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

Gonçalo José Dias Pais
Instituto Superior Técnico, Lisboa
goncalo.pais@tecnico.ulisboa.pt

***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 and classification. One class of such methods is based on deep Reinforcement Learning. However, for 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 paper, we present a novel efficient technique for robust pedestrian detection, using omnidirectional catadioptric camera systems with deep Reinforcement Learning. Our method is tested and compared with current related approaches that do not consider the underlying distortions. Beyond the novel solution, our method improves significantly the results when compared with state-of-the-art methodologies.**

***Index Terms*—Pedestrian detection, omnidirectional vision, central catadioptric camera systems, convolutional neural networks, deep reinforcement learning.**

I. 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 ([1]), along with the availability of larger image datasets (such as [2], [3]).

Nowadays, Pedestrian Detection (PD) is mainly addressed using Deep Learning (DL) techniques ([4]) 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 ([5]); Transfer Learning ([6]); Natural Language processing ([7]); and Computer Vision ([8]) 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) ([8]), which halved the error rate for image classification. Other related techniques and extensions are also available, *e.g.* [9]–[14]. 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 ([15], [16]).

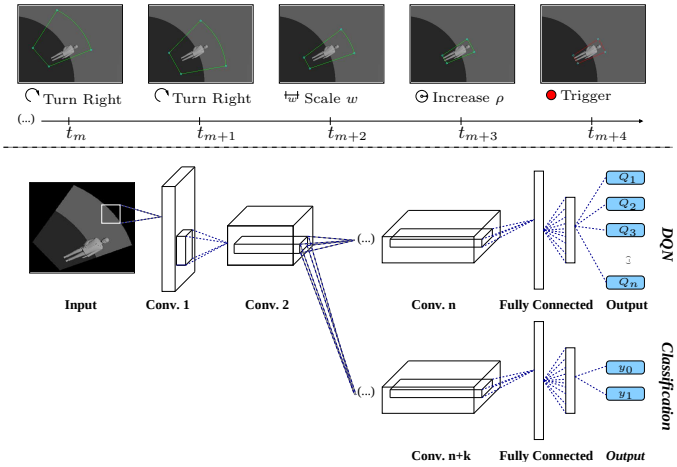


Fig. 1: Problem addressed in this thesis, and the proposed solution for the RL using omnidirectional cameras. At the top, it is presented the results of the proposed technique to a dataset of synthetic images (I am showing only the final steps of the method); and, at the bottom, it is shown the scheme of the proposed network, where the first convolutional layers are shared, and then split into branches (DQN and Classification).

In the context of omnidirectional cameras, we also have assisted to an increasing number of applications, ranging from surveillance to medical imaging applications ([17]–[20]), which require a wide Field of View (FoV) of the environment, usually obtained by means of distortion ([21]). For PD using these type of sensors, one could first apply undistortion techniques to get a perspective image, and then conventional detection tools ([22]–[24]). 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 (see Fig. 1), I overcome the above mentioned issues by performing the PD directly in distorted images, that

* G. Pais is with the Institute for Systems and Robotics (LARSyS), Instituto Superior Técnico, University of Lisbon. This project was partially supported by FCT project [UID/EEA/50009/2013]. The author wants to thank to his supervisors Dr. Pedro Miraldo & Prof. Jacinto C. Nascimento and his lab colleagues (room 6.13 of the IST's North Tower).

[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 [25]).

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.

A. State-of-the-Art

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

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 ([26]), we can easily conclude that the progress slowed from 2010 onward. However, the use of CNNs ([8]) has led to a significant improvement in the accuracy of image classification, for the Imagenet Large Scale Visual Recognition Challenge ([27]). Soon, more methodologies became available, SppNet ([9]), R-CNN ([10], [28]), and its descendants, namely, Mask ([29]), Fast ([11]), Faster R-CNN([12]), SSD ([13]) and YOLOv3 ([14]). 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 ([13], [14]) 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 ([16], [30]). [16] have proposed the use of a Deep Q-Network (DQN) ([15]) 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.* [30], [31]. 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 ([32]), Surface Modeling & Segmentation ([33]), and even 3D Pose Estimation ([34]).

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 deep RL is unusual, however it has been useful to overcome forgetful actions over time ([35]). Alternatively, I propose a hard[2] parameter sharing network ([36]). 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 ([37]). 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 ([38]).

