2017 IEEE International Conference on
Autonomous Robot Systems and Competitions (ICARSC)
April 26-28, Coimbra, Portugal

A Real-Time Deep Learning Pedestrian Detector for Robot Navigation

David Ribeiro, André Mateus, Pedro Miraldo, and Jacinto C. Nascimento
Instituto de Sistemas e Robótica, Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal

***Abstract*— A real-time Deep Learning based method for Pedestrian Detection (PD) is applied to the Human-Aware robot navigation problem. The pedestrian detector combines the Aggregate Channel Features (ACF) detector with a deep Convolutional Neural Network (CNN) in order to obtain fast and accurate performance. Our solution is firstly evaluated using a set of real images taken from onboard and offboard cameras and, then, it is validated in a typical robot navigation environment with pedestrians (two distinct experiments are conducted). The results on both tests show that our pedestrian detector is robust and fast enough to be used on robot navigation applications.**

I. Introduction

In the last few years, robotics has become focused on Human-Robot Interaction and on its role in social environments. Some types of interaction can be, for example: speech; object handover; or a simple navigation behavior, where the robot needs to know if some obstacles are people or not, to decide what is the correct behavior. The study of robot navigation in the presence of people is called Human-Aware Navigation (HAN). For any type of Human-Robot Interaction, one needs to know the position of the people in these environments. Therefore, the Pedestrian Detection (PD) method is one of the most important steps for the robot to interact correctly with the humans.

In this paper, we propose a solution for real-time PD using Computer Vision (onboard and/or offboard cameras) for people state estimation, using a novel deep learning technique. The scheme of our approach is shown in Fig. 1. The solution was firstly tested using offboard and onboard images taken from our testbed and robot platform. Then, to validate our approach, we applied the proposed solution to a HAN problem. The results show that the proposed solution fulfills the respective goals.

The PD task is an important component of our framework, not only in terms of accuracy but also regarding speed, since real-time performance is required. The literature concerning the PD field of study is vast and has been evolving in order to provide improved solutions to this problem (surveys can be found in [1,2]). In general, to perform PD, a detection window is "slided" in several image locations separated by a certain stride, using multiple scales. Features are extracted and classified to determine the presence of a pedestrian. Finally, redundant detections are eliminated using non-maximal-suppression. Initially, the PD methods employed features designed in a handcrafted fashion (e.g., Haar [3], including its informed version [4]). Nevertheless,

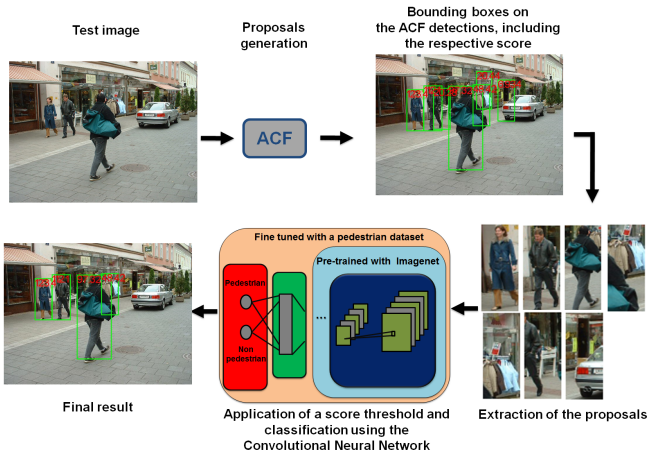


Fig. 1. Scheme of the proposed methodology. The Aggregate Channel Features (ACF) non-deep detector is cascaded with a deep Convolutional Neural Network (CNN). First, the ACF detector generates proposals by providing regions of interest that might contain pedestrians (see green rectangles in the third image). Then, these proposals are classified by the CNN (using the RGB feature map) to improve the accuracy and reduce the number of false positives.

the recent success of Convolutional Neural Networks (i.e. CNNs), achieved in several applications, such as, classification, localization and detection [5], led to the adoption of this methodology to the PD task.

The computations associated with the CNN are expensive when compared to the ones required by methods using handcrafted features. Therefore, to improve the detector's speed, an hybrid solution can be adopted by cascading a faster and shallower method, based on handcrafted features, with a deep CNN. The handcrafted based approach generates proposals (i.e., promising regions for the pedestrians locations), whose classification is refined by the CNN (i.e., the accuracy is enhanced by removing false positives).

