OmniDRL: Robust Pedestrian Detection using Omnidirectional Cameras and Deep Reinforcement Learning

Gonçalo José Dias Pais

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Dr. Pedro Daniel dos Santos Miraldo
Prof. Jacinto Carlos Marques Peixoto do Nascimento

Examination Committee

Chairperson: Prof. João Fernando Cardoso Silva Sequeira
Supervisor: Dr. Pedro Daniel dos Santos Miraldo
Member of the Committee: Prof. Alexandre José Malheiro Bernardino

May, 2018

OmniDRL: Robust Pedestrian Detection using Omnidirectional Cameras and Deep Reinforcement Learning

Gonçalo José Dias Pais

May, 2018

Declaration:

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the *Universidade de Lisboa*.

Declaração:

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Abstract

Pedestrian detection is one of the most explored topics in Computer Vision and Pattern Recognition. The increase use of deep learning methods in the Computer Vision community allowed the development of new and highly competitive algorithms for object detection, particularly for pedestrian detection. A class of such methods is based on deep Reinforcement Learning. However, for object detection in omnidirectional camera systems the literature is still scarce, due to the complexities associated with their high distortions. Nonetheless, these systems, namely catadioptric imaging devices are strongly beneficial for various applications (ranging from video surveillance to perception in robotics). In this thesis, I present a novel efficient technique for robust pedestrian detection, using omnidirectional catadioptric camera systems with deep Reinforcement Learning. My method is tested and compared with current related approaches that do not consider the underlying distortions aforementioned. Besides the novel formalism presented herein, my method improves significantly the results when compared with state-of-the-art methodologies.

Keywords: object detection, omnidirectional vision, central catadioptric camera systems, convolutional neural networks, reinforcement learning.

Resumo

A detecção de pedestres é um dos tópicos mais explorados em Visão por Computador. O aumento do uso de *deep learning* em Visão por Computador nos últimos anos permitiu o aumento de novos algoritmos para detecção de objectos e em particular em detecção de pedestres. Uma das principais classe de algoritmos baseia-se na utilização de *deep learning* em *Reinforcement Learning*. No entanto, a detecção de objectos em sistemas omnidireccionais é um tema pouco investigado na literatura, dada a existência de complexidades associadas à distorção. Apesar disso, estes sistemas são muito úteis para diferentes tipos de aplicações, como por exemplo, desde sistemas de vigilância a percepção em robótica. Nesta dissertação, eu apresento uma nova técnica para uma robusta detecção de pedestres em sistemas catadióptricos centrais usando *deep Reinforcement Learning*. O meu método é testado e comparado com outras técnicas que não consideram os problemas associados com a distorção. Para além do novo formalismo apresentado, os resultados deste novo método superam os do Estado da Arte.

Palavras-chave: detecção de objectos, visão omnidirecional, sistemas catadióptricos centrais, *convolutional neural networks*, *deep learning, reinforcement learning.*

Acknowledgements

First, I would like to thank my parents and my sister for all the support along the years. Then, to my supervisors Pedro Miraldo and Jacinto Nascimento for all the availability, help and patience, through the last year. I want to give a special thanks to my ISR group, constituted by André Mateus, Tiago Dias, João Campos and Francisco Maria. With them, the daily routine was easier and always fun. A thank you for all my friends from IST, NEEC, Erasmus, and Candeia, that were a big part of my life, at some point, during the course of this degree. At last, an enormous thank you to *Curso 2004* from Colégio Militar, specially Tiago, Afonso, António, and Emerson, that were always along side me all these years.

Contents

List of Figures **xii**

List of Tables **xiii**

Acronyms **xvii**

1 Introduction **1**

1.1 State-of-the-Art 2

1.1.1 Object Detection and Classification 3

1.1.2 Omnidirectional Vision 4

1.2 Contributions and Outline of the Thesis 7

2 Line Segment Projection using Omnidirectional Cameras **9**

2.1 Image Formation 9

2.2 3D Line Segment Projection 11

3 Deep Reinforcement Learning and Classification **13**

3.1 Deep Reinforcement Learning 13

3.1.1 Training 14

3.1.2 Inference 17

3.2 Classification 18

4 Robust Deep RL Pedestrian Detection for Omnidirectional Cameras **21**

4.1 Bounding Box Projection 21

4.1.1 Bounding Boxes' Corner Points Computation 23

4.2 Actions and Rewards 24

4.3 Multi-task Training 27

5 Experimental Results **31**

5.1 Dataset Creation 31

5.2 Preliminary Results 34

5.3 Final Results 38

6 Conclusion **41**

A Intersection over Union Computation **51**

List of Figures

1.1 Network Representation. 6

2.1 Image Formation for 3D points and 3D lines. 10

4.1 Environment illustration. 22
4.2 Examples of states. 23
4.3 Bounding Box's top view. 24
4.4 Representation of $\mathbf{A}_t$ and $\mathbf{A}_g$. 26

5.1 Dataset examples and target examples for Ours. 32
5.2 Examples of the ground truth for SotA and SotAU. 34
5.3 Frequency of trigger per steps and example of first test solution. 36
5.4 Steps illustration. 37
5.5 Histograms of rewards. 40

List of Tables

2.1 Geometry of a central catadioptric system. 11
2.2 Unified parameters for central catadioptric mirrors. 11

4.1 Update rules for position and dimension. 25

5.1 Methods comparison for the Proof of Concept. 35
5.2 Position Errors. 38
5.3 Comparison between both my methods and correct detection percentage in testing. 39

Acronyms

CNNs Convolutional Neural Networks. 1, 2, 13, 18

DDQN Double DQN. 15

DL Deep Learning. 1

DQN Deep Q-Network. 3, 5, 13, 14, 15, 17, 27, 28, 38

FoV Field of View. 1, 4

GPUs Graphical Processing Units. 1

HOG Histogram of Gradients. 5

IoU Intersection over Union. 25, 35, 39, 51

PD Pedestrian Detection. 2

RL Reinforcement Learning. 1, 2, 3, 5, 7, 13, 15, 18, 19, 21, 27, 28, 34, 38

Chapter 1

Introduction

In computer vision, object detection has been one of the most relevant research topics addressed in the last two decades. We have observed a significant improvement in the development of new algorithms in different areas, *e.g.* detection and classification (Ren, Li & Fraser (2015)), along with the availability of larger image datasets (such as Russakovsky et al. (2015), Lin et al. (2014), Everingham et al. (2010), Mottaghi et al. (2014)). Nowadays, pedestrian analysis is mainly addressed using Deep Learning (DL) techniques (Liu et al. (2017)) which attempt to learn high level abstractions of the data. It is a technique that is used in quite diverse problems, such as: Semantic Parsing (Bordes et al. (2012)); Transfer Learning (Ren & X (2015)); Natural Language processing (Mikolov et al. (2013)); and Computer Vision (Krizhevsky et al. (2012)) in which object detection is included. Two main reasons contribute to the wide spread of DL methodologies: 1) the number of large datasets; and 2) the increase of hardware capabilities, based on Graphical Processing Units (GPUs). One of the most important and well-known techniques in DL is the Convolutional Neural Networks (CNNs) (Krizhevsky et al. (2012)), which halved the error rate for image classification. Other related techniques and extensions are also available, *e.g.* He et al. (2014), Girshick et al. (2014), Girshick (2015), Ren, He, Girshick & Sun (2015), Liu et al. (2016), Redmon & Farhadi (2018). Very recently, there have been some alternative methods, namely the use of deep Reinforcement Learning (RL), where a policy is created in order to maximize the rewards of an agent (Mnih et al. (2015), Caicedo & Lazebnik (2015)).

In the context of omnidirectional camera system, we also have assisted to an increasing number of applications, ranging from surveillance to medical imaging applications

(Nalwa (1996), Nayar & Baker (1997), Boult et al. (1999), Wang & Lin (2009), Iraqui et al. (2010), Neves et al. (2011), Lourenço et al. (2014), Bergen & Wittenberg (2016), Zhang et al. (2016), Miraldo et al. (2018)), which require a wide Field of View (FoV) of the environment. The respective wide FoV is usually obtained by means of distortion (Swaninathan et al. (2003)).

For Pedestrian Detection (PD) using these type of sensors, one could first apply undistortion techniques to get a perspective image, and then conventional detection tools (Geyer & Daniilidis (2000), Barreto & Araujo (2001), Mei & Rives (2007)). However, these undistortion procedures suffer from the following shortcomings:

1. They are too computationally expensive, specially when considering large image sizes;

2. They generate images whose pedestrians cannot fit properly in regular bounding boxes[1]; and

3. They introduce artifacts in the undistorted image, that will affect the detection accuracy.

In this thesis, I overcome the above mentioned issues by performing the PD directly in distorted images, that is, I do not use images without distortions, often acquired with perspective cameras. More specifically, I revisit the deep RL in a new context, *i.e.* to perform pedestrian detection in highly distorted scenarios using omnidirectional vision systems. The problem formulated herein considers solutions that implicitly take into account distortion on the detection.

1.1 State-of-the-Art

This section revises related approaches in object detection and in omnidirectional vision systems.

[1]Although undistorted images keep the perspective projection constraints (*i.e.* straight lines in the world will be projected into straight lines in the image), the objects will be stretched. This means that the objects should not be approximated by regular bounding boxes, which are used in most of the DL techniques for object detection (we note that there are alternatives that do not consider regular bounding boxes in Jaderberg et al. (2015)).

1.1.1 Object Detection and Classification

Object detection has a wide range of applications comprising quite diverse research areas, such as artificial intelligence, video surveillance and multimedia systems. This problem has seen a significant progress in the last decade. If we look at the performance of algorithms in common datasets, *e.g.* PASCAL VOC (Everingham et al. (2012)), we can easily conclude that the progress slowed from 2010 onward. However, the use of CNNs (Krizhevsky et al. (2012)) has led to a significant improvement in the accuracy of image classification, for the Imagenet Large Scale Visual Recognition Challenge (Deng et al. (2012, 2009)). Soon, more methodologies became available, SppNet (He et al. (2014)), R-CNN (Girshick et al. (2014), Hosang et al. (2014)), and its descendants, namely, Mask (He et al. (2017)), Fast (Girshick (2015)), Faster R-CNN(Ren, He, Girshick & Sun (2015)), SSD (Liu et al. (2016)) and YOLOv3 (Redmon & Farhadi (2018)). The latter four methods were proposed to speed-up the running time of the R-CNN, given the large number of detected proposals per image. These methods (Liu et al. (2016), Redmon & Farhadi (2018)) can avoid high computational cost providing real-time solutions.

Although the methods presented above are valuable contributions in the field, I follow a different approach based on deep RL, mainly because it significantly reduces the inference time for object detection when compared to an exhaustive search (Caicedo & Lazebnik (2015), Maicas et al. (2017)). Caicedo & Lazebnik (2015) have proposed the use of a Deep Q-Network (Mnih et al. (2015)) for object detection, enabling the use of a small amount of training data without compromising its accuracy. The Q-Network has also been used in other domains, such as medical imaging analysis, *e.g.* Ghesu et al. (2016), Maicas et al. (2017). However, the above frameworks have not been explored to model images of objects/pedestrians with high distortion such as the ones encountered in omnidirectional based systems.

All the above classification and detection task algorithms use a joint learning approach. Since the classification task is highly dependent on the bounding box proposal, the presented detection algorithms use a multi-task network that learns both tasks at the same time. This reduces the computational time and increases classification & detection accuracies. This approach has been used in other Computer Vision problems, *e.g.* Semantic Segmentation, for instance detection and depth mapping (Kendall et al.

(2018)), Surface Modeling & Segmentation (Eigen & Fergus (2015)), and even 3D Pose Estimation (Kendall et al. (2015), Arandjelovic et al. (2016)). However, making a joint deep RL and classification multi-task network with a collaborative loss, as the above works propose, is unfeasible in my context. Thus, for each image a set of actions must be learned, instead of a direct proposal for the object's region that helps classify the object. Using a multi-task method in RL is unusual, however it has been useful to overcome forgetful actions over time (Kirkpatrick et al. (2017)). Alternatively, I propose a hard[2] parameter sharing network (Baxter (1997)). In the proposed methodology, each sub-network is trained alternatively, *i.e.* one at each time. So far, this strategy has been only used in the context of natural language processing (Collobert & Weston (2008)). With the proposed solution, the framework is able to retrieve the pedestrian's features for both tasks and, at the same time, speed up the training process (Thrun (1996)).

1.1.2 Omnidirectional Vision

Omnidirectional systems can achieve a wide FoV by two distinct camera system configurations:

1. by using special types of lenses (dioptric cameras); and

2. by combining mirrors with perspective cameras (catadioptric cameras).

The former uses dioptric lenses, *e.g.* fish-eye, instead of conventional ones (Kannala & Brandt (2006)). The latter uses mirror(s) such as parabolic, elliptical, or hyperbolic to obtain a larger FoV (Nalwa (1996), Nayar (1997), Nayar & Baker (1997), Miraldo et al. (2018)). Image formation for these systems has been studied in the literature for more than 20 years. In Baker & Nayar (1999), it was described the necessary conditions to ensure that a catadioptric system is a central camera, *i.e.* share some constraints with the central perspective camera (namely, the preservation of the effective view point). Otherwise, these cameras are non-central (Swaninathan et al. (2003)).

A straightforward approach to build an omnidirectional camera would be a setup with multiple perspective cameras, to increase the FoV. However, this would require finding correspondences between features and merging images from the different cameras, which

[2]The term hard means that the first convolution layers are shared, and then a branch is created for each sub-network.

requires a lot of computational effort. In addition, by merging images from different views, details and properties of the environment are lost (Schechner & Nayar (2001)), which in an omnidirectional system will not happen.

A small number of authors addressed the object/person detection directly on omnidirectional images. The works that are most related to my goals are: Cinaroglui & Bastanlar (2014, 2016), in which they adopt the conventional camera approach that uses sliding windows and Histogram of Gradients features. Since the shape of the sliding window depends on the position of the person w.r.t. the camera, the HOG filters are trained with perspective cameras and then changed to account for the distortion (by modifying the gradient magnitudes using Riemannian metric and converting the gradient orientations to form an omnidirectional non-rectangular sliding window). Although this method does not require the omnidirectional image unwrapping (avoiding expensive computation resources), it will suffer from the artifacts caused by changes in the resolution. By introducing the deep RL in omnidirectional images (OmniDRL), as proposed in this thesis, I avoid the above shortcomings.

