methods”. In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, Korea: IEEE, Oct. 2016, pp. 4150–4157 (cit. on p. 61).
- [117] C.-O. Artizzu, H. Zhang, G. Allibert, and C. Démonceaux. *Supplementary video results for OmniFlowNet: a Perspective Neural Network Adaptation for Optical Flow Estimation in Omnidirectional Images*. http://www.i3s.unice.fr/~allibert/Videos/icpr20_video.mp4. 2021 (cit. on p. 68).
- [118] J. Engel, T. Schöps, and D. Cremers. “LSD-SLAM: Large-Scale Direct monocular SLAM”. In: *Lecture Notes in Computer Science* 8690 LNCS (2 2014), pp. 834–849 (cit. on p. 79).
- [119] D. Scaramuzza, M. C. Achtelik, L. Doitsidis, F. Friedrich, E. Kosmatopoulos, A. Martinelli, et al. “Vision-Controlled Micro Flying Robots: From System Design to Autonomous Navigation and Mapping in GPS-Denied Environments”. In: *Robotics and Automation Magazine* 21 (3 Sept. 2014), pp. 26–40 (cit. on p. 79).
- [120] I. S. Mohamed, G. Allibert, and P. Martinet. “Model Predictive Path Integral Control Framework for Partially Observable Navigation: A Quadrotor Case Study”. In: *Proceedings of the International Conference on Control, Automation, Robotics and Vision (ICARCV)*. Shenzhen, China: IEEE, Dec. 2020, pp. 196–203 (cit. on p. 79).

-
- [121] B. Zhou, Y. Zhang, X. Chen, and S. Shen. “FUEL: Fast UAV Exploration Using Incremental Frontier Structure and Hierarchical Planning”. In: *IEEE Robotics and Automation Letters* 6 (2 Apr. 2021), pp. 779–786 (cit. on p. 79).
- [122] I. S. Mohamed, K. Yin, and L. Liu. “Autonomous Navigation of AGVs in Unknown Cluttered Environments: Log-MPPI Control Strategy”. In: *IEEE Robotics and Automation Letters* 7 (4 Oct. 2022), pp. 10240–10247 (cit. on p. 79).
- [123] M. Mancini, G. Costante, P. Valigi, and T. A. Ciarfuglia. “J-MOD 2 : Joint Monocular Obstacle Detection and Depth Estimation”. In: *IEEE Robotics and Automation Letters* 3 (3 July 2018), pp. 1490–1497 (cit. on p. 79).
- [124] A. Silva, R. Mendonça, and P. Santana. “Monocular Trail Detection and Tracking Aided by Visual SLAM for Small Unmanned Aerial Vehicles”. In: *Journal of Intelligent and Robotic Systems* 97 (3-4 Mar. 2020), pp. 531–551 (cit. on p. 79).
- [125] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. “Learning high-speed flight in the wild”. In: *Science Robotics* 6 (59 Oct. 2021), pp. 1–23 (cit. on pp. 80, 82, 96, 100).
- [126] T. Osa, J. Pajarinen, G. Neumann, J. A. Bagnell, P. Abbeel, and J. Peters. “An Algorithmic Perspective on Imitation Learning”. In: *Foundations and Trends in Robotics* 7 (1-2 2018), pp. 1–179 (cit. on p. 80).
- [127] D. K. Kim and T. Chen. “Deep Neural Network for Real-Time Autonomous Indoor Navigation”. In: *arXiv abs/1511.04668* (Nov. 2015), pp. 1–13 (cit. on p. 80).
- [128] R. P. Padhy, S. Verma, S. Ahmad, S. K. Choudhury, and P. K. Sa. “Deep Neural Network for Autonomous UAV Navigation in Indoor Corridor Environments”. In: *Procedia Computer Science* 133 (2018), pp. 643–650 (cit. on p. 80).
- [129] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, et al. “Learning Monocular Reactive UAV Control in Cluttered Natural Environments”. In: *Proceedings of the International Conference on Robotics and Automation (ICRA)*. Karlsruhe, Germany: IEEE, May 2013, pp. 1765–1772 (cit. on p. 80).