Furthermore, the transfer learning technique [6] should be employed when training the CNN in order to prevent overfitting. This technique consists in transferring parameters from a network trained on an auxiliary task (with a large auxiliary dataset, e.g. Imagenet [5]) to initialize our model. Then, our initialized network is fine-tuned (i.e., retrained) for the task of interest (in our case, using the PD dataset).

This paper is organized as follows. Sec. II-A describes the PD methodology, whereas Sec. II-B addresses the tracking procedure. Section III-A describes how the CNN training is accomplished, and Sec. III-B mentions how a pre-trained model is used for fine-tuning. The PD performance is evalu-

ated in the "corridor" and "Mbot" real scenarios (Section IV-A). Sec. IV-B presents the results of the overall and complete framework (PD + HAN). Finally, Sec. V draws conclusions and discusses further work.

II. Vision-Based People Detection and Tracking through Deep Learning

Regarding PD, we use a combination of handcrafted methods with deep learning methodologies. More specifically, first, the non-deep detector Aggregate Channel Features (ACF) [7] provides regions of interest (i.e., proposals), allowing to reduce the expensive computational effort that would be required by a CNN, in the exhaustive process of sliding window search. Then, the proposals obtained previously are classified by the CNN, allowing to improve the accuracy of the ACF detections (as depicted in Fig. 1)[1].

A. Methodology for PD

In the following, we describe the methodology and introduce the notation for the PD task. Given a training set $\mathscr{D} = \{(\mathbf{x},\mathbf{y})_i\}_{i=1}^{|\mathscr{D}|}$, $\mathbf{x}$ represents the input image with $\mathbf{x}:\Omega\rightarrow\mathbb{R}^3$ and Ω representing the image lattice[2] of size $w\times h\times d$, with $d=3$; the class label is defined in $\mathbf{y}\in\mathscr{Y}=\{0,1\}^C$ that denotes the (absence) presence of the pedestrian in the ith image $\mathbf{x}_i$ (i.e., $C=2$). A detector (e.g., ACF or LDCF [8]) is applied to each input image $\mathbf{x}_i$ in order to generate proposals (including the associated confidence scores). This results in an output set denoted by $\mathscr{O} = \{(\mathbf{x}(\mathscr{B}),\mathscr{S})_i\}_{i=1}^{|\mathscr{O}|}$, where $\mathscr{B} = \{b_k\}_{k=1}^{|\mathscr{B}|}$ denotes the set of bounding boxes coordinates, with $b_k = [x_k, y_k, w_k, h_k]\in\mathbb{R}^4$ representing the top-left point and width and height enclosing (or not) the pedestrian. The content (i.e., the proposals) of the image delimited by the bounding boxes $\mathscr{B}$ is represented by $\mathbf{x}(\mathscr{B})$, and $\mathscr{S} = \{s_k\}_{k=1}^{|\mathscr{S}|}$ correspond to the ACF detector confidence scores assigned to the proposals $\mathbf{x}(\mathscr{B})$.

The use of pre-trained models, during the CNN initialization process, allows to obtain gains concerning the generalization ability of the model [6]. Hence, we select the VGG CNN model [9], pre-trained with Imagenet [5]. We denote the dataset to pre-train the CNN as $\widetilde{\mathscr{D}} = \{(\widetilde{\mathbf{x}},\widetilde{\mathbf{y}})_n\}_{n=1}^{|\widetilde{\mathscr{D}}|}$, with $\widetilde{\mathbf{x}}:\Omega\rightarrow\mathbb{R}^3$ and $\widetilde{\mathbf{y}}\in\widetilde{\mathscr{Y}}=\{0,1\}^{\widetilde{C}}$, where $\widetilde{C}$ is the number of classes in the pre-trained model (for the Imagenet, we have $\widetilde{C}=1000$).

1) CNN model: Typically, the structure of CNNs includes the composition of: convolutional layers with a non-linear activation function; non-linear subsampling layers; fully connected layers; and a multinomial logistic regression layer [10]. Formally, the CNN can be denoted by:

$$f(\mathbf{v},\theta^{(1)}) = \mathbf{v}^\star = f_{\text{out}}\circ f_L\circ\ldots\circ f_2\circ f_1(\mathbf{v}^{(0)}), \qquad (1)$$

where $\mathbf{v}^{(0)}=\mathbf{v}$ is the input data, $\circ$ denotes the composition operator, $\theta^{(1)}$ represents the CNN parameters (i.e., the weights and biases), and $\mathbf{v}^\star$ is the CNN output (prediction). For the PD case, $\mathbf{v}^{(0)}=\mathbf{v}=\mathbf{x}(\mathscr{B}^{(0)})=\mathbf{x}(\mathscr{B})$ (see 4th image in Fig. 1), which represents the proposals, and $\mathbf{v}^\star = f(\mathbf{v},\theta^{(1)}) = \mathbf{y}^\star = f(\mathbf{x}(\mathscr{B}),\theta^{(1)})$, which denotes the prediction. The CNN is applied to these proposals, outputting the probability of the existence of a pedestrian in each one of them. If a proposal is classified as pedestrian, it is saved and no changes are made to its original ACF score. The proposals considered to be non pedestrians are eliminated, in order to reduce the number of false positives.