Although there have been some advances in feature extraction for omnidirectional catadioptric systems (*e.g.* Jamzad et al. (2007), Puig & Guerrero (2011), Lourenco et al. (2012)), this is still a difficult task to be accomplished due to the distortion on the image. A feature in this imaging device depends on its position on the image (due to distortion), that is why the traditional feature extraction methods are not suited for omnidirectional systems. Some methods exists for soccer robots using catadioptric systems (Neves et al. (2011)) but require knowledge of the object shape and color. There are also tracking methods for omnidirectional cameras based on object's motion, by using background subtraction (Yamazawa & Yokoya (2003)) and the object's egomotion (Gandhi & Trivedi (2005)).

Other works address this problem by first applying transformations to the images followed by conventional techniques, *e.g.* Wang & Lin (2009), Iraqui et al. (2010). In Tang et al. (2011), the authors search for objects directly in omnidirectional images, but do not consider the underlying distortion. The robustness of affine co-variant features as a function of distortion in the image is analyzed in Furnari et al. (2017).

This work does not aim at performing the extraction of features in images of omnidirectional cameras. Instead, the goal is to define regions of interest as a function of the pedestrian position, in the presented case it is considered 3D world position. Also,

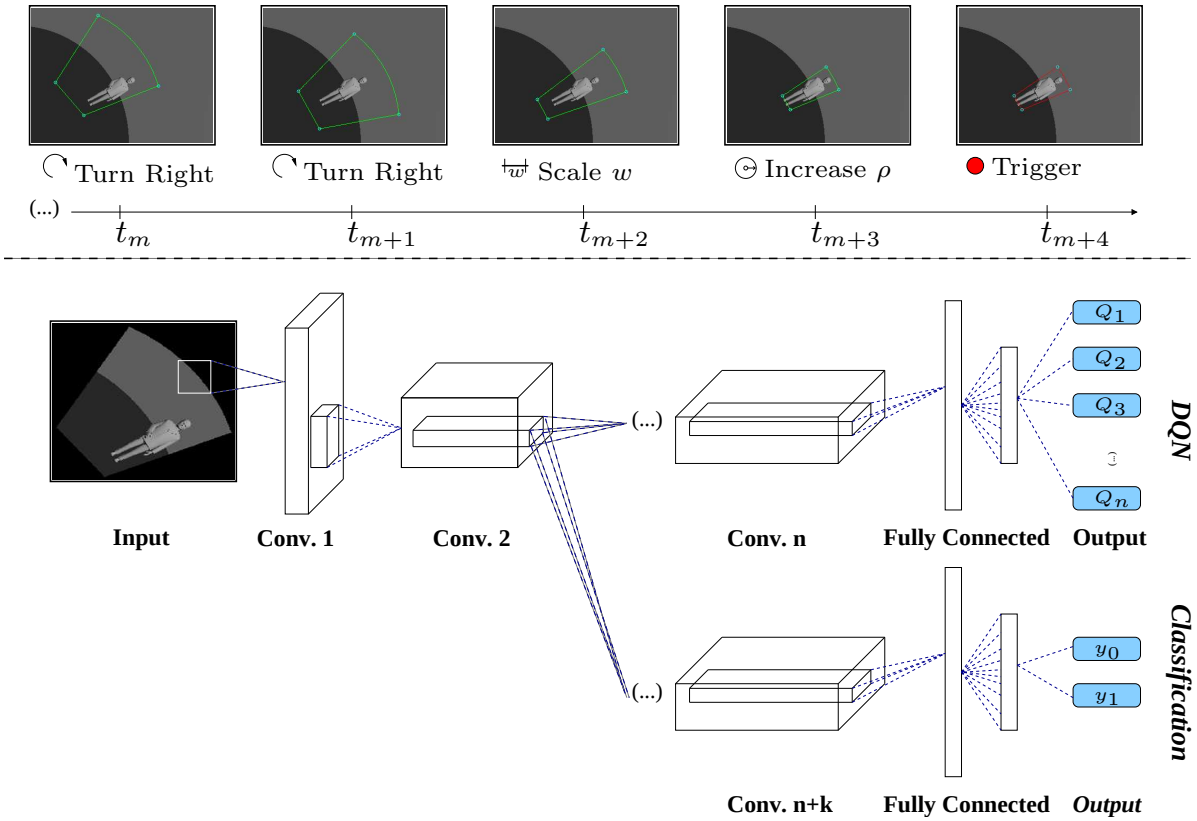


Figure 1.1: Representation of the problem addressed in this thesis, and the proposed solution for the RL using omnidirectional cameras. At the top, it is presented the results of the proposed technique to a dataset of synthetic images (I am showing only the final steps of the method); and, at the bottom, it is shown the scheme of the proposed network, where the first convolutional layers are shared, and then split into branches (DQN and Classification).

it will be introduced a novel PD in omnidirectional cameras, inspired in previous DQN methodologies and classification tasks. The proposed approach comprises an artificial agent (*i.e.* DQN agent) that can automatically learn policies directly from high dimensional data, building a deep learning feature representation of the current bounding box, which is then used by the DQN to decide on the next action. In this case, the actions can be a translation, rotation, or scale. Finally, a trigger signals the end of the detection process. A multi-task learning is implemented to assist the DQN training and make the test an end-to-end solution. This iterative process starts from an initial bounding box, signaled by a classification network, that covers a large area in the image, to a tight bounding box that contains the pedestrian. See Fig. 1.1 for an illustration of the

proposed method in a synthetic example and the network's representation.

1.2 Contributions and Outline of the Thesis

In this thesis, I propose a novel technique for pedestrian detection using omnidirectional cameras. My approach requires the understanding of the underlying projection of a world's bounding box through an omnidirectional image, and a learned agent that acts on the world environment until it detects the pedestrian.

In Chp. 2, I present the image formation and the 3D line projection representation for central catadioptric imaging devices. Then, it is presented the deep RL agent, whose training and a classification networks are described in Chp. 3. The classification network identify the presence of a pedestrian in the distorted bounding box. Having defined the projection model (notice that bounding boxes are defined by four line segments) and the agent, I propose the OmniDRL in Chp. 4. The proposed method is evaluated in Chp. 5 on a new dataset that was acquired and labeled. Finally, Chp. 6 presents the conclusions and future work.

Chapter 2

Line Segment Projection using Omnidirectional Cameras

This section presents the image formation for a central omnidirectional camera. I start by defining the image formation model (Sec. 2.1), and then the projection of 3D line segments, required to define the bounding box projection (Sec. 2.2).

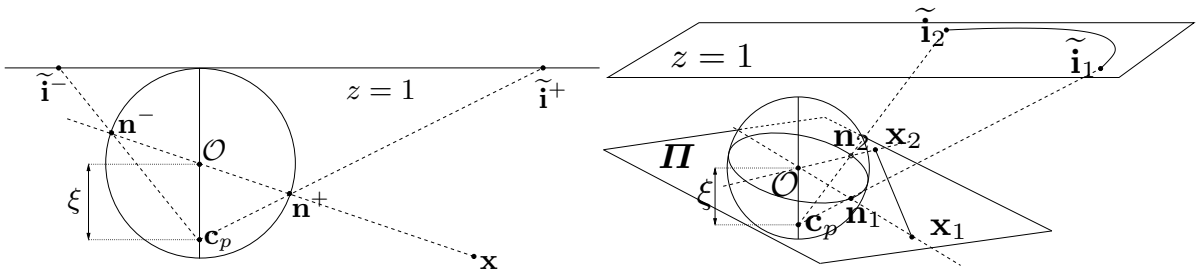
2.1 Image Formation

To deal with general omnidirectional cameras, it is used the spherical model firstly proposed by Geyer & Daniilidis (2000) and modified by Barreto & Araujo (2001) and Mei & Rives (2007). Notice that, this model can also be used to represent fisheye cameras (Ying & Hu (2004), Mei & Rives (2007)). Assuming a unit sphere centered at the origin of the of mirror's reference frame, a 3D point $\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3$ is projected onto the sphere's surface (point $\mathbf{n} \in \mathcal{S}^2 \subset \mathbb{R}^3$), resulting in a pair of antipodal points:

$$\left\{\mathbf{n}^+, \mathbf{n}^-\right\} \doteq \Omega(\mathbf{x}) = \left(\pm\frac{x_1}{r}, \pm\frac{x_2}{r}, \pm\frac{x_3}{r}\right), \tag{2.1}$$

where $r = \sqrt{x_1^2 + x_2^2 + x_3^2}$, (see Fig. 2.1(a) for more details). The antipodal point closer to $\mathbf{x}$ (*i.e.* $\mathbf{n}^+$ by the convention shown in the figure) is chosen to be the point projected to the image plane (which is true according to the *Fermat's* principle Born & Wolf (1970)). Afterwards, the reference frame is changed and it will be centered at $\mathbf{c}_p = (0, 0, \xi)$. In the new reference frame, the point $\mathbf{n}^+$ will be given by the function $\mathcal{H} : \mathbb{R}^3 \to \mathbb{R}^3$:

$$\mathbf{n}_p^+ = \mathcal{H}(\mathbf{x}, \xi) = \left(\frac{x_1}{r}, \frac{x_2}{r}, \frac{x_3 + \xi}{r}\right). \tag{2.2}$$



(a) Image Formation (b) Projection of 3D Lines

Figure 2.1: This figure shows the image formation using unified central catadioptric cameras. In (a), it is shown the projection of a point $\mathbf{x} \in \mathbb{R}^3$ onto the normalized image plane $\{\widetilde{\mathbf{i}}^-,\ \widetilde{\mathbf{i}}^+\}$ (in which there is an intermediate projection on the unitary sphere $\{\mathbf{n}^-,\ \mathbf{n}^+\}$). (b) shows the projection of 3D straight line segments for images using this model ($\mathbf{x}_1$ and $\mathbf{x}_2$ are the edges of the line's segment).

Then, the sphere surface's point $\mathbf{n}^+$ will be projected to the normalize plane $z = 1$ mapped by the function $\mathcal{F} : \mathbb{R}^3 \to \mathbb{R}^2$, such that:

$$\widetilde{\mathbf{i}} = \widetilde{\mathbf{i}}^+ = \mathcal{F}(\mathbf{x}, \xi) = \left(\frac{n^+_{p,1}}{n^+_{p,3}}, \frac{n^+_{p,2}}{n^+_{p,3}} \right). \tag{2.3}$$

Finally, the resulting point has to be mapped on the image plane through the camera projection parameters:

$$\mathbf{i} = \begin{bmatrix} f_1\eta & f_1\eta\alpha \\ 0 & f_2\eta \end{bmatrix} \widetilde{\mathbf{i}} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}, \tag{2.4}$$

where f_1 and f_2 are the focal lengths, u_0 and v_0 are the coordinates of the principal point in the image, and α is the skew parameter (for more details see Hartley & Zisserman (2003)). Finally, η is the distance from the center of the first reference frame $\mathcal{O}$ to the image plane. This transformation only occurs at the end, in order to simplify the mapping of the point from the unit sphere to the image plane.

From this final result, one can see that the projection of a point to the image plane is a function of the parameters ξ and η, which depend on the mirror's geometry and the camera parameters, which depends on the camera. Notice that the duplet of parameters (ξ, η) is what characterizes the central geometry of the cameras (Tab. 2.1), which are shown in the Tab. 2.2, where $4p$ is the *latus rectum* and d the distance between the two focal points in hyperboloid and ellipsoid mirrors.

Table 2.1: This table presents the equations that central catadioptric mirrors (paraboloid, hyperboloid, ellipsoid and planar) must fulfill in order to be modelled as central system, *i.e.* verifies the effective viewpoint constraint.

Paraboloid	$\sqrt{x_1^2 + x_2^2 + x_3^2} = x_3 + 2p$
Hyperboloid	$\frac{\left(x_3+\frac{d}{2}\right)^2}{a^2} - \frac{x_1^2}{b^2} - \frac{x_2^2}{b^2} = 1$
Ellipsoid	$\frac{\left(x_3+\frac{d}{2}\right)^2}{a^2} + \frac{x_1^2}{b^2} + \frac{x_2^2}{b^2} = 1$
Planar	$x_3 = -\frac{d}{2}$
With '-' for the hyperboloid and '+' for Ellipsoid mirrors $a = \frac{1}{2\left(\sqrt{d^2+4p^2}\pm 2p\right)} \quad b = \sqrt{p\left(\sqrt{d^2+4p^2} \pm 2p\right)}$	

Table 2.2: This table presents the unified model parameters that parameterize central catadioptric mirrors to a general spherical projection.

	ξ	η
Paraboloid	1	-2p
Hyperboloid	$\frac{d}{\sqrt{d^2+4p^2}}$	$\frac{-2p}{\sqrt{d^2+4p^2}}$
Ellipsoid	$\frac{d}{\sqrt{d^2+4p^2}}$	$\frac{2p}{\sqrt{d^2+4p^2}}$
Planar	0	-1

2.2 3D Line Segment Projection

In central camera models, all 3D lines in a plane that pass through the origin of the camera coordinate system will have the same image (interpretation plane – Hartley & Zisserman (2003)). Therefore, since I am interested in central omnidirectional cameras, 3D straight lines are parameterized by a plane $\mathbf{\Pi}$, which comprises by the center of the projection sphere ($\mathcal{O}$ in Fig. 2.1(b)) and the respective 3D straight line (notice that, without lost of generality, all lines on this plane will be equally parameterized). By definition, considering the above parameterization, for two 3D points on the line (say $\mathbf{x}_1$ and $\mathbf{x}_2$), one can represent the line using $\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2 \in \mathbb{R}^3$ (line's moment), and every point $\mathbf{x}$ belonging to the line must verify: $\mathbf{l}^T\mathbf{x} = 0$. To parameterize the projection of the 3D line onto the image plane, it is followed the technique proposed by Barreto & Araujo (2001). Thus, the image of a 3D straight line on the normalized plane must also belong to a quadric that is defined by the intersection of the plane $\mathbf{\Pi}$ and the unit sphere (see (2.1)). See Fig. 2.1(b)) for more details. After some manipulations, one can obtain

a quadric equation $\mathbf{x}^T\mathbf{C}\mathbf{x} = 0$, where:

$$\mathbf{C} = \begin{bmatrix} l_1^2\,(1-\xi^2) - l_3^2\xi^2 & l_1 l_2\,(1-\xi^2) & l_1 l_3 \\ l_1 l_2\,(1-\xi^2) & l_2^2\,(1-\xi^2) - l_3^2\eta^2 & l_2 l_3 \\ l_1 l_2 & l_2 l_3 & l_3^2 \end{bmatrix}, \tag{2.5}$$

represents the curve in the sphere. To get a representation of the curve in the image plane, we have to consider the constraint $z = 1$, *i.e.*, the projection to the normalized plane.