2) Omnidirectional Vision: Omnidirectional cameras 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 ([39]). The latter uses mirror(s) such as parabolic, elliptical, or hyperbolic to obtain a larger FoV ([40]). Image formation for these systems has been studied in the literature for more than 20 years. In [41], 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 ([21]).

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 requires a lot of computational effort. In addition, by merging images from different views, details and properties of the environment are lost ([42]), which in an omnidirectional system will not happen.

A small number of authors addressed the object/person detection directly on omnidirectional images. The work that is most related to my goals are: [43], in which the authors adopt the conventional camera approach that uses sliding windows and Histogram of Gradients (HOG) 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 am aiming at avoiding the above shortcomings.

Although there have been some advances in feature extraction for omnidirectional catadioptric systems (*e.g.* [44]), 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 ([17]) but require knowledge

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

of the object shape and color. There are also tracking methods for omnidirectional cameras based on object's motion, by using background subtraction ([45]) and its egomotion ([46]).

Other works address this problem by first applying transformations to the images followed by conventional techniques, *e.g.* [47]. In [48], 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 [49].

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, 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 for an illustration of the proposed method in a synthetic example and the network's representation.

II. 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. II-A), and then the projection of 3D line segments, required to define the bounding box projection (Sec. II-B).

A. Image Formation

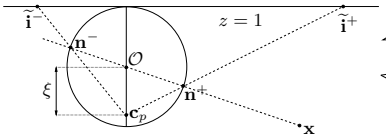
To deal with general omnidirectional cameras, it is used the spherical model firstly proposed by [22] and modified by [23] and [24]. Notice that, this model can also be used to represent fisheye cameras ([24], [50]). Assuming a unit sphere centered at the origin of the of mirror's reference frame, a 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:

$$\left\{\mathbf{n}^+, \mathbf{n}^-\right\} \doteq \Omega(\mathbf{x}) = \left(\pm\frac{x_1}{r}, \pm\frac{x_2}{r}, \pm\frac{x_3}{r}\right), \tag{1}$$

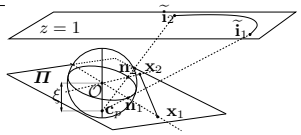
where $r = \sqrt{x_1^2 + x_2^2 + x_3^2}$, (see Fig. 2(a) for more details). The antipodal point closer to $\mathbf{x}$ (*i.e.* $\mathbf{n}^+$ as the convention in the figure) is chosen to be the point projected to the image plane (which is true according to the *Fermat's* principle [51]).

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 $\mathcal{H} : \mathbb{R}^3 \to \mathbb{R}^3$:

$$\mathbf{n}_p^+ = \mathcal{H}(\mathbf{x}, \xi) = \left(\frac{x_1}{r}, \frac{x_2}{r}, \frac{x_3 + \xi}{r}\right). \tag{2}$$



(a) Image Formation



(b) Projection of 3D Lines

Fig. 2: This figure shows the image formation using unified central catadioptric cameras. In (a), it is shown the projection of a point $\mathbf{R} \in \mathbb{R}^3$ 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:

$$\widetilde{\mathbf{i}} = \widetilde{\mathbf{i}}^+ = \mathcal{F}(\mathbf{x}, \xi) = \left(\frac{n_{p,x}^+}{n_{p,z}^+}, \frac{n_{p,y}^+}{n_{p,z}^+}\right). \tag{3}$$

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

$$\mathbf{i} = \begin{bmatrix} f_1\eta & f_1\eta\alpha \\ 0 & f_2\eta \end{bmatrix} \widetilde{\mathbf{i}} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}, \tag{4}$$

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 [52]). 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 camera.

B. 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 – [52]). Therefore, since I am interested in central omnidirectional cameras, 3D straight lines are parameterized by a plane $\boldsymbol{\Pi}$, which comprises by the center of the projection sphere ($\mathcal{O}$ in Fig. 2(b)) and the respective 3D straight line (notice that, without lost of generality, all lines on this plane will be equally parameterized).

By definition and 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 [23]. 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 $\boldsymbol{\Pi}$ and the unit sphere. See Fig. 2(b)) for more details.