The convolution of a layer's input with a set of filters, followed by a non-linearity, is represented by:

$$\mathbf{v}^{(k)} = f_k(\mathbf{v}^{(k-1)}) = \sigma(\mathbf{W}_k(i,j)^\top\mathbf{v}^{(k-1)}+\beta_k), \qquad (2)$$

where the convolutional filters are represented by the weight matrix $\mathbf{W}_k$ and the bias vector β_k, and where $\sigma(.)$ represents the non-linearity (e.g. the Rectified Linear Unit [10]). The non-linear subsampling layers are denoted by $\mathbf{v}^{(k)}=\downarrow\mathbf{v}^{(k-1)}$, where $\downarrow$ represents the function (e.g., mean or max) applied to the input regions, leading to the size reduction. The fully connected layers employ a special case of the convolution represented in (2), because the entire input is convolved with individual filters. The multinomial logistic regression layer uses the soft-max function: $\mathbf{y}(i) = \frac{\exp(\mathbf{v}^L(i))}{\sum_j\exp(\mathbf{v}^L(j))}$ to calculate the probability for each class (indexed by i), using the input $\mathbf{v}^L$ from the L^{th} layer.

The loss function used during training is the binary cross entropy loss expressed as:

$$\mathscr{L} = \frac{1}{|\mathscr{D}|}\sum_{i=1}^{|\mathscr{D}|} -y(i)\times\log(y^\star(i)) - (1-y(i))\times\log(1-y^\star(i)) \qquad (3)$$

where the training set $\mathscr{D}$ is indexed by i.

We denote a pre-trained CNN model as: $\widetilde{\mathbf{y}} = f(\widetilde{\mathbf{x}},\widetilde{\theta})$, with $\widetilde{\theta} = [\widetilde{\theta}_{\text{cn}},\widetilde{\theta}_{\text{fc}},\widetilde{\theta}_{\text{lr}}]$, where $\widetilde{\theta}_{\text{cn}}$ are the parameters for the convolutional and non-linear subsampling layers, $\widetilde{\theta}_{\text{fc}}$ are the parameters for the fully connected layers, and $\widetilde{\theta}_{\text{lr}}$ are the parameters for the multinomial logistic regression layer.

The parameters $\widetilde{\theta}_{\text{cn}}$ and $\widetilde{\theta}_{\text{fc}}$ (or a subset of them) can be transferred to another CNN model, in order to provide a rich initialization [6]. The layers, whose parameters were not transferred, can be randomly initialized. Finally, the resulting CNN is fine-tuned with the dataset corresponding to the task of interest.

In the PD case, we transfer the parameters from the convolutional and non-linear subsampling layers, and randomly initialize all the other layers. Due to changes in the CNN input size, the parameters for the fully connected layers were adjusted accordingly. The multinomial logistic regression layer was adapted to consider only two classes (pedestrian and non-pedestrian). Finally, this CNN model for PD was fine-tuned with the pedestrian dataset $\mathscr{D}$, resorting to the binary cross entropy loss in (3).

B. Tracking

Taking the position measurements from the people detection scheme (described in the previous subsection), the goal of the tracking phase is to associate detections between

[1]Other non-deep detectors could be used, but we adopted the ACF detector because it is fast.

[2]In this paper, the RGB feature map is considered for the image $\mathbf{x}$.

frames and to estimate the direction of a person's velocity. For that purpose, we use a simple Kalman filter[3]. Moreover, since humans tend to walk at a constant velocity [11], a constant velocity model was assumed. For each iteration, the PD detections are associated with the current tracks. This is performed using either the Nearest Neighbor or the Nearest Neighbor Joint Probabilistic Data Association methods. The associated detections are then used in the update phase of the respective track. If the detection is not associated with any track, it is stored, and if it is stable for some time, a new track is started. Finally, if a track has no detection associated with it for a while, it is deleted. For more details on the tracker and on the association methods used, please refer to [12] and [13], respectively.