To conclude, one can define the projection curve (image of the line's segment) in this plane by:

$$\begin{aligned} \mathcal{L} = \mathcal{G}(\mathbf{x}_1, \mathbf{x}_2) = \{\ (\widetilde{i}_x, \widetilde{i}_y) \in \mathbb{R}^2\ :\ & \begin{bmatrix} \widetilde{i}_x & \widetilde{i}_y & 1 \end{bmatrix}^T \mathbf{C} \begin{bmatrix} \widetilde{i}_x & \widetilde{i}_y & 1 \end{bmatrix} = 0 \wedge \\ & \widetilde{i}_{1,x} < \widetilde{i}_x < \widetilde{i}_{2,x} \wedge \\ & \widetilde{i}_{1,y} < \widetilde{i}_y < \widetilde{i}_{2,y}\ \}, \end{aligned} \tag{2.6}$$

where $(\widetilde{i}_{1,x}, \widetilde{i}_{1,y}) = \mathcal{F}(\mathbf{x}_1, \xi)$ and $(\widetilde{i}_{2,x}, \widetilde{i}_{2,y}) = \mathcal{F}(\mathbf{x}_2, \xi)$ (see (2.3) and also Fig. 2.1(b)). Finally, to the points that belong to the curve, a final mapping must be applied to project them to the image plane, which is computed by applying (2.4).

Chapter 3

Deep Reinforcement Learning and Classification

In this chapter, I present the principles behind the training of a deep RL agent and a classification network. First, I define the agent's training, with different exploration methods, and how it infers its actions, in Sec. 3.1, and then I give an overview on the classification task using CNNs in Sec. 3.2.

3.1 Deep Reinforcement Learning

This section describes how object detection is performed using deep RL which has been a promising and thriving path of research. Very recently, Caicedo & Lazebnik (2015), Mnih et al. (2015) have proposed the use of a Q-network as a way to improve the object detection efficiency. The challenge herein is to adapt the above methodology for the pedestrian detection in omnidirectional cameras which inherently holds high distortion in the observations (represented by bounding boxes). The aforementioned issue has never been tackled in the deep RL context in the sense that the visual classes used until now (*e.g.* Caicedo & Lazebnik (2015)) have consistent patterns. In this problem, the above consistency decreases substantially, since now the texture, appearance, location, as well as background are more challenging to deal with. Two basic concepts characterize the deep RL framework: an artificial agent and a policy. Basically, the agent automatically learns a policy, and performs subsequent actions that allow to iteratively modify the focus of attention from an initial and large bounding box to a tight bounding box containing the target (*i.e.* the pedestrian), that is accomplished by finding an

optimal policy. This is achieved by obtaining a high level feature representation of the current state (*i.e.* DQN) constructed by the agent that enables to decide further actions, either by adjusting the bounding box or enabling the trigger, thus ending the detection process.

Next, we define the training of the above DQNs.

3.1.1 Training

The training process of the DQN follows a traditional Markov Decision process that is accomplished by a sequence of *state observations* "s", *actions* "a" and *rewards* "r". Each state observation can be formally defined by $s_t = \mathcal{I}(\mathbf{p}_t))$, where $\mathbf{p}_t$ is the pedestrian position at time instant t in the image $\mathcal{I}(.)$.

A good strategy for an agent would be to always choose an action that maximizes the amount of rewards, *i.e.* $R_t = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}$, where γ is a discount factor. A Value-Action Function $Q(s, a)$ that represents the maximum discounted future reward when performing a certain action a in given state s can be defined formally as:

$$Q^{\star}(s, a) = \max_{\pi} \mathbb{E}[R_t | s_t = s, a_t = a, \pi], \tag{3.1}$$

which is achievable by a policy $\pi = p(a|s)$, and $Q^{\star}(.)$ representing the *quality* for performing an action a in the state s. In practice, obtaining $Q^{\star}(s, a)$ can be achieved using the *Bellman* equation and the Q-Learning algorithm (see Sutton & Barto (1998) for more details). The intuition is that, if the optimal value $Q^{\star}(s_{t+1}, a_{t+1})$ at the next time-step was known for all possible actions a_{t+1}, then the optimal strategy would be to select the action a_{t+1} maximizing the expected value of $r_t + \gamma Q^{\star}(s_{t+1}, a_{t+1})$, that is $Q^{\star}(s, a) = \mathbb{E}_{s_{t+1}}[r + \gamma Q^{\star}(s_{t+1}, a_{t+1})]$. This goal is achieved by computing the *Bellman* equation in an iterative fashion, *i.e.* by following a value-iteration algorithm:

$$Q(s_t, a_t; \boldsymbol{\theta}_t) = \mathbb{E}_{s_{t+1}}\left[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \boldsymbol{\theta}_t) | s_t, a_t \right] \tag{3.2}$$

In practice the value-iteration that characterizes Q-learning is limited in the number of actions and states, and cannot generalize to unobserved states. To overcome the above issue, the Deep Q-Learning is proposed, where a parameter-function is used to approximate the Q-function. Formally, one has $Q(s_t, a_t; \boldsymbol{\theta}_t) = Q^{\star}(s_t, a_t)$. The action-valued function is modeled by a DQN, $Q(s_t, a_t; \boldsymbol{\theta}_t)$, where $\boldsymbol{\theta}_t$ denotes the weights of the

prediction network. The training of the DQN is based on *experience-replay* memory and the target function (Mnih et al. (2015)). This process makes use of a memory $\mathbf{D}_t = \{e_1, ..., e_t\}$ that is built with the agent's experiences $\mathbf{e}_t = (s_t, a_t, r_t, s_{t+1})$. These samples are drawn uniformly from the memory and will be used as a batch to train the prediction network. The target network containing the parameters $\boldsymbol{\theta}_t^-$ computes the target values that allows the DQN updates. These values $\boldsymbol{\theta}_t^-$ are held unchanged and updated periodically.

The problem with the Q-learning and DQN approaches is that the action selection and the evaluation use the same Q-values, leading to overoptimistic estimates (Thrun & Schwartz (1993)). However, Hasselt (2010) introduced Double Q-learning, that was later applied in the context of deep learning, giving rise to the so called Double DQN, in Hasselt et al. (2016), in order to overcome this issue. Now, the prediction network selects the action that maximizes the Q-value (greedy policy), and the target network estimates its value (Hasselt et al. (2016)). The goal is to change the DQN algorithm as minimum as possible, *i.e.* the training algorithm (detailed in Alg. 1) remains the same, but incorporates the new update rule. To be more specific, in the DQN, I have:

$$q_j^{DQN} = r_j + \gamma \max_{a_{j+1}} Q(\mathbf{s}_{j+1}, a_{j+1}; \boldsymbol{\theta}_t^-), \tag{3.3}$$

which is now replaced by:

$$q_j^{DDQN} = r_t + \gamma Q(s_{t+1}, \arg\max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t^-). \tag{3.4}$$

Therefore, the loss function that models $Q(s_t, a_t; \boldsymbol{\theta}_t)$ minimizes the following mean squared error of the modified version of the *Bellman* equation:

$$L_{drl}(\boldsymbol{\theta}_t) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim U(\mathbf{D}_t)} \left[(q_j^{DDQN} - Q(s_t, a_t; \boldsymbol{\theta}_t))^2 \right], \tag{3.5}$$

where γ is the discount factor, $\boldsymbol{\theta}_t$ are the parameters of the DDQN at iteration t, $\boldsymbol{\theta}_t^-$ are used to compute the target at iteration t and $U(\mathbf{D}_t)$ the batch retrieved from the memory $\mathbf{D}_t$ with uniform distribution U. As mentioned, the target parameters $\boldsymbol{\theta}_t^-$ are updated periodically (every C steps that is later defined) from the parameters $\boldsymbol{\theta}_t$ on the iteration $t-1$ and maintained constant between periods. The initial parameters $\boldsymbol{\theta}_0$ are initialized randomly and $\boldsymbol{\theta}_0^- = \boldsymbol{\theta}_0$.

Algorithm 1 Deep RL training with an ϵ-greedy exploration policy and Double Q-Learning (Hasselt et al. (2016)).

Initialize replay memory D to capacity N
Initialize action-value function $Q(\mathbf{s}_t, a_t; \boldsymbol{\theta}_0)$ with random weights $\boldsymbol{\theta}_0$
Initialize target action-value function $Q(\mathbf{s}_t, a_t; \boldsymbol{\theta}_0^-)$ with weights $\boldsymbol{\theta}_0^- = \boldsymbol{\theta}_0$
for $t = 1, M$ **do**
  Select random image from the dataset
  Initialize $\mathbf{p}_0$ and $\mathbf{dim}_0$
  while $terminal == False$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = argmax_a Q(\mathbf{s}_t, a_t; \boldsymbol{\theta}_t)$
    Execute action a_t in environment and observe the reward r_t and state s_{t+1}
    Store in $\mathbf{D}_t$ the experience $\mathbf{e}_t = (\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$
    Sample random minibatch of transitions $\mathbf{e_j}$ from $\mathbf{D}_t$
    set $q_j^{DDQN} = \begin{cases} r_j & \text{if episodes ends at step } j+1 \\ r_j + \gamma Q(s_{j+1}, \arg\max_{a_{j+1}} Q(s_{j+1}, a_{j+1}; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t^-) & \text{otherwise} \end{cases}$
    Perform gradient descent on $L_{drl}(\boldsymbol{\theta}_t)$ (3.5) with the respect of the parameters $\boldsymbol{\theta}_t$
    Decrease current ϵ value until it is equal to final ϵ value
    Every C steps update $Q(\mathbf{s}_t, a_t; \boldsymbol{\theta}_t^-)$
    Check if *terminal*
    Update the current state $\mathbf{s}_t = \mathbf{s}_{t+1}$
  end while
end for

Exploration Policies:

In the training process, the agent must first explore the environment in order to better understand which actions will maximise its future reward and then exploit those states that leads it to that objective. I choose two different strategies that will balance the two phases, *i.e. exploration* and *exploitation*.

The first is known as a ϵ-greedy strategy. In this strategy, the agent explores randomly with probability $\epsilon \in]0, 1]$, and it follows a greedy policy $a_t = \arg\max_a Q(\mathbf{s}_t, a_t; \boldsymbol{\theta}_t)$ (exploitation) with probability $(1-\epsilon)$ for training at time step t. At the beginning, we set $\epsilon = 1$ (*i.e.*, pure exploration). We then gradually decrease ϵ as the training progresses, thus increasing the exploitation. This allows the system to explore the environment randomly, first. Then, with a probability of $1-\epsilon$ (which decreases every step), it exploit the areas of the environment that converge to the optimal policy, choosing the greedy

action.

The second is referred as Boltzmann or Softmax exploration policy (Sutton (1990)). Instead of always computing the optimal action, this strategy chooses an action based on a weighted probability through its Q-values, that is:

$$p(Q(s_t, a_t^i; \boldsymbol{\theta}_t), \epsilon) = \frac{e^{\frac{Q(s_t, a_t^i; \boldsymbol{\theta}_t)}{\epsilon}}}{\sum_{i=0}^{N-1} e^{\frac{Q(s_t, a_t^i; \boldsymbol{\theta}_t)}{\epsilon}}}, \tag{3.6}$$

where $Q(s_t, a_t^i; \boldsymbol{\theta}_t)$ is the Q-value for the set of actions $\{a_t^1, ..., a_t^N\}$, N is the number of actions, and, similar to the previously mentioned strategy, $\epsilon \in]0, 1]$ decreases at each step t. The parameter ϵ controls the exploration, *i.e.* by decreasing ϵ, the optimal value emerges as the most probable action during training. In the beginning, every action has an uniform probability, since the network weights are initialized uniformly random. Then, by decreasing ϵ, with a distribution given by (3.6), the optimal action is the most likely chosen, but many sub-optimal are considered also. If the network outputs three Q-values with a Softmax probability $p(Q(s_t, a_t^i; \boldsymbol{\theta}_t), \epsilon) = \{0.47, 0.48, 0.05\},\ i \in \{0, 1, 2\}$, the first action is likely to be chosen, since its probability is almost matches the second action, even though it is not the optimal one (by a small margin). The third action is immediately discarded. Therefore, the agent is allowed to explore actions that may lead to better ones. The goal is to discard obvious undesirably actions, instead of choosing a random or a greedy action (like ϵ-greedy).

3.1.2 Inference

The trained DQN model is parameterized by $\boldsymbol{\theta}^\star$ learned in (3.5) that outputs the action-value function for the state observation s_t. Formally, the action to follow from the current observation is defined by:

$$a_t^\star = \arg\max_{a_t} Q(s_t, a_t; \boldsymbol{\theta}^\star). \tag{3.7}$$

Finally, given that the number and location of pedestrians are unknown in a test image, this inference is initialized with different bounding boxes at several locations, and it runs until it either finds the pedestrian (with the selection of the trigger action), or runs for a maximum number of iterations.

3.2 Classification

In this section, I describe how object classification network is trained. The main objective in a classification task is to assign a certain object to a class, within a proposed region. These regions are: rectangular bounding box (He et al. (2014), Girshick et al. (2014), Girshick (2015), Ren, He, Girshick & Sun (2015), Liu et al. (2016), Redmon & Farhadi (2018)), segmented region (He et al. (2017)) or, as I propose, distorted bounding box.