After some manipulations, one can obtain a quadric equation $\mathbf{x}^T\mathbf{C}\mathbf{x} = 0$, where:

$$\mathbf{C} = \begin{bmatrix} l_1^2\left(1-\xi^2\right) - l_3^2\xi^2 & l_1 l_2\left(1-\xi^2\right) & l_1 l_3 \\ l_1 l_2\left(1-\xi^2\right) & l_2^2\left(1-\xi^2\right) - l_3^2\eta^2 & l_2 l_3 \\ l_1 l_2 & l_2 l_3 & l_3^2 \end{bmatrix}, \quad (5)$$

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) into the plane by:

$$\begin{aligned} \mathcal{L} = \mathcal{G}(\mathbf{x}_1, \mathbf{x}_2) = \{\ (\widetilde{i}_x, \widetilde{i}_y) \in \mathbb{R}^2\ :\ \\ \begin{bmatrix}\widetilde{i}_x & \widetilde{i}_y & 1\end{bmatrix}^T \mathbf{C} \begin{bmatrix}\widetilde{i}_x & \widetilde{i}_y & 1\end{bmatrix} = 0 \wedge \\ \widetilde{i}_{1,x} < \widetilde{i}_x < \widetilde{i}_{2,x} \wedge \\ \widetilde{i}_{1,y} < \widetilde{i}_y < \widetilde{i}_{2,y}\ \}, \end{aligned} \quad (6)$$

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 (3) and also Fig. 2(b)). Finally, to the points that belong to the curve, a final mapping must be applied to project them into the image plane, which is computed by applying (4).

III. Deep Reinforcement Learning and Classification

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

A. Deep Reinforcement Learning

This section describes how object detection is performed using deep RL which has been a promising and thriving path of research. Recently, [15], [16] 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.* [16]) 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.

1) Training: The training process of the DQN follows a traditional Markov Decision Process (known as MDP) 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:

$$Q^\star(s, a) = \max_{\pi}\ \mathbb{E}[R_t | s_t = s, a_t = a, \pi], \quad (7)$$

which is achievable by a policy $\pi = p(a|s)$, and $Q^\star(.)$ representing the *quality* of performing an a in s.

Obtaining $Q^\star(s, a)$ can be achieved using the *Bellman* equation and the Q-Learning algorithm (see [53] 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:

$$Q(s_t, a_t; \boldsymbol{\theta}_t) = \mathbb{E}_{s_{t+1}}\Big[r_t + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \boldsymbol{\theta}_t) | s_t, a_t\Big]. \quad (8)$$

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 ([15]). 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 ([54]). However, in [55] it was introduced Double Q-learning, that was later applied in the context of deep learning, giving rise to the so called Double DQN (DDQN – [56]), 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 ([56]).

The goal here is to change the DQN algorithm as minimum as possible, *i.e.* the training algorithm remains the same, but

incorporates the new update rule. To be more specific, in the DQN, I have:

$$q_j^{DQN} = r_j + \gamma \max_{a_{j+1}} Q(\mathbf{s}_{j+1}, a_{j+1}; \boldsymbol{\theta}_t^-), \tag{9}$$

which is now replaced by:

$$q_j^{DDQN} = r_t + \gamma Q(s_{t+1}, \arg\max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \boldsymbol{\theta}_t); \boldsymbol{\theta}_t^-). \tag{10}$$

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:

$$L_{drl}(\boldsymbol{\theta}_t) = \mathbb{E}_{(s_t, a_t, r_t, s_{t+1}) \sim U(\mathbf{D}_t)} \left[(q_j^{DDQN} - Q(s_t, a_t; \boldsymbol{\theta}_t))^2 \right], \tag{11}$$

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 $\boldsymbol{\theta}_0^- = \boldsymbol{\theta}_0$.

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*, which I will describe below.

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 ([57]). Instead of always computing the optimal action, this strategy chooses an action based on a weighted probability through its Q-values, that is:

$$p(Q(s_t, a_t^i; \boldsymbol{\theta}_t), \epsilon) = \frac{e^{\frac{Q(s_t, a_t^i; \boldsymbol{\theta}_t)}{\epsilon}}}{\sum_{i=0}^{N-1} e^{\frac{Q(s_t, a_t^i; \boldsymbol{\theta}_t)}{\epsilon}}}, \tag{12}$$

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 (12), the optimal action is the most likely chosen, but many sub-optimal are considered also.