III. Material and methods

In this section, we describe the training details used for the pedestrian detector, taking into account the overall framework for the navigation setup.

A. Dataset for the CNN training

The pedestrian dataset chosen to train the CNN was the INRIA dataset [14], which is a popular benchmark in the PD field of study[4]. Originally, this dataset is divided in train (1832 images) and test (288 images) sets. Within the train set, there are 1218 negative images (i.e., without pedestrians), and 614 positive images (i.e., with pedestrians).

For the process of training the CNN, we extracted proposals from the original train set as follows. To construct the positive set, first, we use the ground truth positive training bounding boxes to extract proposals, resulting in $\mathscr{B}_{\text{pos}} = 1237$ samples. Then, we augment (i.e., with data augmentation) this set using two steps:

1) Horizontal flipping applied to $\mathscr{B}_{\text{pos}}$, resulting in $\mathscr{B}_{\text{pos}}^{(1)} = 2474$ (including also $\mathscr{B}_{\text{pos}}$); and
2) Random deformations (by affecting pixels in the range $\mathscr{R} = [0,5]$ for the beginning and end) performed in the previous set $\mathscr{B}_{\text{pos}}^{(1)}$, resulting in $\mathscr{B}_{\text{pos}}^{(2)} = 4948$.

To construct the negative set $\mathscr{B}_{\text{neg}}$, we use the methodology from [15] (i.e., applying a non-fully trained LDCF detector) to extract proposals from the negative images. This results in $\mathscr{B}_{\text{neg}} = 12552$ negative proposals. The final set of CNN train proposals comprises a total of 17500 samples, which are divided in train (15751 proposals, i.e. 90% of the total) and validation (1749 proposals, i.e. 10% of the total).

B. CNN model for training

Since we use a pre-trained CNN model (which can be considered a regularization technique), we have to adapt it to our task of interest (i.e., PD). Next, we mention the selected pre-trained CNN, the changes made, and the final architecture training details.

[3]Other filters could be used, but we used the Kalman filter because of its simplicity–this is not the main focus of the paper.

[4]More details can be found at: `http://pascal.inrialpes.fr/data/human/`.

1) Pre-trained model original architecture: The VGG Very Deep 16 architecture (VGG-VD16) (configuration D) [9][5] was chosen to be the pre-trained architecture. This model's original input size is $224 \times 224 \times 3$, and has 13 convolutional layers (with a 3×3 window), the Rectified Linear Unit non-linearity, five max-pooling operations (with a 2×2 window with 2-times reduction), three fully connected layers and a multinomial logistic regression layer (see Sec. II-A.1). The classification results from the output of the last layer, which has 1000 filters corresponding to each one of the classes in ILSVRC [5]. The dataset used to perform the pre-training of this model is Imagenet [5], containing 1K visual classes, 1.2M training, 50K validation and 100K test images.

2) Pre-trained model changes and fine-tuning: Motivated by the pre-trained model's expensive and time consuming computations, the original dimensions of the CNN input were downscaled from $224 \times 224 \times 3$ to $64 \times 64 \times 3$. With this modification, inference cannot be performed after the first fully connected layer. Furthermore, the classification related layer must be adapted to transition from 1000 ILSVRC classes to two PD classes (i.e., pedestrian and non-pedestrian). To overcome this problems, we randomly initialize the parameters of the three fully connected layers with the correct dimensions. For this initialization procedure, we selected a Gaussian distribution, with mean $\mu = 0$ and variance $\sigma^2 = 0.01$. The modified CNN model is fine-tuned with the positive and negative proposals training sets, acquired from the INRIA dataset (as described in Sec. III-A).

In terms of the fine-tuning hyperparameters, we used 10 epochs with a minibatch of 100 samples, a learning rate of 0.001, and a momentum of 0.9.

For the test, first, the proposals (i.e., promising regions for the existence of pedestrians) are extracted by running the ACF detector in the test images. Then, these proposals are classified as pedestrians or non-pedestrians, by the fine-tuned CNN model described previously.