The purpose is to classify the proposed region as having a pedestrian in it or not, thus I consider only two classes $y_c = \{0, 1\}$ (No Pedestrian and Pedestrian, respectively). Assuming one pedestrian per image, I consider that for each image $\mathcal{I}(.)$ a label is assigned. Then, our primary goal is to create a network, described by the parameters $\boldsymbol{\theta}_t$, capable of generating a prediction of the desired class for a test image that has never been seen before. It was use CNNs to model this behavior, by minimizing the classification error of a batch of labeled images according to cross-entropy's mean:

$$L_{cls}(\boldsymbol{\theta}_t) = \mathbb{E}_{\sim U(\mathbf{D}_t)} \left[-y_c \log(p(\widehat{y}_1; \boldsymbol{\theta}_t)) - (1 - y_c) \log(p(\widehat{y}_0; \boldsymbol{\theta}_t)) \right], \tag{3.8}$$

where: y_c is the labeled class; $p_i(\widehat{y}_i; \boldsymbol{\theta}_t)$ is the computed probability for each class; $\boldsymbol{\theta}_t$ are the network's weights & $\widehat{y}_i,\ i \in \{0, 1\}$ are its estimates (for each class at iteration t); and $U(\mathbf{D}_t)$ is the uniformly retrieved batch of labeled images from the memory used in the deep RL system[1]. This probability is computed through the Softmax function of the network's output:

$$p(\widehat{y}_i; \boldsymbol{\theta}_t) = \frac{e^{\widehat{y}_i}}{\sum_{i=0}^{1} e^{\widehat{y}_i}}, \quad i \in \{0, 1\}. \tag{3.9}$$

I use a cross-entropy loss for the binary classification, instead of the classification mean square error, since it is less computational expensive when computing the networks gradients, which, consequently, converges faster to the optimal value. The mean absolute error is used to cross-validate the results during training and, afterwards, to test them, in order to get a perception how the network extrapolates from untrained images. For inference, it is simply chosen i that has the highest $p_i(\widehat{y}_i; \boldsymbol{\theta}_t)$:

$$y_c^\star = \arg\max_i p(\widehat{y}_i; \boldsymbol{\theta}^\star), \tag{3.10}$$

[1] In Sec. 4.3, it is established how the labels are created and used for the classification task.

where $\boldsymbol{\theta}^{\star}$ are the optimal parameters of the model learned in (3.8). While training, to observe the learning process every V steps, I cross-validate my results from a batch retrieved from the memory $\mathbf{D}_t$, and using (3.10), check the accuracy.

Since classification is an auxiliary task to the main goal, in Sec. 4.3 I am going to specify how it is integrated with my deep RL framework.

Chapter 4

Robust Deep RL Pedestrian Detection for Omnidirectional Cameras

This chapter describes the environment and how the deep RL agent interacts with it. First, in Sec. 4.1, the bounding box shape and position that characterizes the agent's states in the world and its respective projection are detailed. Second, in Sec. 4.2, it is described how the agent's actions progressively modify the bounding box throughout the detection process, as well as how the rewards are computed. Third, I complement the deep RL agent's training with a classification network and update the final training algorithm, in Sec. 4.3.

4.1 Bounding Box Projection

Like most of the state-of-the-art detection algorithms, my goal is to define the pedestrian via bounding box. However, in contrast to other related approaches (*e.g.*, Caicedo & Lazebnik (2015), Ghesu et al. (2016)), the novelty herein proposed is the ability to provide the coordinates of the bounding box in the 3D world, instead of the image domain. Furthermore, it takes into account the underlying distortion of the omnidirectional cameras. The proposed framework allows a better fitting of the person and, at the same time, gives the pedestrian's world position.

The bounding box is characterized by a position $\mathbf{p}_t = [\rho, \beta, z]^T$, in cylindrical coordinates[1], which ideally is located at the pedestrian's feet, and a dimension $\mathbf{dim}_t = [w, h]^T$

[1]Due to the nature of the omnidirectional images, this is the best way of representing the world's bounding box position.

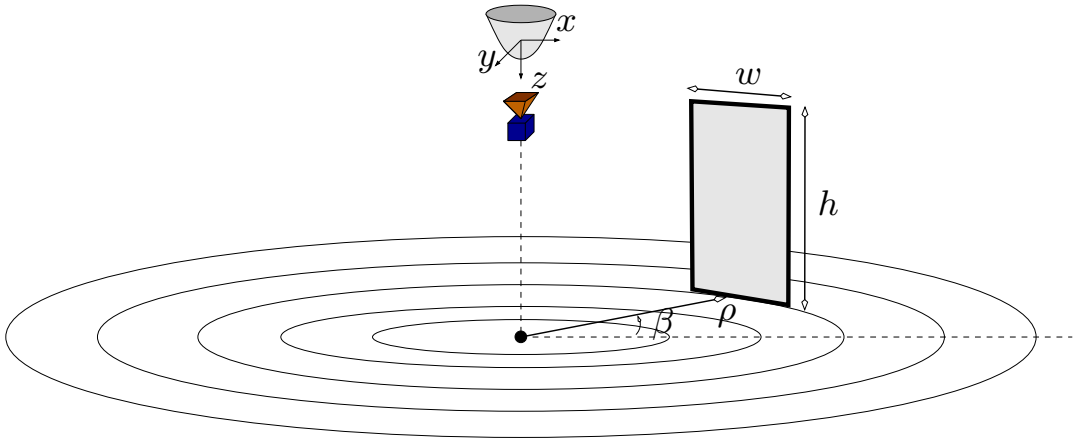


Figure 4.1: Illustration of the environment defined in this work. The camera position w.r.t. to the world coordinate system is assumed to be known, which means that the position of the bounding both in the world can be defined in cylindrical coordinates (assuming that the pedestrian is in vertical position in the ground plan).

for the width and height of the bounding box that fits the person, as shown in Fig 4.1. For simplicity, the bounding box is assumed to face the imaging device, which means that it will be tangent to the curve defined by ρ and β. Taking this into account, the function $\mathcal{M}$, can be formally defined by:

$$\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\} = \mathcal{M}\left(\mathbf{p}_t, \mathbf{dim}_t\right), \tag{4.1}$$

that takes $\mathbf{p}_t$ and $\mathbf{dim}_t$, and computes four points that set the rectangular limits of the box in cartesian coordinates [2]. The transformation to the regular cartesian coordinates is a requirement for the line projection as described in Sec. 2.2. The lines represented on the normalized plane are given by:

$$\mathcal{L}_{i,j} = \mathcal{G}(\mathbf{x}_i, \mathbf{x}_j), \text{with } (i,j) \in \{(1,2),\ (1,3),\ (2,4),\ (3,4)\}. \tag{4.2}$$

Finally, $\mathcal{L}_{i,j}$ are projected into the image, by using (2.4).

Considering the goal of developing a deep RL agent, the states are defined by the resulting image cropped to the limits of the projected curves, and then resized & normalized to the network's input dimensions. Examples of these image processing steps are shown in Figs. 1.1 and 4.2. States are defined as:

$$\mathbf{s}_t = \mathcal{I}\left(\mathcal{L}_{i,j}\right) \in \mathbf{S}, \text{with } (i,j) \in \{(1,2),\ (1,3),\ (2,4),\ (3,4)\}. \tag{4.3}$$

[2] See the next section for the formulation of $\mathcal{M}$.

(a) (b) (c)

Figure 4.2: This figure shows three different examples of states, from three different environments. Figs. (a) and (b) are states that contain pedestrians. The first one is further away from the pedestrian and the second is close to the camera system. None of them are a good estimates for the final bounding box. Fig (c) does not contain a pedestrian.

where $\mathcal{I}(.)$ is the resulting image.

4.1.1 Bounding Boxes' Corner Points Computation

In the previous section, I define $\mathcal{M}$ as the function that computes the four points describing the limits of the 3D bounding box from its position $\mathbf{p}_t$ and dimension $\mathbf{dim}_t$. Recall that $\mathbf{p}_t = [\rho, \beta, z]^T$, then the first step is to map into Cartesian coordinates:

$$\mathbf{p}_t \mapsto \widehat{\mathbf{x}} = [\widehat{x}_1, \widehat{x}_2, \widehat{x}_3]^T, \tag{4.4}$$

where $\widehat{\mathbf{x}}_1^i$ is the center point of the bounding box on the floor. By observing Fig. 4.3, we compute δ_1 from its known width (denoted by w):

$$\delta_1 = cos\,(\beta)\left(\frac{w}{2}\right). \tag{4.5}$$

Then, one must add and subtract to $\widehat{x}_1$, in order to obtain the x-coordinates to the opposite edges of the bounding box, respectively:

$$\widehat{x}_1^+ = \widehat{x}_1 + \delta_1 \quad \text{and} \quad \widehat{x}_1^- = \widehat{x}_1 - \delta_1. \tag{4.6}$$

The y-coordinates of these edges are computed through the affine equation:

$$\widehat{x}_2^i = \frac{-1}{tan\,(\beta)}\widehat{x}_1^i + \left(\widehat{x}_2 + \frac{1}{tan\,(\beta)}\right), \quad i \in \{+, -\}, \tag{4.7}$$

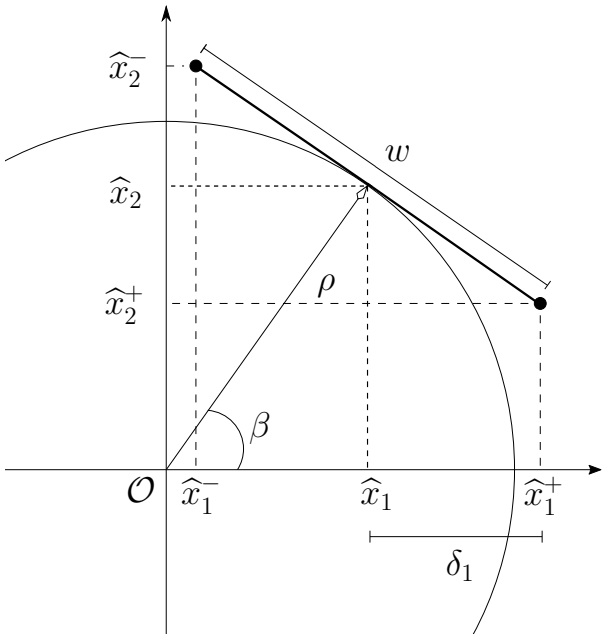


Figure 4.3: In this figure, we observe how the bounding box is presented in the world (top view, *i.e.* in the xy–plane). The bounding box is tangent to the curve defined by ρ and β, and its length is equal to the width w.

where x_i is the x-coordinate of that edge of the bounding box.

To sum-up, the four points defining the edges of the bounding box are defined as follows:

$$\begin{aligned}
&\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\} = \mathcal{M}(\mathbf{p}_t, \mathbf{dim}_t) \Leftrightarrow \\
&\begin{cases}
\mathbf{x}_1 = [\widehat{x}_1^+, \frac{-1}{tan(\beta)}\widehat{x}_1^+ + \left(\widehat{x}_2 + \frac{1}{tan(\beta)}\right), \widehat{x}_3]^T \\
\mathbf{x}_2 = [\widehat{x}_1^-, \frac{-1}{tan(\beta)}\widehat{x}_1^- + \left(\widehat{x}_2 + \frac{1}{tan(\beta)}\right), \widehat{x}_3]^T \\
\mathbf{x}_3 = [\widehat{x}_1^+, \frac{-1}{tan(\beta)}\widehat{x}_1^+ + \left(\widehat{x}_2 + \frac{1}{tan(\beta)}\right), \widehat{x}_3 - h]^T \\
\mathbf{x}_4 = [\widehat{x}_1^-, \frac{-1}{tan(\beta)}\widehat{x}_1^- + \left(\widehat{x}_2 + \frac{1}{tan(\beta)}\right), \widehat{x}_3 - h]^T.
\end{cases}
\end{aligned} \tag{4.8}$$

4.2 Actions and Rewards

For the detection task, the agent must perform actions, according to the policy $\pi(a|s)$. Since the agent's state depends on the duplet ($\mathbf{p}_t$, $\mathbf{dim}_t$), the actions have to be applied to both parameters. Every possible state has a set of nine actions, which are function

Table 4.1: Set of actions considered in the proposed deep RL. The superscripts $+, -$ stand for positive or negative updates for depth (ρ), rotation (β) z-scale (h) and (x, y)-scale (w). In (a) and (b) the dimension $\mathbf{dim}_{t+1} = \mathbf{dim}_t$ and the position $\mathbf{p}_{t+1} = \mathbf{p}_t$ remains unchanged, respectively.