B. Inference

The trained DQN model is parameterized by $\boldsymbol{\theta}^\star$ learned in (11) that outputs the action-value function for the state observation s_t. Formally, the action to follow from the current observation is defined by:

$$a_t^\star = \arg\max_{a_t} Q(s_t, a_t; \boldsymbol{\theta}^\star). \tag{13}$$

Finally, given that the number and the 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.

C. Classification

In this section, I describe how the object classification network is trained. The main goal in a classification task is to assign a certain object to a class, within a proposed region. These regions are: rectangular bounding box ([12]), segmented region ([29]) 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 a CNN to model this behavior, by minimizing the classification error of a batch of labeled images according to cross-entropy's mean:

$$L_{cls}(\boldsymbol{\theta}_t) = \mathbb{E}_{\sim U(\mathbf{D}_t)} \left[-y_c \log(p(\widehat{y}_1; \boldsymbol{\theta}_t)) - (1 - y_c) \log(p(\widehat{y}_0; \boldsymbol{\theta}_t)) \right], \tag{14}$$

where y_c is the labeled class; $p_i(\widehat{y}_i; \boldsymbol{\theta}_t)$ is the calculated 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. This probability is computed through the Softmax function of the network's output:

$$p(\widehat{y}_i; \boldsymbol{\theta}_t) = \frac{e^{\widehat{y}_i}}{\sum_{i=0}^{1} e^{\widehat{y}_i}}, \quad i \in \{0, 1\}. \tag{15}$$

For inference, it is simply chosen i that has the highest $p_i(\widehat{y}_i; \boldsymbol{\theta}_t)$:

$$y_c^\star = \arg\max_i p(\widehat{y}_i; \boldsymbol{\theta}^\star), \tag{16}$$

where $\boldsymbol{\theta}^\star$ are the optimal parameters of the model learned in (14).

IV. 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. IV-A, the bounding box shape and position that characterizes the agent's states in the world and its respective projection are detailed. Second, in Sec. IV-B, 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. IV-C.

A. 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.*, [16], [31]), 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[3], which ideally is located at the pedestrian's feet, and a dimension $\mathbf{dim}_t = [w, h]^T$ for the width and height of the bounding box that fits the person, as shown in Fig 3. 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:

$$\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4\} = \mathcal{M}\left(\mathbf{p}_t, \mathbf{dim}_t\right), \tag{17}$$

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

$$\mathcal{L}_{i,j} = \mathcal{G}(\mathbf{x}_i, \mathbf{x}_j), \text{with } (i,j) \in \{(1,2),\ (1,3),\ (2,4),\ (3,4)\}. \tag{18}$$

Finally, $\mathcal{L}_{i,j}$ are projected into the image, by using (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 Fig. 1. States are defined as:

$$\mathbf{s}_t = \mathcal{I}\left(\mathcal{L}_{i,j}\right) \in \mathbf{S}, \text{with } (i,j) \in \{(1,2),\ (1,3),\ (2,4),\ (3,4)\}, \tag{19}$$

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

B. 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 of the β, ρ, h, and w as shown in Fig. 3, and the trigger σ, such that:

$$a_t \in \mathcal{A} = \{\rho^+, \rho^-, \beta^+, \beta^-, w^+, w^-, h^+, h^-, \sigma\}, \tag{20}$$

where the superscripts $+$ and $-$ 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. I. 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 (19).

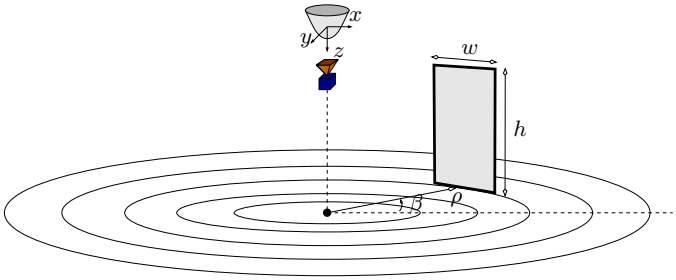


Fig. 3: Illustration of the environment defined in this work. The camera position w.r.t. to the world coordinate system is assumed to be known, which means that the position of the bounding both in the world can be defined in cylindrical coordinates (assuming that the pedestrian is in vertical position in the ground plan).