- [130] A. Giusti, J. Guzzi, D. C. Ciresan, F.-L. He, J. P. Rodriguez, F. Fontana, et al. “A Machine Learning Approach to Visual Perception of Forest Trails for Mobile Robots”. In: *IEEE Robotics and Automation Letters* 1 (2 July 2016), pp. 661–667 (cit. on pp. 80, 81).
- [131] R. S. Sutton, A. G. Barto, et al. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge, 1998, 340 p. (Cit. on pp. 81, 84, 87, 88, 89).

- [132] G. Kahn, P. Abbeel, and S. Levine. “BADGR: An Autonomous Self-Supervised Learning-Based Navigation System”. In: *IEEE Robotics and Automation Letters* 6 (2 Apr. 2021), pp. 1312–1319 (cit. on pp. 81, 83).
- [133] O. Doukhi and D.-J. Lee. “Deep Reinforcement Learning for End-to-End Local Motion Planning of Autonomous Aerial Robots in Unknown Outdoor Environments: Real-Time Flight Experiments”. In: *Sensors* 21 (7 Apr. 2021), pp. 1–18 (cit. on p. 82).
- [134] D. C. Guastella and G. Muscato. “Learning-Based Methods of Perception and Navigation for Ground Vehicles in Unstructured Environments: A Review”. In: *Sensors* 21 (1 Dec. 2020), p. 73 (cit. on p. 82).
- [135] L. Kastner, X. Zhao, T. Buiyan, J. Li, Z. Shen, J. Lambrecht, et al. “Connecting Deep-Reinforcement-Learning-based Obstacle Avoidance with Conventional Global Planners using Waypoint Generators”. In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, Sept. 2021, pp. 1213–1220 (cit. on p. 82).
- [136] T. Tezenas Du Montcel, A. Nègre, J.-E. Gomez-Balderas, and N. Marchand. “BOARR : A Benchmark for quadrotor Obstacle Avoidance based on ROS and RotorS”. May 2019 (cit. on p. 83).
- [137] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza. “Flightmare: A Flexible Quadrotor Simulator”. In: *Proceedings of the Conference on Robot Learning (CoRL)*. Online, Nov. 2020, pp. 1147–1157 (cit. on p. 83).
- [138] L. He, N. Aouf, and B. Song. “Explainable Deep Reinforcement Learning for UAV autonomous path planning”. In: *Aerospace Science and Technology* 118 (Nov. 2021), pp. 1–12 (cit. on pp. 83, 91, 96, 100).
- [139] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, et al. “Playing Atari with Deep Reinforcement Learning”. In: *Proceedings of the International Conference on Neural Information Processing Systems (NeurIPS)*. Lake Tahoe, USA, Dec. 2013, pp. 1–9 (cit. on p. 90).
- [140] S. Adam, L. Busoniu, and R. Babuska. “Experience Replay for Real-Time Reinforcement Learning Control”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 42 (2 Mar. 2012), pp. 201–212 (cit. on p. 90).
- [141] R. J. Williams. “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning”. In: *Machine Learning* 8 (3-4 May 1992), pp. 229–256 (cit. on p. 90).
- [142] S. Fujimoto, H. van Hoof, and D. Meger. “Addressing Function Approximation Error in Actor-Critic Methods”. In: *Proceedings of the International Conference on Machine Learning (ICML)*. Vol. 80. Stockholm, Sweden: JMLR, July 2018, pp. 1582–1591 (cit. on p. 91).

-
- [143] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, et al. “Continuous Control with Deep Reinforcement Learning”. In: *Proceedings or the International Conference on Learning Representations (ICLR)*. San Juan, Puerto Rico, May 2016, pp. 1–14 (cit. on p. 91).
- [144] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. “Proximal Policy Optimization Algorithms”. In: *arXiv abs/1707.06347* (July 2017), pp. 1–12 (cit. on p. 91).
- [145] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. “Trust Region Policy Optimization”. In: *Proceedings of the International Conference on Machine Learning (ICML)*. Vol. 37. Lille, France: PMLR, July 2015, pp. 1889–1897 (cit. on p. 91).