(a) Update Position

Action	**Position** ($\mathbf{p}_{t+1}$)
$a_t = \rho^+$, $a_t = \rho^-$	$[\rho + a_t, \beta, z]^T$
$a_t = \beta^+$, $a_t = \beta^-$	$[\rho, \beta + a_t, z]^T$

(b) Update Dimension

Action	**Dimension** ($\mathbf{dim}_{t+1}$)
$a_t = w^+$, $a_t = w^-$	$[w + a_t, h]^T$
$a_t = h^+$, $a_t = h^-$	$[w, h + a_t]^T$

of the β, ρ, h, and w as shown in Fig. 4.1, and the trigger σ, such that:

$$a_t \in \mathcal{A} = \{\rho^+, \rho^-, \beta^+, \beta^-, w^+, w^-, h^+, h^-, \sigma\}, \tag{4.9}$$

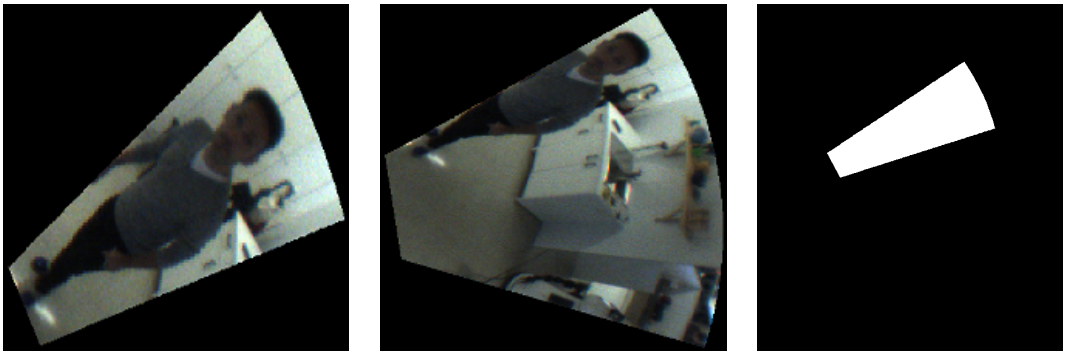
where the superscripts $+, -$ stand for the positive and negative updates for each of the parameters. The effects of each action on the bounding box position and dimension is shown in Tab. 4.1. Notice that σ does not perform any modification on the bounding box, but signals if the pedestrian has been found through the best fitting. Then, the next state $\mathbf{s}_{t+1}$ is defined, using (4.3). Finally, the reward function r_t depends on which action a_t is performed. When the agent chooses the trigger σ, the reward function is formally given by:

$$r_t = \mathcal{R}(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}) = \begin{cases} +\alpha & \text{if } \ IoU(\mathbf{A}_{t+1}, \mathbf{A}_g) \geq \tau \\ -\alpha & \text{otherwise} \end{cases}, \tag{4.10}$$

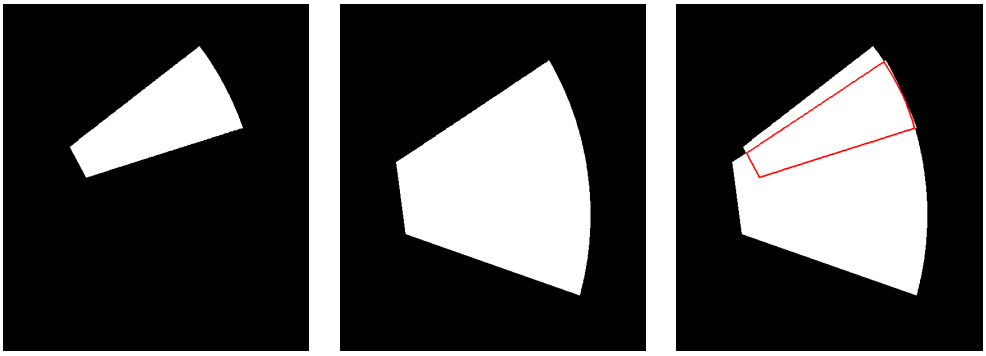
where $IoU(.) \geq 0$ (see App. A for a detailed explanation of how the IoU is computed) is the Intersection over Union, r_t is the immediate reward, and the threshold τ that flags if the box on the next state is suitable to define the pedestrian. Notice that, the notation $\mathbf{A}$ stands for pixels that lie within the lines of the distorted bounding box (see Fig. 4.4). Thus, the IoU is performed between the estimated detection $\mathbf{A}_{t+1}$ and the available ground truth $\mathbf{A}_g$[3].

For the other actions that modify the $\mathbf{pos}_t$ and $\mathbf{dim}_t$, the reward function must be proportional to the improvement. Therefore, the reward is positive if the position or the

[3]In the training stage, every image in the dataset has a label that describes the ground truth, $\mathbf{s}_g = \mathcal{I}(\mathbf{p}_g, \mathbf{dim}_g)$ of the ideal bounding box of the person.



(a) Target State $\mathbf{s}_g$ (b) Next State $\mathbf{s}_{t+1}$ (c) Bounding Boxes' Intersection



(d) Target $\mathbf{A}_g$ (e) Next $\mathbf{A}_{t+1}$ (f) Bounding Boxes' Union

Figure 4.4: In this figure, we observe the representation of $\mathbf{A}$ for a step. The next state $\mathbf{s}_{t+1}$ (with state representation (b)) produces, in a auxiliary image, the group of pixels $\mathbf{A}_{t+1}$. This image (e) corresponds to the exact projection of the bounding box to the real image before being cropped and resized. From the same method, we extract $\mathbf{A}_g$, (d). Then, by using image processing, we create (c) and (f) which represent the intersection and union of both $\mathbf{A}_{t+1}$ and $\mathbf{A}_g$, respectively. Also in (f), the intersection of both bounding boxes is highlighted in red.

dimension is modified, such that the projection is closer to the ground truth. The reward is negative otherwise. Then, the improvement is measured by the sign of the difference between the intersection of the next state's bounding box $\mathbf{A}_{t+1}$ with the ground truth $\mathbf{A}_g$, and the current bounding box with the ground truth, formally:

$$r_t = \mathcal{R}(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}) = sign(IoU(\mathbf{A}_{t+1}, \mathbf{A}_g) - IoU(\mathbf{A}_t, \mathbf{A}_g)). \tag{4.11}$$

The primary problem with this reward function is that the algorithm has to ensure that $IoU(\mathbf{A}_t, \mathbf{A}_g) \neq 0$ or $IoU(\mathbf{A}_{t+1}, \mathbf{A}_g) \neq 0$. If this is not guaranteed, the agent cannot

measure any improvement, and therefore make a valid action. This is one of the problems with this approach. The two bounding boxes need an overlap between each other for the method to work. Hence, if this overlap does not exist, the episode ends and another image is analyzed.

The training process follows (Sec. 3.1) that models the DQN by maximizing the cumulative rewards. As above mentioned, this relies on the experience-replay memory and the target network (Mnih et al. (2015)). Experience replay uses the dataset $\mathbf{D}_t = \{e_1, ..., e_t\}$ and the target network uses the parameters $\boldsymbol{\theta}_t^-$ that computes the target for the DQN periodically updates. The insight of using the $sign(.)$ (*i.e.* quantization of the reward), is used to avoid to confuse the agent about which actions are good or bad, and it helps restrict the derivatives when updating, improving the stability of the algorithm (Mnih et al. (2015)). The DQN is trained from scratch, *i.e.* the network was trained to extract features from omnidirectional camera systems.

4.3 Multi-task Training

The problem with my method is that it requires an initial estimate of the pedestrian's position. Caicedo & Lazebnik (2015) had the same problem. Since it is needed an overlap between the ground truth and the initial position, they chose different zones of the image to start the detection process, in order to obtain all objects in the image. Since the environment is in the world domain, we would have to search areas of the image until the trigger is applied (which is what we did in Sec. 5.2). However, to overcome this issue, the proposed method uses a classification network (Sec. 3.2) to indicate us whether there is a pedestrian in the proposed initial area or not, instead of searching with the developed DQN.

To accomplish this, I propose a multi-task network that trains both networks at the same time, and, eventually, it will make the test phase an end-to-end solution. This network will share the first convolutional layers, and then split up in to two branches, one for each task (RL and classification). Note that both tasks are not complementary, like other classification and region proposal methods (Ren, Li & Fraser (2015), He et al. (2017)), and, consequently, a shared multi-task loss cannot be computed. Instead, it trains both branches separately, *i.e.* at each step the algorithm chooses if it trains the DQN branch or the classification one. Furthermore, since our focus is on the detec-

tion task, the classification branch is only trained every K steps. This will help retrieve the pedestrian's features and accelerate the DQN training. The multi-task network also prevents training two separate networks, reducing computational time and resources.

The training process of the new network follows the same process as the DQN. A sequence of experiences $\mathbf{e}_t = (s_t, a_t, r_t, s_{t+1})$ are stored in the memory $\mathbf{D}_t$, and then a random minibatch of $\mathbf{e}_j$ is selected to train. When the new algorithm selects the DQN branch, it will train as I previously described (Alg. 1). However, the method changes when it trains the classification branch.

Let us consider a state $\mathbf{s}_j$ from the minibatch. From its $IoU(\mathbf{A}_j, \mathbf{A}_g)$, the label for the classification task can be obtained. If $IoU(\mathbf{A}_j, \mathbf{A}_g) \neq 0$, then I consider that there is a pedestrian in the image ($y_c = 1$), and if $IoU(\mathbf{A}_j, \mathbf{A}_g) = 0$, then there is no pedestrian ($y_c = 0$). The problem with our deep RL formulation is that in $\mathbf{D}_t$, most of the states (or even all of them) have an $IoU(\mathbf{A}_j, \mathbf{A}_g) \neq 0$. Therefore, states that do not have pedestrians in them must be generated.

From the minibatch, a random number of states are selected. In these states, their position ($\mathbf{p}_j$) is modified by adding a $\Delta\beta$, ensuring that $IoU(\mathbf{A}_j, \mathbf{A}_g) = 0$. Next, the modified states are stacked and shuffled in the minibatch. Finally, the model is trained following (3.5). The method is formalized in Alg 2.

Algorithm 2 Multi-task Deep RL and Classification training with a Boltzmann exploration policy and Double Q-Learning (Hasselt et al. (2016)).

Initialize replay memory D to capacity N;
Initialize action-value function $Q(\mathbf{s}_t, a_t; \boldsymbol{\theta}_0)$ and Softmax function $p(\widehat{y}, \boldsymbol{\theta}_0)$ with random weights $\boldsymbol{\theta}_0$;
Initialize target action-value function $Q(\mathbf{s}_t, a_t; \boldsymbol{\theta}_0^-)$ with weights $\boldsymbol{\theta}_0^- = \boldsymbol{\theta}_0$;
for $t = 1, M$ **do**
  Select random image from the dataset;
  Initialize $\mathbf{p}_0$ and $\mathbf{dim}_0$;
  while *terminal* $==$ *False* **do**
    Select action a_t based on the probability distribution $p(Q(s_t, a_t^i; \boldsymbol{\theta}_t), \epsilon)$ (3.6);
    Execute action a_t in environment and observe the reward r_t and state s_{t+1};
    Store in $\mathbf{D}_t$ the experience $\mathbf{e}_t = (\mathbf{s}_t, a_t, r_t, \mathbf{s}_{t+1})$;
    Sample random minibatch of transitions $\mathbf{e_j}$ from $\mathbf{D}_t$;
    if $t\%K == 0$ **then**
      Select a random number of states $\mathbf{s}_j^\star$ from the minibatch;
      Modify the states $\mathbf{p}_j^\star = [\rho, \beta + \Delta\beta, z]$, add them to the minibatch and shuffle it;
      Perform gradient descent on $L_{cls}(\boldsymbol{\theta}_t)$ (3.8) with the respect of the parameters $\boldsymbol{\theta}_t$;
    else
      set $q_j^{DDQN} = \begin{cases} r_j & \textit{if episode ends at step } j+1 \\ r_j + \gamma Q(s_{j+1}, \arg\max_{a_{j+1}} Q(s_{j+1}, a_{j+1}; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t^-) & \textit{otherwise} \end{cases}$
      Perform gradient descent on $L_{drl}(\boldsymbol{\theta}_t)$ (3.5) with the respect of the parameters $\boldsymbol{\theta}_t$;
    end if
    Decrease current ϵ value until it is equal to final ϵ value
    Every C steps update $Q(\mathbf{s}_t, a_t; \boldsymbol{\theta}_t^-)$
    Check if *terminal*
    Update the current state $\mathbf{s}_t = \mathbf{s}_{t+1}$
  end while
end for

Chapter 5

Experimental Results

In this section, I describe how the omnidirectional images' dataset is created (Sec. 5.1), and how the network architecture is designed. Then, I compare the newly developed agent with the perspective state-of-the-art method (Sec. 5.2). I perform a comparison between: 1) my bounding boxes methodology, set in the environment discussed in Chp. 4 (denoted "Ours"); and 2) with the method proposed in Caicedo & Lazebnik (2015) using rectangular bounding boxes (i) in an omnidirectional setting (denoted "SotA") and (ii) performing an unwrapping (undistorted) to the original image (denoted "SotAU"). Note that the environment where the actions are performed for the "SotA" and "SotAU" methods are in the image domain. Second, I create a new pipeline for testing using the developed multi-task network (denoted "OursV2"), and compare it to the previously developed work (Sec. 5.3). My methodology is developed using `Python` with `TensorFlow 1.4` (Abadi et al. (2015)) and `OpenCV 3.1` (Itseez (2015)).

5.1 Dataset Creation

To the best of my knowledge, there is no publicly available omnidirectional images' dataset labeled for pedestrian detection. Therefore, I have acquired a new dataset using three different environments in ISR laboratory with several subjects (see Fig. 5.1). With a total of 921 images, 70% of them are used training and the other 30% are used for testing. These images were obtained with PointGrey (2017)'s Flea3 attached to a hyperboloid mirror, modeled by a central catadioptric camera system. The imaging device was calibrated using Mei & Rives (2007) framework.

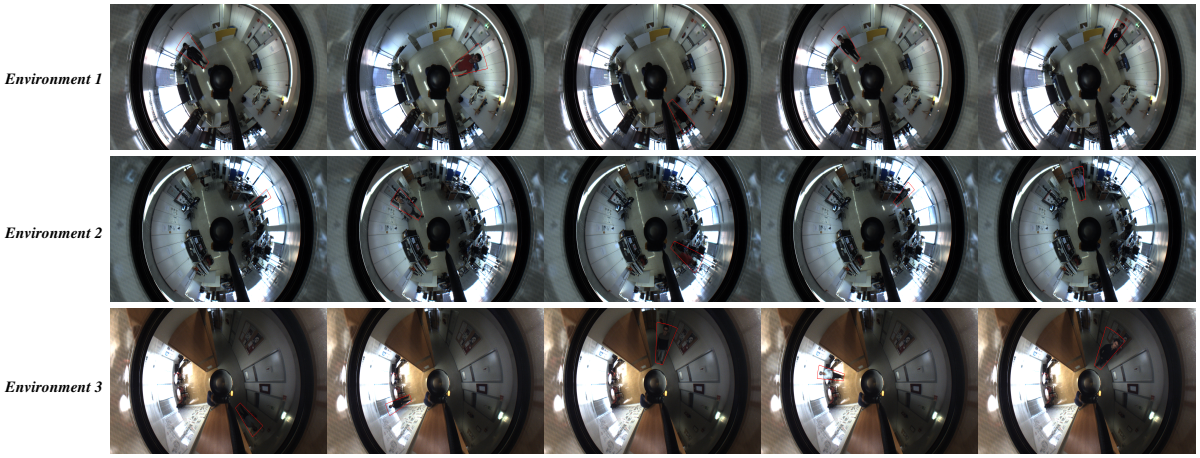


Figure 5.1: In this figure, I show dataset examples, as well as, the ground truth for the pedestrians in those images. The environments are set in the robotics lab's testbed, in the office space of the lab, and in the corridor for the collaborative room, respectively.

In the acquired dataset, the images are labelled considering the location of the pedestrians for the three above referred methods. For "Ours", one must find the pedestrian's position in the world's coordinate system from an image's pixel reprojection on the omnidirectional image.