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

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:

$$r_t = \mathcal{R}(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}) = \begin{cases} +\alpha & \text{if } \ IoU(\mathbf{A}_{t+1}, \mathbf{A}_g) \geq \tau \\ -\alpha & \text{otherwise} \end{cases}, \tag{21}$$

where $IoU(.) \geq 0$ is the Intersection over Union (IoU), 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. Thus, the IoU is performed between the estimated detection $\mathbf{A}_{t+1}$ and the available ground truth $\mathbf{A}_g$[4].

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 dimension is modified, such that the projection is closer to

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

[4]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.

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:

$$r_t = \mathcal{R}(\mathbf{s}_t, a_t, \mathbf{s}_{t+1}) = sign(IoU(\mathbf{A}_{t+1}, \mathbf{A}_g) - IoU(\mathbf{A}_t, \mathbf{A}_g)). \quad (22)$$

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.

The training process follows (7) that models the DQN by maximizing the cumulative rewards. As above mentioned, this relies on the experience-replay memory and the target network ([15]). 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 ([15]). The DQN is trained from scratch, *i.e.* the network was trained to extract features from omnidirectional camera systems.

C. Multi-task Training

The problem with my method is that it requires an initial estimate of the pedestrian's position. [16] 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. However, to overcome this issue, the proposed method uses a classification network (Sec. III-C) 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, 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 ([1], [29]), 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 detection 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. 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 (11).

V. Experimental Results

In this section, I describe how the omnidirectional images' dataset is created and how the network architecture is designed. Then, I compare the newly developed agent with the perspective state-of-the-art method. I perform a comparison between: 1) my bounding boxes methodology, set in the environment discussed in Sec. IV (denoted "Ours" and "OursV2"); and 2) with the method proposed in [16] 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. The new pipeline for testing using the developed multi-task network is called "OursV2". My methodology is developed using `Python` with `TensorFlow` ([58]) and `OpenCV` ([59]).

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 our laboratory with several subjects. 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 [60]'s Flea3 camera attached to a hyperboloid mirror, that can be modeled by a central catadioptric camera system. The imaging device was calibrated using the framework presented in [24].

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. IV-A. 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", "SotA" and "SotAU" have five convolutional layers and two fully-connected layers, where the input is a 224×224 image size, and the output contains the Q-values for the nine possible actions (see Sec. IV-B). Notice that we can maintain the same dimension of actions as in [16], thus we can keep the same architecture to train both methods.

TABLE II: This table shows the average steps in the test sequence to set the trigger correctly for the different methods, the average IoU, that evaluate the distorted image on the distorted ground truth, the correct percentage of detections on the test dataset, and the final training steps. 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	Correct (%)	Training Steps
SotA	62.65	0.385	83.9	0.975×10^6
SotAU	28.464	–	77.2	1.385×10^6
Ours	27.85	0.623	71.3	1.306×10^6
OursV2	**21.39**	**0.718**	**86.5**	$\mathbf{0.936 \times 10^6}$

The three networks are trained from scratch with a ϵ-greedy exploration. "OursV2" has three shared convolutional layers, two more convolutional layers and two fully-connected layers for each branch. This network tackles the issues discussed in Sec. IV-B, by creating a pipeline that uses both the deep RL and the classifications tasks, when testing to develop an end-to-end solution. In addition to the nine Q-values output, in the "OursV2" method, the Softmax probability is also retrieved for the two classes, in the classification network. This network is trained from scratch, with a boltzmann exploration. The training parameters are also the same for all the above networks.

For the test initialization of "Ours', 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 at detecting some pedestrian location. For "OursV2" the testing pipeline starts with the same approach as before. 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 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.