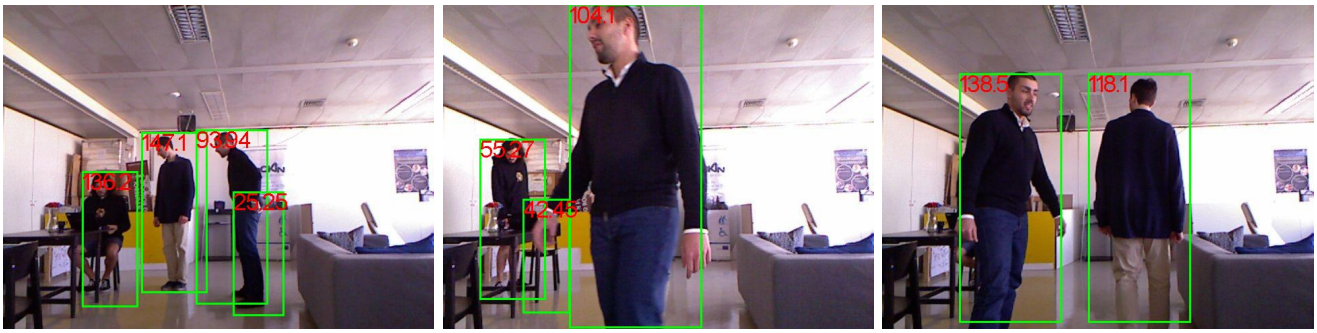
3) Implementation: The PD methodology was implemented in MATLAB, running on CPU mode on 2.50 GHz Intel Core i7-4710 HQ with 12 GB of RAM and 64 bit architecture. To run the ACF detector and to evaluate the performance, the Piotr's Computer Vision MATLAB Toolbox [16] (2014, version 3.40) was employed. Concerning the CNN framework, we utilized the MatConvNet toolbox [17]. The experiments described in the next section, namely: Sec. IV, and Table I), were conducted using the same settings.

IV. Experimental Results

In this section, we evaluate the PD method in two real scenarios (i.e., datasets named "corridor" and "Mbot"). To conclude, we validate our approach by using the PD method on a Human-Aware Navigation problem, in which a real-time detection performance is required.

[5]Additional details can be found at: `http://www.robots.ox.ac.uk/~vgg/research/very_deep/`.

(a) Pedestrian detection in the "corridor" sequence.



(b) Pedestrian detection in the "Mbot" sequence.

Fig. 2. To test our PD methodology, we used two sequences of images acquired from possible real scenarios camera locations, which can be the ceiling and on the robot. In Figs. (a) and (b), three images of the "corridor" and "Mbot'" datasets are depicted with the obtained detections (in green) and the corresponding scores (in red), respectively.

A. *Evaluation of the PD method*

In order to conduct experiments in real scenarios, we acquired two indoor datasets and tested the proposed PD method on them. These datasets are: 1) the "corridor" dataset, which comprises 5556 images, and 2) the "Mbot" dataset, which comprises 3966 images. For both sets, the image (i.e., frame) dimensions are 480×640. The detection results for some samples of these datasets are depicted in Fig. 2.

For each dataset, we measure the runtime of the final PD method (i.e., ACF+CNN) proposed in Sec. III-B. As a result, we obtain approximately 707.27 seconds, which is equivalent to 7.85 FPS, for the "corridor" dataset (5556 frames), and 839 seconds, which is equivalent to 4.84 FPS, for the "Mbot" dataset (3966 frames). Further details about the runtime figures are presented in Tab. I (top, the two columns in the field named "Baseline"), where the values represent per frame metrics.

To reach the real-time specifications required in robot navigation tasks, the speed should be improved. This can be accomplished by filtering the ACF proposals based on the confidence score, since this procedure reduces the number of proposals to be processed by the CNN, increasing the detection speed. The goal is to improve the previous speed, and achieve real-time performance, without substantially degrading the accuracy.

Taking into account that the confidence scores are important indicators to determine the relevance of each proposal, a score rejection threshold can be established. Only the proposals with score above this threshold are classified by the CNN. Following [18], we selected a threshold value of 40.

Consequently, in the process of discarding false positives, first we should eliminate the easier ones resorting to the threshold operation, and then we should eliminate the harder ones using the CNN. The possible loss in accuracy versus the speed improvement is determined by the choice of the threshold value.

Accordingly, using the threshold operation strategy, the detection speed of the overall method (i.e., ACF+CNN) is improved, in comparison with the baseline metrics. The cases before ("Baseline" field) and after ("Threshold" field) the threshold operation are depicted in Tab. I. As presented in Tab. I, we are able to achieve the real-time requirements for our navigation setup, by reaching a detection speed of approximately 10 FPS.