First, the point $\mathbf{i}$ (recall (2.4)), that represents the pixel's coordinates of the mid point between the pedestrian's feet, must be projected to the normalize plane through:

$$\begin{bmatrix} \widetilde{\mathbf{i}} \\ 1 \end{bmatrix} = \mathbf{K}^{-1} \begin{bmatrix} \mathbf{i} \\ 1 \end{bmatrix} \text{ where } \mathbf{K} = \begin{bmatrix} f_1\eta & f_1\eta\alpha & u_0 \\ 0 & f_2\eta & v_0 \\ 0 & 0 & 1, \end{bmatrix}, \tag{5.1}$$

where $\widetilde{\mathbf{i}} = \begin{bmatrix} \widetilde{i}_1, \widetilde{i}_2 \end{bmatrix}^T$ is its projection to the normalize plane.

Next, from Barreto & Araujo (2001), it is known that the point computed before projected back on to the unit sphere is given by:

$$\mathbf{n} = \begin{bmatrix} \lambda_c \widetilde{i}_1 \\ \lambda_c \widetilde{i}_2 \\ \lambda_c - \xi \end{bmatrix}, \tag{5.2}$$

where:

$$\lambda_c = \frac{\xi + \sqrt{1 + (1 - \xi)\left(\widetilde{i}_1 + \widetilde{i}_y\right)}}{\widetilde{i}_1^2 + \widetilde{i}_2^2 + 1}. \tag{5.3}$$

This transformation takes into account the change in reference frame to the origin $\mathcal{O}$ (from $\mathbf{c}_p$), where the projection to the normalized plane was made. For the sake of calculations' simplicity, I define $\mathbf{n} = [n_1, n_2, n_3]^T$ (point on the unit sphere), being the line that contains $\mathbf{n}$:

$$\widehat{\mathbf{x}} = \mathbf{n} + \lambda \mathbf{d} = \begin{cases} \widehat{x}_1 = n_1 + \lambda d_1 \\ \widehat{x}_2 = n_2 + \lambda d_2 \\ \widehat{x}_3 = n_3 + \lambda d_3 \end{cases}, \tag{5.4}$$

where $\widehat{\mathbf{x}} = [\widehat{x}_1, \widehat{x}_2, \widehat{x}_3]^T$ is a point that belongs to the line for a given depth λ with a specific direction $\mathbf{d} = [d_1, d_2, d_3]^T$. Since I am projecting $\mathbf{n}$ to the world coordinates through $\mathcal{O}$, its direction is known and it is equal to $\mathbf{n}$. Recall that the point $\mathbf{n}$ is on the unit sphere, therefore the direction is already normalized (*i.e.* $||\mathbf{d}|| = ||\mathbf{n}|| = 1$).

From my problem, I consider that the z-plane is known (*i.e.* $\widehat{x}_3$). Then, from (5.4), one can compute λ:

$$\lambda = \frac{\widehat{x}_3 - n_3}{n_3}, \tag{5.5}$$

Replacing the above result in $\widehat{x}_1$ and $\widehat{x}_2$ (see (5.4)), it is obtained $\widehat{\mathbf{x}}$ in the world's coordinate system. Finally, the point can be mapped to cylindrical coordinates $\widehat{\mathbf{x}} \mapsto \mathbf{p}_g$, where $\mathbf{p}_g$ is given as:

$$\mathbf{p}_g = \begin{bmatrix} \sqrt{\widehat{x}_1^2 + \widehat{x}_2^2} \\ \arctan(\widehat{x}_2 / \widehat{x}_1) \\ \widehat{x}_3 \end{bmatrix}. \tag{5.6}$$

The ground truth dimension $\mathbf{dim}_g$ is chosen depending on the person's height and width. The initial position $\mathbf{p}_0$ and dimension $\mathbf{dim}_0$, for each label, was also set offline, ensuring that $IoU(\mathbf{A}_0, \mathbf{A}_g) \neq 0$.

For "SotA" and "SotAU", the bounding box in the image plane is defined. ÂłThen two pixels from opposite sides ($\widetilde{\mathbf{i}}^1$, $\widetilde{\mathbf{i}}^2$) are chosen, to create a rectangular box fitting the pedestrian. In order to keep the same environment definition from the method, the bounding box ground truth for these methods is also defined by a dimension $\mathbf{dim}_g$:

$$\mathbf{dim}_g = \begin{bmatrix} \widetilde{i}_1^2 - \widetilde{i}_1^1 \\ \widetilde{i}_2^2 - \widetilde{i}_2^1 \end{bmatrix} = \begin{bmatrix} w \\ h \end{bmatrix}, \tag{5.7}$$

where w and h are the width and height of the bounding box, respectively, and the position $\mathbf{p}_g$ is given by:

$$\mathbf{p}_g = \begin{bmatrix} \widetilde{i}_1^1 + \frac{w}{2} \\ \widetilde{i}_2^1 + \frac{h}{2} \end{bmatrix}, \tag{5.8}$$

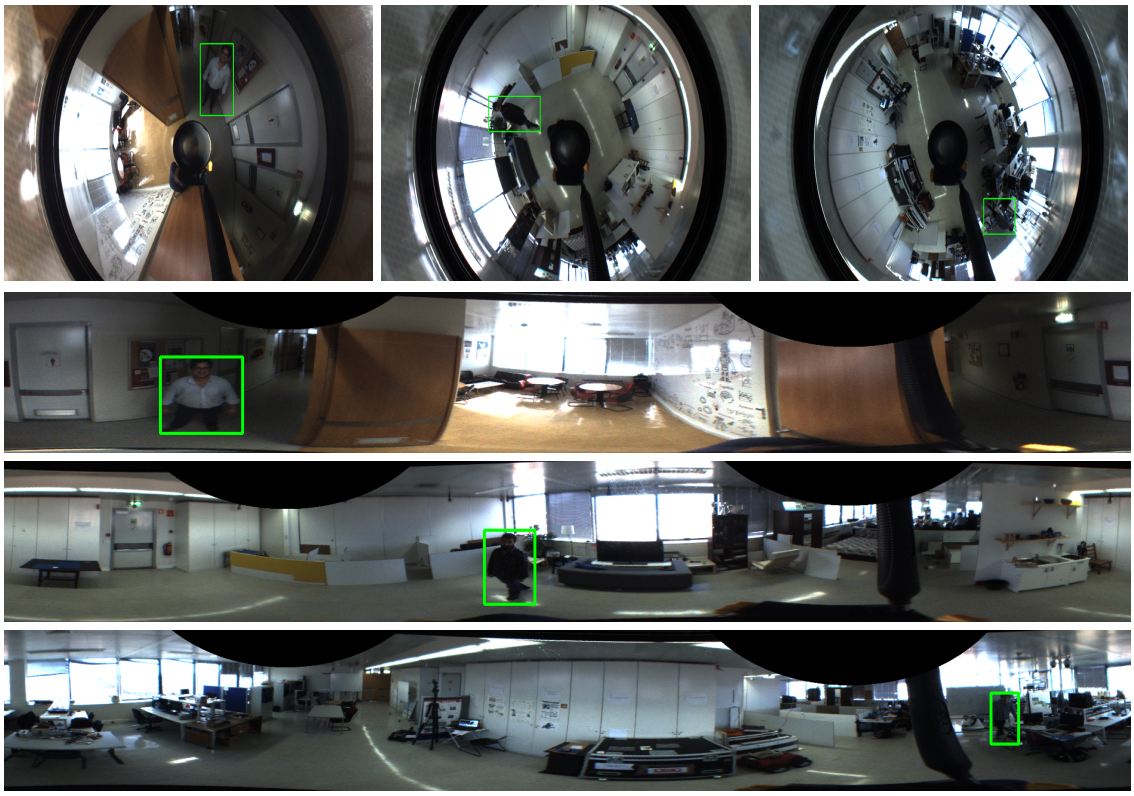


Figure 5.2: This figure shows examples of the ground truth representation for the "SotA" and "SotAU" methods. For the three on the top ("SotA"), one observes that the bounding box does not fit well the pedestrian, due to its geometry. In the bottom ones ("SotAU"), the bounding box fits the pedestrian, but the image loses some of its properties.

that is the center of the bounding box (see Fig. 5.2). This position describes the center of the person, instead of the pedestrian's feet, in comparison to "Ours". The initial position and dimension were set by the same criteria mentioned before.

5.2 Preliminary Results

Once the labelling process is concluded, the model is built and the environment parameters are fine-tuned, in order to train the deep network, according to the environment discussed in Sec. 4.1. This first network is going to serve as proof of concept to my method, *i.e.* only the deep RL is trained. The multi-task network is not used for the validity of my main contribution (use deep RL to detect pedestrians).

The network "Ours" has five convolutional layers and two fully-connected layers,

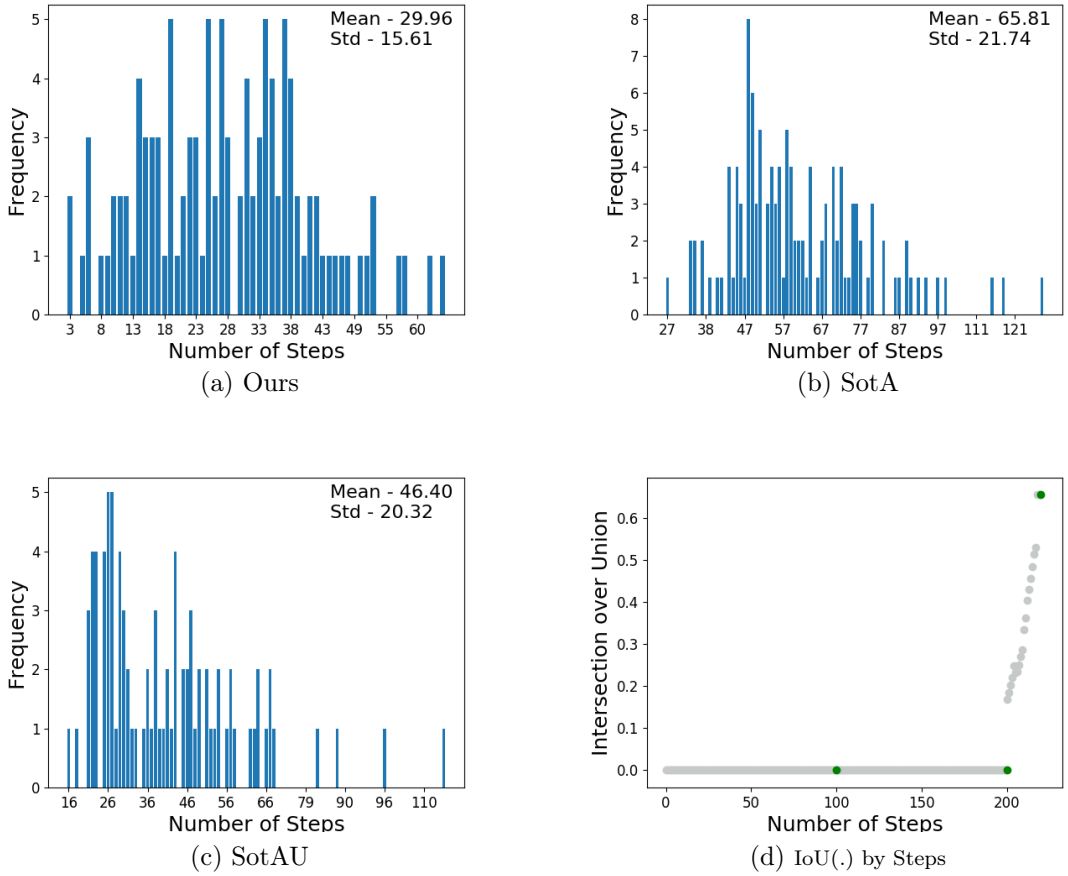
Table 5.1: This table shows the average steps in the test sequence to set the trigger correctly for the different methods, and the average IoU for both methods that evaluate the distorted image on the distorted ground truth. The average IoU is only computed for those methods, because the projection of the bounding box to the omnidirectional image after its undistortion in "SotAU" cannot be obtained.

	Average Steps	Average IoU
SotA	62.65	0.385
SotAU	28.46	–
Ours	**27.85**	**0.623**

where the input is a 224×224 image size, and the output contains the Q-values for the nine possible actions (see Sec. 4.2). The above architecture will be the same for both "SotA" and "SotAU" networks. Notice that, the same dimension of actions can be maintained as in Caicedo & Lazebnik (2015), thus the same architecture to train both methods can be kept. The three networks (*i.e.* "SotA", "SotAU" and "Ours") are trained from scratch and using Alg. 1 with a ϵ-greedy exploration. The training parameters are also the same for all the above networks. More specifically, I considered $\tau = 0.6$, $\alpha = 10$, $C = 15000$ steps and a fixed learning rate of 0.0001.

For the test initialization purposes, I choose a fixed ρ closed to the camera center and a fixed dimension $\mathbf{dim}_0$ with a relatively large size. Then, I select six (random) values for the β parameter in order to cover the whole image domain, aiming to detect some pedestrian location. In Fig. 5.3(d), it is illustrated three different values of β. For the third value of this parameter, we see that after a couple of iterations it was obtained $IoU(.) > 0.6$ and the trigger is activated, meaning that the pedestrian has been detected. For the other values, no detection has been triggered (a maximum number of 100 steps has been used).

For evaluation, I used the test sequence to compute the average steps required to detect the pedestrian, as well as the average IoU over the final detections. Notice that the steps correspond to actions taken and are counted from the first initial position until the detection of the pedestrian has been triggered. For the IoU computation, I only take in consideration the two methods that deal with the distorted image (*i.e.* "SotA" and "Ours"). From Tab. 5.1, it can be seen that my method takes much less steps to achieve the correct detection than the "SotA", even though my environment is much more complex, as the coordinate system is set in the world and not in the image.



(a) Ours (b) SotA

(c) SotAU (d) IoU(.) by Steps

Figure 5.3: The bar graphs (a), (b) and (c) show the number of required steps to detect pedestrian for "Ours", "SotA" and "SotAU" in the experiments, respectively. In each graph, it is computed the mean number of steps and its standard deviation (Std). Fig. (d) illustrates how the initialization procedure acts by choosing three different values for β parameter (a maximum of 100 iterations used for each value). Notice that for the third value of β a detection has been triggered.