For evaluation, I used the test dataset 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. II, 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. An example illustrating the steps required for each algorithm is shown in Fig. 4. 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

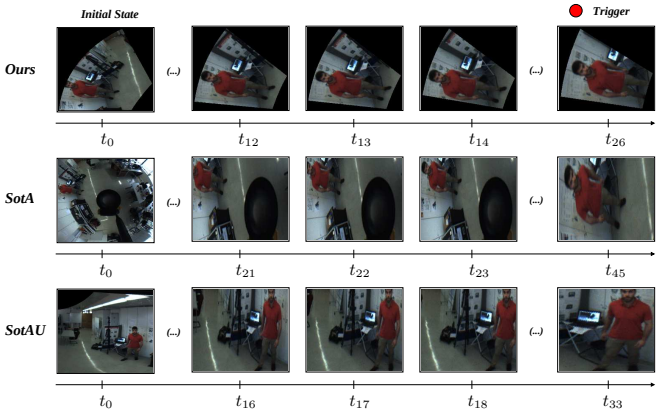


Fig. 4: Illustration of the states evolution for the detection. "Ours" method (first row), "SotA" method (second row) both performed in omnidirectional images, and "SotAU" method (third row), performed in undistorted images. Notice that with my method, the bounding box takes into account the distortion, 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. For "OursV2" method, it would result in a similar experiment as in "Ours", thus not shown here.

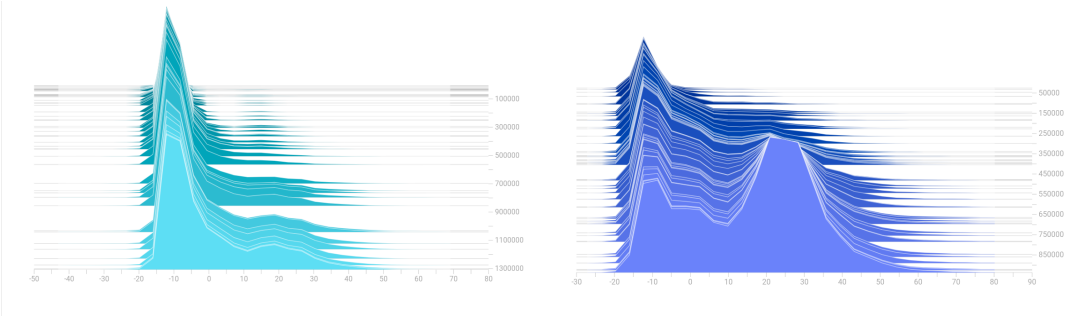
(result of the shortcomings discussed in Sec. I).

Also note that, my methods reach to an improvement of over 23.8% regarding the "SotA" method in relation to the average IoU. Also, from Tab. II, 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 improvements are also seen during training. In Fig. 5, we observe the reward histograms from both networks during training. The second method (Fig. 5(b)) was substantially better than the first one (Fig. 5(a)), since the amount of rewards above α on "OursV2", on the last steps, are substantially larger than "Ours", even though it trained for longer time than the multi-task network. It was reduced from 1.306×10^6 to 0.936×10^6 steps (a 28.3% decrease), as seen in The proposed method "OursV2" triggers the detection using a smaller number of actions and deals better with the distortion when compared to both "Ours", "SotA" and "SotAU" methodologies.

I consider a correct detection when the trigger is signaled above a threshold[5]. 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, and for "Ours" and "OursV2", the distorted one. In Tab. II, it is shown the percentage of correct identifications done by each network. Initially, with "Ours", the network did not surpass the "SotA" and "SotAU" on the detection task. However, the latest approach outperforms 2.5% over the previous solutions.

Finally, Tab. III 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

[5]The same value τ used in training.



(a) Ours (b) OursV2

Fig. 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 $\alpha = 10$.

TABLE III: 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 our 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 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. IV-B) 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. III, 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.

VI. 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 ([61], [62]).

References

[1] H. Ren, Z.-N. Li, and S. Fraser, "Object Detection Using Generalization and Efficiency Balanced Co-occurrence Features," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2015, pp. 46–54. 1, 7

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *Int'l J. Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. 1

[3] T. Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollar, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," in *European Conf. Computer Vision (ECCV)*, 2014, pp. 740–755. 1

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

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

[6] J. S. J. Ren and L. X, "On Vectorization of Deep Convolutional Neural Networks for Vision Tasks," in *AAAI Conf. Artificial Intelligence*, 2015, pp. 1840–1846. 1

[7] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 3111–3119. 1

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105. 1, 2

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in *European Conf. Computer Vision (ECCV)*, 2014, pp. 345–361. 1, 2

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *IEEE Conf. computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587. 1, 2

[11] R. Girshick, "Fast R-CNN," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2015, pp. 1440–1448. 1, 2