- [146] J. Xu, T. Du, M. Foshey, B. Li, B. Zhu, A. Schulz, et al. “Learning to Fly: Computational Controller Design for Hybrid UAVs with Reinforcement Learning”. In: *ACM Transactions on Graphics* 38 (4 Aug. 2019), pp. 1–12 (cit. on p. 91).
- [147] E. Bohn, E. M. Coates, S. Moe, and T. A. Johansen. “Deep Reinforcement Learning Attitude Control of Fixed-Wing UAVs Using Proximal Policy optimization”. In: *Proceedings of the International Conference on Unmanned Aircraft Systems (ICUAS)*. Atlanta, USA: IEEE, June 2019, pp. 523–533 (cit. on p. 91).
- [148] V. J. Hodge, R. Hawkins, and R. Alexander. “Deep Reinforcement Learning for Drone Navigation using Sensor Data”. In: *Neural Computing and Applications* 33 (6 Mar. 2021), pp. 2015–2033 (cit. on p. 91).
- [149] Y. Song, M. Steinweg, E. Kaufmann, and D. Scaramuzza. “Autonomous Drone Racing with Deep Reinforcement Learning”. In: *Proceedings of the International Conference on Intelligent Robots and Systems (IROS)*. Prague, Czech Republic: IEEE, Sept. 2021, pp. 1205–1212 (cit. on p. 91).
- [150] F. Aleotti, G. Zaccaroni, L. Bartolomei, M. Poggi, F. Tosi, and S. Mattocchia. “Real-Time Single Image Depth Perception in the Wild with Handheld Devices”. In: *Sensors* 21 (1 Dec. 2020), pp. 1–15 (cit. on p. 95).
- [151] N. Polosky, T. Gwin, S. Furman, P. Barhanpurkar, and J. Jagannath. “Machine Learning Subsystem for Autonomous Collision Avoidance on a small UAS with Embedded GPU”. In: *Proceedings of the Annual Consumer Communications & Networking Conference (CCNC)*. Las Vegas, USA: IEEE, Jan. 2022, pp. 1–7 (cit. on p. 95).
- [152] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, et al. “Human-Level Control through Deep Reinforcement Learning”. In: *Nature* 518 (7540 Feb. 2015), pp. 529–533 (cit. on pp. 96, 100).
- [153] S. Zhang, Y. Li, and Q. Dong. “Autonomous Navigation of UAV in Multi-Obstacle Environments Based on a Deep Reinforcement Learning Approach”. In: *Applied Soft Computing* 115 (Jan. 2022), pp. 1–13 (cit. on p. 98).

-
- [154] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, et al. “On Evaluation of Embodied Navigation Agents”. In: *arXiv abs/1807.06757* (July 2018), pp. 1–7 (cit. on p. 99).
- [155] C.-O. Artizzu, G. Allibert, and C. Demonceaux. *Supplementary video results for Deep Reinforcement Learning with Omnidirectional Images: application to UAV Navigation in Forests*. <https://www.youtube.com/@coat3943/videos>. 2022 (cit. on p. 112).
- [156] J. Zhang, K. Yang, C. Ma, S. Reiss, K. Peng, and R. Stiefelhagen. “Bending Reality: Distortion-aware Transformers for Adapting to Panoramic Semantic Segmentation”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, USA: IEEE, June 2022, pp. 16896–16906 (cit. on p. 114).
- [157] J. Zhang, K. Yang, H. Shi, S. Reiß, K. Peng, C. Ma, et al. “Behind Every Domain There is a Shift: Adapting Distortion-aware Vision Transformers for Panoramic Semantic Segmentation”. In: *arXiv abs/2207.11860* (July 2022), pp. 1–18 (cit. on p. 114).
- [158] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. “High-Resolution Image Synthesis With Latent Diffusion Models”. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. New Orleans, USA: IEEE, June 2022, pp. 10684–10695 (cit. on p. 115).
- [159] L. Zhang and M. Agrawala. “Adding Conditional Control to Text-to-Image Diffusion Models”. In: *arXiv abs/2302.05543* (Feb. 2023), pp. 1–33 (cit. on pp. 115, 116).