Regarding the perspective method on the undistorted image ("SotAU"), I obtained a small number of average steps due to the simplicity of its environment, since the image is much smaller and the initialization of the initial bounding box already covers a big part of the environment, result of the shortcomings discussed in Chp. 1. Also note that, my method reaches an improvement of 23.8% regarding the "SotA" method in relation to the average IoU.

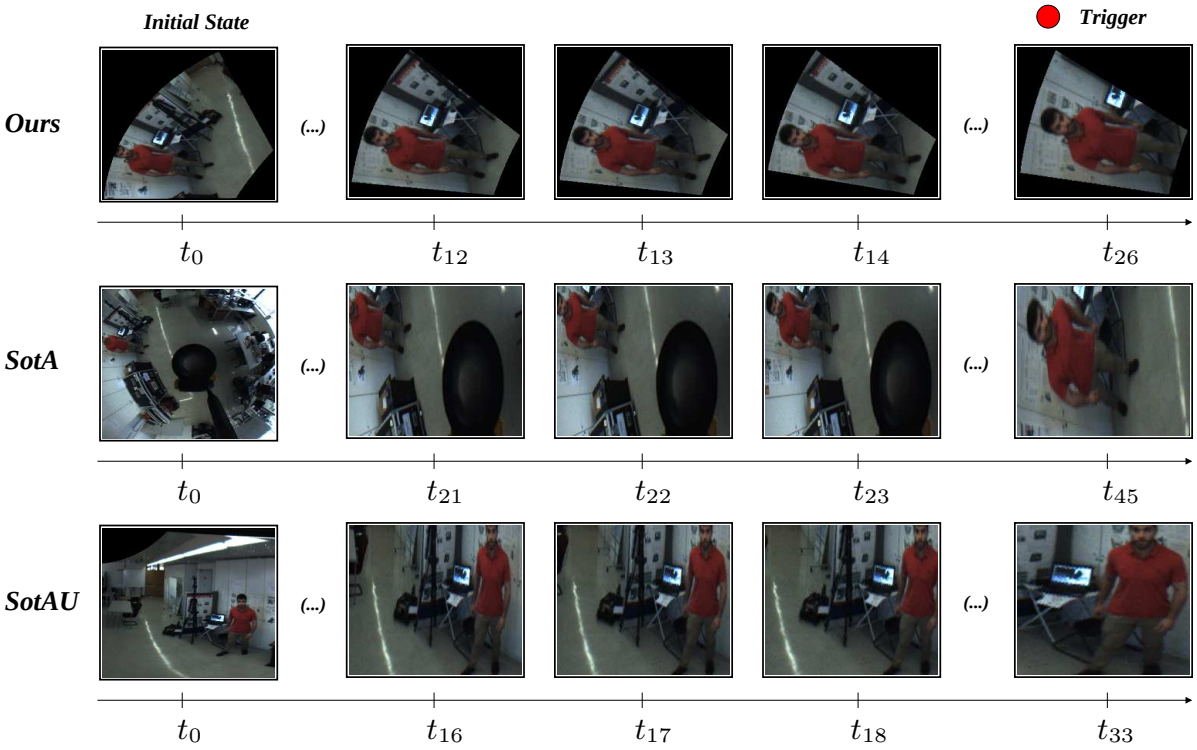


Figure 5.4: Illustration of the states evolution for the detection. "Ours" method (first row), "SotA" method (second row) both performed in omnidirectional images, and "SotAU" method (third row), performed in undistorted images. Notice that with my method, bounding box that takes into account the distortion can be obtained, in contrast to "SotA"s, when the resize for the states is made, the bounding box tries to compensate the distortion, as we can observe on the final state.

Fig. 5.3(a), (b) and (c) provide a more detailed information regarding Tab. 5.1. In this figure it is shown how many actions are required ("Number of steps") to detect a given number of pedestrians ("Frequency"). It can be concluded that the proposed approach, needs much less number of steps (a maximum of about 60 steps) to detect a larger number of pedestrians. Also note that, the "SotA" uses a larger number of steps (approx. 120 steps) but never reaches the same score of detections, when compared to "Ours". This figure also shows the "SotAU" results, where it can be seen that performing an undistortion of the images, it is able to improve over the state-of-the-art method "SotA"(Caicedo & Lazebnik (2015)).

Fig. 5.4 illustrates the actions performed until the trigger has been activated. Although only one example is shown, it is expected that this result can be extended to

Table 5.2: This table shows the position errors of ρ and β for both "Ours" and "OursV2". The difference of the position obtained when triggered and the ground truth is computed, and then the Root Mean Square Error (RMSE) and the error Standard Deviation (Std). We can observe a slight improvement using the second version of my algorithm in the position errors.

	RMSE ρ (m)	RMSE β (rad)	Error Std ρ (m)	Error Std β (rad)
Ours	0.6748	0.0875	**0.4097**	0.0837
OursV2	**0.5541**	**0.0316**	0.5506	**0.0318**

the entire test set (see results in Tab. 5.1 and Fig. 5.3(a), (b) and (c)). The proposed method "Ours" triggers the detection using a smaller number of actions and deals better with the distortion when compared to both "SotA" and "SotAU" methodologies.

5.3 Final Results

Now that it was proved that "Ours" performs better than "SotA" and "SotAU", the problem mentioned in Sec. 4.2 remain, *i.e.* the fast convergence of my method depends highly on the bounding box's initialization. These issued are tackled by creating a pipeline that uses both the deep RL and the classifications tasks, when testing to develop an end-to-end solution.

The network has three shared convolutional layers, two more convolutional layers and two fully-connected layers for each branch. The input size is the same as in the previous section, but the output changes. In addition to the nine Q-values, the Softmax probability is retrieved for the two classes, in the classification network. For comparison purposes, I analyze the new results ("OursV2") with the previous "Ours", "SotA" and "SotAU" approaches. The network was trained using Alg. 2 with a Boltzmann exploration. The training parameters are the same as before, with the addition of $K = 10$ (see K in Alg. 2).

The testing pipeline starts with the same approach as before. A ρ and a dimension $\mathbf{dim}_0$ are fixed with the same value. However the number of consecutive values of β that cover the whole omnidirectional environment increases to 20. Then, by using the classification network, the signaled position with pedestrians is retrieved and saved in a vector. Assuming that only one pedestrian is present in the image, the position in the middle of this vector is chosen. At last, the DQN starts the detection process using the

Table 5.3: The Tab.(a) shows the average steps and the IoU of both my approaches. We notice a slight improvement comparing "OursV2" to "Ours". "OursV2" converges faster to the optimal detection (lower average steps) and has a better fitting of the bounding box (larger average IoU). In Tab.(b), the percentage of correct detections and the training steps are shown between each method. With "OursV2", I obtain an higher percentage than any other method, while having the lowest amount of steps during training.

(a)

	Average Steps	Average IoU
Ours	28.46	0.623
OursV2	**21.39**	**0.718**

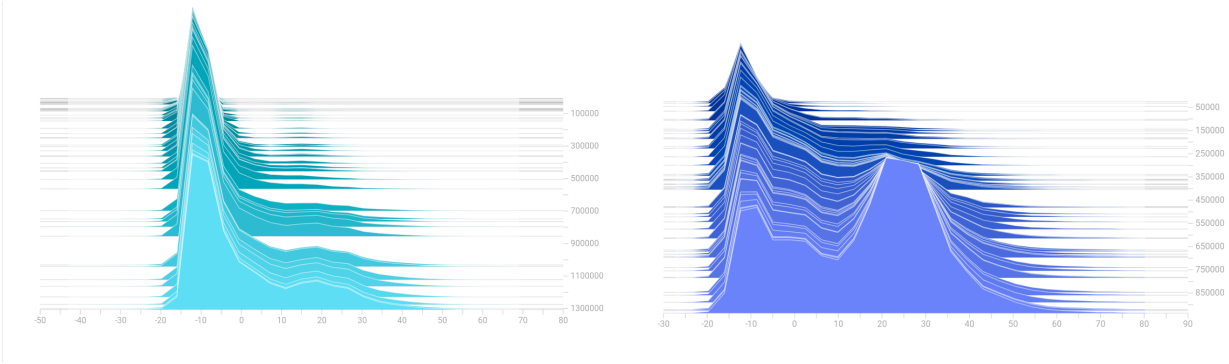
(b)

	Correct (%)	Training Steps
SotA	83.9	0.975×10^6
SotAU	77.2	1.385×10^6
Ours	71.3	1.306×10^6
OursV2	**86.5**	$\mathbf{0.936 \times 10^6}$

chosen $\mathbf{p}_0$. This approach can be expanded to multiple pedestrians by creating a vector each time a transition between a signaled pedestrian to a non pedestrian classification is made.

Tab. 5.2 shows the obtained errors for the duplet of model parameters (ρ, β) for the two approaches. The proposed solutions exhibit small errors in the detection task in spite of the environment's complexity. The new methodology improves on the results obtained with the first network. However, by observing the position error ρ and its standard deviation, the errors obtained are larger than it would be expected. This is justified by the IoU (as computed in Sec. 4.2) not fully demonstrate the changes on the distortion given by ρ, *i.e.* $\mathbf{A}_g$ and $\mathbf{A}_{t+1}$ can be identical ($IoU(\mathbf{A}_g, \mathbf{A}_{t+1}) > 0.65$), even though the current state is further away from the ground truth, due to the distortion in central catadioptric systems being radial. Nonetheless, as we can also examine in Tab. 5.2, the position error of β is small, thus small changes in β will lower the IoU, for the same reason as before. The distortion in this kind of system will have a higher effect when the bounding box moves along the radius given by ρ.

From Tab. 5.3(a), we observe an improvement from "Ours" to "OursV2". The trained model with the multi-task network takes less steps to find the pedestrians (from 28.46 to 21.39 steps) and the final bounding box is closer to the ground truth (9.5% better). These differences are also seen during training. In Fig. 5.5, we observe the reward histograms from both networks during training. The second method (Fig. 5.5(b)) was substantially better than the first one (Fig. 5.5(a)), since the amount of rewards above



(a) Ours (b) OursV2

Figure 5.5: Fig.(a) and (b) show the reward histogram over training time. In both cases, we observe an increase of positive rewards, indicating that the network is being trained correctly. However, we can also observe that in (b), the network trained has better results than the "Ours", shown by amount of rewards above the trigger reward $\alpha = 10$.

α on "OursV2", on the last steps, are substantially bigger than "Ours", even though it trained for longer time. Training time was also of the areas that the new methodology improved on. It was reduced from 1.306×10^6 to 0.936×10^6 steps (a 28.3% decrease), as seen in Tab. 5.3(b).

Finally, I consider a correct detection when the trigger is signaled above a threshold[1]. For each method, the trained network signals a bounding box, that, afterwards, is compared to the corresponding ground truth. For the "SotA" and "SotAU", the ground truth is the perspective one, created in Sec. 5.1, and for "Ours" and "OursV2", the distorted one. In Tab. 5.3(b), it is shown the percentage of correct identifications done by each network. Initially, with "Ours", my network did not surpass the "SotA" and "SotAU" on the detection task. However, the latest approach outperforms 2.5% over the previous solutions.

[1]The same value (τ) used in training.

Chapter 6

Conclusion

In this thesis, I have presented a novel methodology for PD in omnidirectional environments. The approach uses the underlying geometry of general central omnidirectional cameras along with deep RL. From the extensive conducted evaluation, the methodology is able to accurately detect pedestrians without resorting to undistortion procedures which are computationally expensive. Moreover, my proposal can provide the 3D world position of the pedestrian instead of 2D image coordinates. It needs a smaller number of agent actions to reach the correct detection, and exhibits a higher accuracy on the final bounding box representation. This is due to the fact that our framework inherently holds high levels of distortions where the bounding box is represented in cylindrical coordinates which are tailored for these omnidirectional environments. The final solution has a better rate of detection and a better bounding box fitting than the previous discussed methods.

Further work can extend my approach to other object classes, and a generalization for non-central catadioptric systems. Different type of approaches for this problem can use other techniques (instead of deep RL), such as the newly developed Spherical CNNs (Cohen et al. (2018), Esteves et al. (2018)).

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. & Zheng, X. (2015), 'TensorFlow: Large-scale machine learning on heterogeneous systems'.
URL: *https://www.tensorflow.org/* 31

Arandjelovic, R., Gronat, P., Torii, A., Pajdla, T. & Sivic, J. (2016), NetVLAD: CNN architecture for weakly supervised place recognition, *in* 'IEEE Conf. Computer Vision and Pattern Recognition (CVPR)', pp. 5297–5307. 4

Baker, S. & Nayar, S. K. (1999), 'A Theory of Single-Viewpoint Catadioptric Image Formation', *Int'l J. Computer Vision (IJCV)* **35**(2), 175–196. 4

Barreto, J. P. & Araujo, H. (2001), Issues on the Geometry of Central Catadioptric Image Formation, *in* 'IEEE Conf. Computer Vision and Pattern Recognition (CVPR)', Vol. 2, pp. 422–427. 2, 9, 11, 32

Baxter, J. (1997), 'A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling', *Machine Learning* **28**(1), 7–39. 4

Bergen, T. & Wittenberg, T. (2016), 'Stitching and Surface Reconstruction From Endoscopic Image Sequences: A Review of Applications and Methods', *IEEE J. Biomedical and Health Informatics* **20**(1), 304–321. 2

Bordes, A., Glorot, X., Weston, J. & Bengio, Y. (2012), Joint Learning of Words and Meaning Representations for Open-Text Semantic Parsing, *in* 'Int'l Conf. ArtiiňĄcial Intelligence and Statistics (AISTATS)', Vol. 22, pp. 127–135. 1

Born, M. & Wolf, E. (1970), *Principles of Optics*, 4 edn, Pergamon Press. 9

Boult, T. E., Micheals, R., Gao, X., Lewis, P., Power, C., Yin, W. & Erkan, A. (1999), Frame-Rate Omnidirectional Surveillance & Tracking of Camouflaged and Occluded Targets, *in* 'IEEE Int'l Workshop on Visual Surveillance', pp. 48–55. 2

Caicedo, J. C. & Lazebnik, S. (2015), Active Object Localization with Deep Reinforcement Learning, *in* 'IEEE Int'l Conf. Computer Vision (ICCV)', pp. 2488–2496. 1, 3, 13, 21, 27, 31, 35, 37