[12] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99. 1, 2, 5

[13] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *European Conf. Computer Vision (ECCV)*, 2016, pp. 21–37. 1, 2

[14] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018. 1, 2

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 517, pp. 529–533, 2015. 1, 2, 4, 7

[16] J. C. Caicedo and S. Lazebnik, "Active Object Localization with Deep Reinforcement Learning," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2015, pp. 2488–2496. 1, 2, 4, 6, 7

[17] A. J. R. Neves, A. J. Pinho, D. A. Martins, and B. Cunha, "An efficient omnidirectional vision system for soccer robots: From calibration to object detection," *Mechatronics*, vol. 21, no. 2, pp. 399–410, 2011. 1, 2

[18] M. Lourenço, J. P. Barreto, F. Fonseca, H. Ferreira, R. M. Duarte, and J. Correia-Pinto, "Continuous Zoom Calibration by Tracking Salient Points in Endoscopic Video," in *Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2014, pp. 456–463. 1

[19] T. Bergen and T. Wittenberg, "Stitching and Surface Reconstruction From Endoscopic Image Sequences: A Review of Applications and Methods," *IEEE J. Biomedical and Health Informatics*, vol. 20, no. 1, pp. 304–321, 2016. 1

[20] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza, "Benefit of large field-of-view cameras for visual odometry," in *IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2016, pp. 801–808. 1

[21] R. Swaninathan, M. D. Grossberg, and S. K. Nayar, "A Perspective on Distortions," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2003, pp. 594–601. 1, 2

[22] C. Geyer and K. Daniilidis, "A Unifying Theory for Central Panoramic Systems and Practical Implications," in *European Conf. Computer Vision*, 2000, pp. 445–461. 1, 3

[23] J. P. Barreto and H. Araujo, "Issues on the Geometry of Central Catadioptric Image Formation," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2001, pp. 422–427. 1, 3

[24] C. Mei and P. Rives, "Single View Point Omnidirectional Camera Calibration from Planar Grids," in *IEEE Int'l Conf. Robotics and Automation (ICRA)*, 2007, pp. 3945–3950. 1, 3, 7

[25] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, "Spatial Transformer Networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 2017–2025. 1

[26] M. Everingham, L. V. Gool, C. K. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *Int'l J. Computer Vision (IJCV)*, vol. 88, no. 2, pp. 303–338, 2012. 2

[27] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and L. Fei-Fei, "Imagenet large scale visual recognition competition 2012," 2012, available: http://www.image-net.org/challenges/LSVRC/2012/ [Online]. 2

[28] J. Hosang, R. Benenson, and B. Schiele, "How good are detection proposals, really?" in *British Machine Vision Conf. (CVPR)*, 2014. 2

[29] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2017, pp. 2980–2988. 2, 5, 7

[30] G. Maicas, G. Carneiro, A. Bradley, J. C. Nascimento, and I. Reid, "Deep Reinforcement Learning for Active Breast Lesion Detection from DCE-MRI," in *Int'l Conf. Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2017, pp. 665–673. 2

[31] F. C. Ghesu, B. Georgescu, T. Mansi, D. Neumann, J. Hornegger, and D. Comaniciu, "An artificial agent for anatomical landmark detection in medical images," in *Int'l Conf. Medical Image Computing and comp.-Assisted Intervention (MICCAI)*, 2016, pp. 229–237. 2, 6

[32] A. Kendall, Y. Gal, and R. Cipolla, "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[33] D. Eigen and R. Fergus, "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture," in *IEEE Int'l Conf. Computer Vision (ICCV)*, 2015, pp. 2650–2658. 2

[34] A. Kendall, M. Grimes, and R. Cipolla, "Posenet: A convolutional network for real-time 6-dof camera relocalization," in *IEEE Int'l Conf. Computer Vision (ICCV)*. IEEE, 2015, pp. 2938–2946. 2

[35] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proc. National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. 2

[36] J. Baxter, "A Bayesian/Information Theoretic Model of Learning to Learn via Multiple Task Sampling," *Machine Learning*, vol. 28, no. 1, pp. 7–39, 1997. 2

[37] R. Collobert and J. Weston, "A unified architecture for natural language processing: Deep neural networks with multitask learning," in *Int'l Conf. Machine learning (ICML)*, 2008, pp. 160–167. 2