Cinaroglui, I. & Bastanlar, Y. (2014), A Direct approach for human detection with catadioptric omnidirectional cameras, *in* 'IEEE Signal Processing and Communications Applications Conf.', pp. 2275–2279. 5

Cinaroglui, I. & Bastanlar, Y. (2016), 'A direct approach for object detection with catadioptric omnidirectional cameras', *Signal, Image and Video Processing* **10**(2), 413–420. 5

Cohen, T. S., Geiger, M., KÃűhler, J. & Welling, M. (2018), Spherical CNNs, *in* 'International Conference on Learning Representations (ICLR)'. 41

Collobert, R. & Weston, J. (2008), A unified architecture for natural language processing: Deep neural networks with multitask learning, *in* 'Int'l Conf. Machine learning (ICML)', pp. 160–167. 4

Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A. & Fei-Fei, L. (2012), 'Imagenet large scale visual recognition competition 2012'. Available: `http://www.image-net.org/challenges/LSVRC/2012/` [Online]. 3

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K. & Fei-Fei, L. (2009), ImageNet: A large-scale hierarchical image database, *in* 'IEEE Conf. Computer Vision and Pattern Recognition (CVPR)', pp. 248–255. 3

Eigen, D. & Fergus, R. (2015), Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture, *in* 'IEEE Int'l Conf. Computer Vision (ICCV)', pp. 2650–2658. 4

Esteves, C., Allen-Blanchette, C., Makadia, A. & Daniilidis, K. (2018), 'Learning SO(3) Equivariant Representations With Spherical CNNs', *arXiv preprint arXiv:1711.06721v2* . 41

Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J. & Zisserman, A. (2010), 'The PASCAL Visual Object Classes (VOC) Challenge', *Int'l J. Computer Vision (IJCV)* **88**(2), 303–338. 1

Everingham, M., Gool, L. V., Williams, C. K., Winn, J. & Zisserman, A. (2012), 'The Pascal Visual Object Classes (VOC) Challenge', *Int'l J. Computer Vision (IJCV)* **88**(2), 303–338. 3

Furnari, A., Farinella, G. M., Bruna, A. R. & Battiato, S. (2017), 'Affine Covariant Features for Fisheye Distortion Local Modeling', *IEEE Trans. Image Processing (T-IP)* **26**(2), 696–710. 5

Gandhi, T. & Trivedi, M. (2005), 'Parametric ego-motion estimation for vehicle surround analysis using an omnidirectional camera', *Machine Vision and Applications* **16**(2), 85–95. 5

Geyer, C. & Daniilidis, K. (2000), A Unifying Theory for Central Panoramic Systems and Practical Implications, *in* 'European Conf. Computer Vision', pp. 445–461. 2, 9

Ghesu, F. C., Georgescu, B., Mansi, T., Neumann, D., Hornegger, J. & Comaniciu, D. (2016), An artificial agent for anatomical landmark detection in medical images, *in* 'Int'l Conf. Medical Image Computing and comp.-Assisted Intervention (MICCAI)', pp. 229–237. 3, 21

Girshick, R. (2015), Fast R-CNN, *in* 'IEEE Int'l Conf. Computer Vision (ICCV)', pp. 1440–1448. 1, 3, 18

Girshick, R., Donahue, J., Darrell, T. & Malik, J. (2014), Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, *in* 'IEEE Conf. computer Vision and Pattern Recognition (CVPR)', pp. 580–587. 1, 3, 18

Hartley, R. & Zisserman, A. (2003), *Multiple View Geometry in comp. Vision*, Cambridge University Press. 10, 11

Hasselt, H. V. (2010), Double Q-learning, *in* 'Advances in Neural Information Processing Systems (NIPS)', pp. 2613–2621. 15

Hasselt, H. V., Guez, A. & Silver, D. (2016), Deep Reinforcement Learning with Double Q-Learning., *in* 'AAAI Conf. Artificial Intelligence', Vol. 16, pp. 2094–2100. 15, 16, 29

He, K., Gkioxari, G., Dollar, P. & Girshick, R. (2017), Mask R-CNN, *in* 'IEEE Int'l Conf. Computer Vision (ICCV)', pp. 2980–2988. 3, 18, 27

He, K., Zhang, X., Ren, S. & Sun, J. (2014), Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, *in* 'European Conf. Computer Vision (ECCV)', pp. 345–361. 1, 3, 18

Hosang, J., Benenson, R. & Schiele, B. (2014), How good are detection proposals, really?, *in* 'British Machine Vision Conf. (CVPR)'. 3

Iraqui, A., Dupuis, Y., Boutteau, R., Ertaud, J. Y. & Savatier, X. (2010), Fusion of Omnidirectional and PTZ Cameras for Face Detection and Tracking, *in* 'Int'l Conf. Emerging Security Technologies', pp. 18–23. 2, 5

Itseez (2015), 'Open Source Computer Vision Library'.
URL: *https://github.com/itseez/opencv* 31

Jaderberg, M., Simonyan, K., Zisserman, A. & Kavukcuoglu, K. (2015), Spatial Transformer Networks, *in* 'Advances in Neural Information Processing Systems (NIPS)', pp. 2017–2025. 2

Jamzad, M., Hadjkhodabakhshi, A. & Mirrokni, V. (2007), 'Object Detection and Localization Using Omnidirectional Vision in the RoboCup Environment', *Scientia Iranica* **14**(6), 599–611. 5

Kannala, J. & Brandt, S. S. (2006), 'A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses', *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)* **28**(8), 1335–1340. 4

Kendall, A., Gal, Y. & Cipolla, R. (2018), Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics, *in* 'IEEE Conf. Computer Vision and Pattern Recognition (CVPR)'. 3

Kendall, A., Grimes, M. & Cipolla, R. (2015), Posenet: A convolutional network for real-time 6-dof camera relocalization, *in* 'IEEE Int'l Conf. Computer Vision (ICCV)', IEEE, pp. 2938–2946. 4

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A. et al. (2017), 'Overcoming catastrophic forgetting in neural networks', *Proc. National Academy of Sciences* **114**(13), 3521–3526. 4

Krizhevsky, A., Sutskever, I. & Hinton, G. E. (2012), ImageNet Classification with Deep Convolutional Neural Networks, *in* 'Advances in Neural Information Processing Systems (NIPS)', pp. 1097–1105. 1, 3

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P. & Zitnick, C. L. (2014), Microsoft COCO: Common Objects in Context, *in* 'European Conf. Computer Vision (ECCV)', pp. 740–755. 1

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. & Berg, A. C. (2016), SSD: Single Shot MultiBox Detector, *in* 'European Conf. Computer Vision (ECCV)', pp. 21–37. 1, 3, 18

Liu, X., Zhao, H., Tian, M., Sheng, L., Shao, J., Yi, S., Yan, J. & Wang, X. (2017), HydraPlus-Net: Attentive Deep Features for Pedestrian Analysis, *in* 'IEEE Int'l Conf. Computer Vision (ICCV)', pp. 740–755. 1

Lourenço, M., Barreto, J. P., Fonseca, F., Ferreira, H., Duarte, R. M. & Correia-Pinto, J. (2014), Continuous Zoom Calibration by Tracking Salient Points in Endoscopic Video, *in* 'Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI)', pp. 456–463. 2

Lourenco, M., Barreto, J. P. & Vasconcelos, F. (2012), 'sRD-SIFT: Keypoint Detection and Matching in Images With Radial Distortion', *IEEE Trans. Robotics (T-RO)* **28**(3), 752–760. 5

Maicas, G., Carneiro, G., Bradley, A., Nascimento, J. C. & Reid, I. (2017), Deep Reinforcement Learning for Active Breast Lesion Detection from DCE-MRI, *in* 'Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI)', pp. 665–673. 3

Mei, C. & Rives, P. (2007), Single View Point Omnidirectional Camera Calibration from Planar Grids, *in* 'IEEE Int'l Conf. Robotics and Automation (ICRA)', pp. 3945–3950. 2, 9, 31

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. & Dean, J. (2013), Distributed Representations of Words and Phrases and their Compositionality, *in* 'Advances in Neural Information Processing Systems (NIPS)', pp. 3111–3119. 1

Miraldo, P., Eiras, F. & Ramalingam, S. (2018), Analytical Modeling of Vanishing Points and Curves in Catadioptric Cameras, *in* 'IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)', p. to appear. 2, 4

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S. & Hassabis, D. (2015), 'Human-level control through deep reinforcement learning', *Nature* **517**, 529–533. 1, 3, 13, 15, 27

Mottaghi, R., Chen, X., Liu, X., Cho, N. G., Lee, S. W., Fidler, S., Urtasun, R. & Yuille, A. (2014), The Role of Context for Object Detection and Semantic Segmentation in the Wild, *in* 'IEEE Conf. Computer Vision and Pattern Recognition (CVPR)', pp. 891–898. 1

Nalwa, V. (1996), A true omnidirectional viewer, Technical report, Bell Laboratory. 2, 4

Nayar, S. K. (1997), Catadioptric omnidirectional camera, *in* 'IEEE Conf. Computer Vision and Pattern Recognition (CVPR)', pp. 482–488. 4

Nayar, S. K. & Baker, S. (1997), Catadioptric image formation, *in* 'DARPA Image Understanding Workshop'. 2, 4

Neves, A. J. R., Pinho, A. J., Martins, D. A. & Cunha, B. (2011), 'An efficient omnidirectional vision system for soccer robots: From calibration to object detection', *Mechatronics* **21**(2), 399–410. 2, 5

PointGrey (2017), 'Flea3 USB3 Vision cameras for industrial, life science, traffic, and security applications.'.
URL: *https://www.ptgrey.com/flea3-usb3-vision-cameras* 31

Puig, L. & Guerrero, J. J. (2011), Scale Space for Central Catadioptric Systems: Towards a Generic Camera Feature Extractor, *in* 'IEEE Int'l Conf. Computer Vision (ICCV)', pp. 1599–1606. 5

Redmon, J. & Farhadi, A. (2018), 'YOLOv3: An Incremental Improvement', *arXiv preprint arXiv:1804.02767* . 1, 3, 18

Ren, H., Li, Z.-N. & Fraser, S. (2015), Object Detection Using Generalization and Efficiency Balanced Co-occurrence Features, *in* 'IEEE Int'l Conf. Computer Vision (ICCV)', pp. 46–54. 1, 27

Ren, J. S. J. & X, L. (2015), On Vectorization of Deep Convolutional Neural Networks for Vision Tasks, *in* 'AAAI Conf. Artificial Intelligence', pp. 1840–1846. 1

Ren, S., He, K., Girshick, R. & Sun, J. (2015), Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, *in* 'Advances Neural Information Processing Systems (NIPS)', pp. 91–99. 1, 3, 18

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C. & Fei-Fei, L. (2015), 'ImageNet Large Scale Visual Recognition Challenge', *Int'l J. Computer Vision (IJCV)* **115**(3), 211–252. 1

Schechner, Y. Y. & Nayar, S. K. (2001), Generalized mosaicing, *in* 'IEEE Int'l Conf. Computer Vision (ICCV)', Vol. 1, pp. 17–24. 5

Sutton, R. (1990), Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming, *in* 'Machine Learning Proc.', pp. 216–224. 17

Sutton, R. & Barto, A. G. (1998), *Reinforcement learning: An introduction*, Vol. 2, MIT press. 14

Swaninathan, R., Grossberg, M. D. & Nayar, S. K. (2003), A Perspective on Distortions, *in* 'IEEE Conf. Computer Vision and Pattern Recognition (CVPR)', Vol. 2, pp. 594–601. 2, 4

Tang, Y., Li, Y., Bai, T., Zhou, X. & Li, Z. (2011), Human tracking in thermal catadioptric omnidirectional vision, *in* 'IEEE Int'l Conf. Information and Automation (ICIA)', pp. 97–102. 5

Thrun, S. (1996), Is learning the n-th thing any easier than learning the first?, *in* 'Advances in Neural Information Processing Systems (NIPS)', pp. 640–646. 4

Thrun, S. & Schwartz, A. (1993), Issues in using function approximation for reinforcement learning, *in* 'Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum'. 15

Wang, M. & Lin, H. (2009), Object Recognition from Omnidirectional Visual Sensing for Mobile Robot Applications, *in* 'IEEE Int'l Conf. Systems, Man and Cybernetics', pp. 1941–1946. 2, 5

Yamazawa, K. & Yokoya, N. (2003), Detecting moving objects from omnidirectional dynamic images based on adaptive background subtraction, *in* 'IEEE Int'l Conf. Image Processing (ICIP)', Vol. 3, pp. III–953–6. 5

Ying, X. & Hu, Z. (2004), Can we consider central catadioptric cameras and fisheye cameras within a unified imaging model, *in* 'European Conf. Computer Vision (ECCV)', pp. 442–455. 9

Zhang, Z., Rebecq, H., Forster, C. & Scaramuzza, D. (2016), Benefit of large field-of-view cameras for visual odometry, *in* 'IEEE Int'l Conf. Robotics and Automation (ICRA)', pp. 801–808. 2

Appendix A

Intersection over Union Computation

Let us consider, $\mathbf{A}$ to be the pixels that lie within the lines of the distorted bounding box. By using functions of the `OpenCV 3.1 Python` library, it is possible to retrieve the areas defined by the different $\mathbf{A}$.

Assuming that the state $\mathbf{s}_{t+1}$ has a pixel representation $\mathbf{A}_{t+1}$ and the ground truth $\mathbf{s}_g$ has $\mathbf{A}_g$, then we can calculate the areas of each state, $area(\mathbf{A}_{t+1})$ and $area(\mathbf{A}_g)$, respectively. By applying a bitwise combination of the two pixel representations, one extracts the intersection of those two states $\mathbf{A}_{intersection}$ and the respective $area(\mathbf{A}_{intersection})$. At last, the union area is computed from the areas above:

$$area(\text{Union}) = area(\mathbf{A}_{t+1}) + area(\mathbf{A}_g) - area(\mathbf{A}_{intersection}). \tag{A.1}$$

Finally, the IoU can be computed as:

$$IoU(\mathbf{A}_{t+1}, \mathbf{A}_g) = \frac{area(\mathbf{A}_{intersection})}{area(\text{Union})}. \tag{A.2}$$