[38] S. Thrun, "Is learning the n-th thing any easier than learning the first?" in *Advances in Neural Information Processing Systems (NIPS)*, 1996, pp. 640–646. 2

[39] J. Kannala and S. S. Brandt, "A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses," *IEEE Trans. Pattern Analysis and Machine Intelligence (T-PAMI)*, vol. 28, no. 8, pp. 1335–1340, 2006. 2

[40] S. K. Nayar and S. Baker, "Catadioptric image formation," in *DARPA Image Understanding Workshop*, 1997. 2

[41] S. Baker and S. K. Nayar, "A Theory of Single-Viewpoint Catadioptric Image Formation," *Int'l J. Computer Vision (IJCV)*, vol. 35, no. 2, pp. 175–196, 1999. 2

[42] Y. Y. Schechner and S. K. Nayar, "Generalized mosaicing," in *IEEE Int'l Conf. Computer Vision (ICCV)*, vol. 1, 2001, pp. 17–24. 2

[43] I. Cinaroglui and Y. Bastanlar, "A direct approach for object detection with catadioptric omnidirectional cameras," *Signal, Image and Video Processing*, vol. 10, no. 2, pp. 413–420, 2016. 2

[44] M. Lourenco, J. P. Barreto, and F. Vasconcelos, "sRD-SIFT: Keypoint Detection and Matching in Images With Radial Distortion," *IEEE Trans. Robotics (T-RO)*, vol. 28, no. 3, pp. 752–760, 2012. 2

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

[46] T. Gandhi and M. Trivedi, "Parametric ego-motion estimation for vehicle surround analysis using an omnidirectional camera," *Machine Vision and Applications*, vol. 16, no. 2, pp. 85–95, 2005. 3

[47] A. Iraqui, Y. Dupuis, R. Boutteau, J. Y. Ertaud, and X. Savatier, "Fusion of Omnidirectional and PTZ Cameras for Face Detection and Tracking," in *Int'l Conf. Emerging Security Technologies*, 2010, pp. 18–23. 3

[48] Y. Tang, Y. Li, T. Bai, X. Zhou, and Z. Li, "Human tracking in thermal catadioptric omnidirectional vision," in *IEEE Int'l Conf. Information and Automation (ICIA)*, 2011, pp. 97–102. 3

[49] A. Furnari, G. M. Farinella, A. R. Bruna, and S. Battiato, "Affine Covariant Features for Fisheye Distortion Local Modeling," *IEEE Trans. Image Processing (T-IP)*, vol. 26, no. 2, pp. 696–710, 2017. 3

[50] X. Ying and Z. Hu, "Can we consider central catadioptric cameras and fisheye cameras within a unified imaging model," in *European Conf. Computer Vision (ECCV)*, 2004, pp. 442–455. 3

[51] M. Born and E. Wolf, *Principles of Optics*, 4th ed. Pergamon Press, 1970. 3

[52] R. Hartley and A. Zisserman, *Multiple View Geometry in comp. Vision*. Cambridge University Press, 2003. 3

[53] R. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 1998, vol. 2. 4

[54] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993. 4

[55] H. V. Hasselt, "Double Q-learning," in *Advances in Neural Information Processing Systems (NIPS)*, 2010, pp. 2613–2621. 4

[56] H. V. Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning." in *AAAI Conf. Artificial Intelligence*, vol. 16, 2016, pp. 2094–2100. 4

[57] R. Sutton, "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming," in *Machine Learning Proc.*, 1990, pp. 216–224. 5

[58] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online]. Available: https://www.tensorflow.org/ 7

[59] Itseez, "Open Source Computer Vision Library," 2015. [Online]. Available: https://github.com/itseez/opencv 7

[60] PointGrey, "Flea3 USB3 Vision cameras for industrial, life science, traffic, and security applications." 2017. [Online]. Available: https://www.ptgrey.com/flea3-usb3-vision-cameras 7

[61] T. S. Cohen, M. Geiger, J. Khler, and M. Welling, "Spherical CNNs," in *International Conference on Learning Representations (ICLR)*, 2018. 9

[62] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning SO(3) Equivariant Representations With Spherical CNNs," *arXiv preprint arXiv:1711.06721v2*, 2018. 9