B. *Real Experiments in an Indoor Scenario*

In this section we evaluate our PD framework on a Human-Aware Navigation (HAN) application. For that purpose, we consider a setup as shown in Fig. 3:

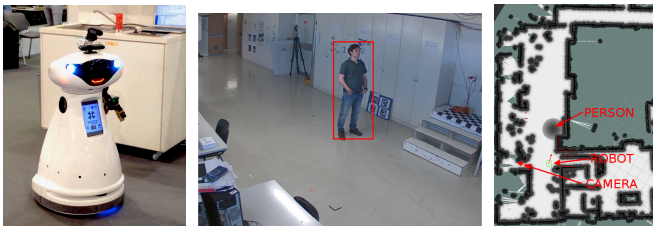
- We use a MBOT mobile platform [19] (see Fig. 3(a)) in a typical domestic indoor scenario; and
- A perspective camera mounted on the ceiling (an example of an image acquired using this camera is shown in Fig. 3(b)) was placed in the environment as shown in Fig. 3(c).

The HAN is not the focus of the paper. Then, we follow the navigation (including the constraints associated with the HAN) proposed at [20]. Basically, the authors use the A^* as a

TABLE I

RUNTIME FIGURES BEFORE (TOP, "BASELINE") AND AFTER (BOTTOM, "THRESHOLD") THE THRESHOLD OPERATION APPLIED TO THE ACF PROPOSALS, WHEN USING THE OVERALL PD METHOD (I.E., ACF+CNN).

Dataset	Data seq. 1 (corridor)	Data seq. 2 (Mbot)
Baseline	Total time = 0.1273 sec. ACF time = 0.0326 sec. CNN time = 0.0947 sec. **Frame rate = 7.85** FPS	Total time = 0.2066 sec. ACF time = 0.0367 sec. CNN time = 0.17 sec. **Frame rate = 4.84** FPS
Threshold	Total time = 0.0961 sec. ACF time = 0.0333 sec. CNN time = 0.0628 sec. **Frame rate = 10.41** FPS	Total time = 0.1026 sec. ACF time = 0.0381 sec. CNN time = 0.0645 sec. **Frame rate = 9.74** FPS



(a) (b) (c)

Fig. 3. Representation of the setup used in the experiments. Fig. (a) shows the robot platform and Fig. (b) shows an image of the camera that will be used to estimate the pedestrians (as it can be seen, in this image we already show the bounding box identifying a person in the environment). To conclude, Fig. (c) shows the environment (ROS rviz package), with the position of all the cameras, the position of the robot, and the pedestrian, with the respective HAN constraint (in this case the pedestrian was standing).

path planner (to ensure a minimal cost path) and define a set of HAN constraints as cost functions. These cost functions are shown in the figures of the experimental results (Figs. 4 (a) and (b)), and are computed using the following procedure:

1) Selecting the middle point of the lower edge of the bounding box that is given by the PD;
2) Projecting this middle point on the image onto the floor plane (assuming that the position of the robot is known); and
3) Estimating the pedestrian velocity in the world coordinate system.

The goal of these experiments is to evaluate the proposed PD on the image, using a robot navigation application in the presence of people. Two experiments were conducted, in which we apply our PD and the previous method for HAN:

1) Firstly, we consider a simple example where a robot is going towards a goal and people are standing in the environment (in front of the robot). The robot must take their positions into account (which are given by the PD mapped onto the floor plane) on the path planning, in order to avoid a collision; and
2) In the second experiment, a person starts walking when the robot is moving, blocking the path of the robot. Following the social rules, the robot must replan its path, to overtake the person by the left.

The results of both experiments are shown in Figs. 4(a) and 4(b), respectively. Videos with these experiments will be included in the authors websites. As it can be seen by these figures, the robot behaves as expected, which proves that our PD method is suitable for robot navigation tasks, in the presence of people.

V. CONCLUSIONS

This paper presents a novel framework that integrates pedestrian detection in the problem of robot navigation. More specifically, it integrates a novel pedestrian detection approach jointly with specific motion constraints, representing the human-aware concerns. The novelty inherent to the PD methodology, is that it allows to improve the accuracy of a non-deep detector, by efficiently cascading a CNN. The PD method is evaluated using two sets of real images acquired on a typical robot navigation environment (considering both on-board and external camera sensors). The results show that the proposed solution is suitable for robot navigation tasks, namely in terms of both runtime and robustness. In addition, two experiments were conducted in a realistic scenario to assess the overall framework performance.

As future work, we are planning on fusing data from multiple external and on-board cameras, and other types of sensors, such as lasers, as well as test the proposed framework with different people behaviors.

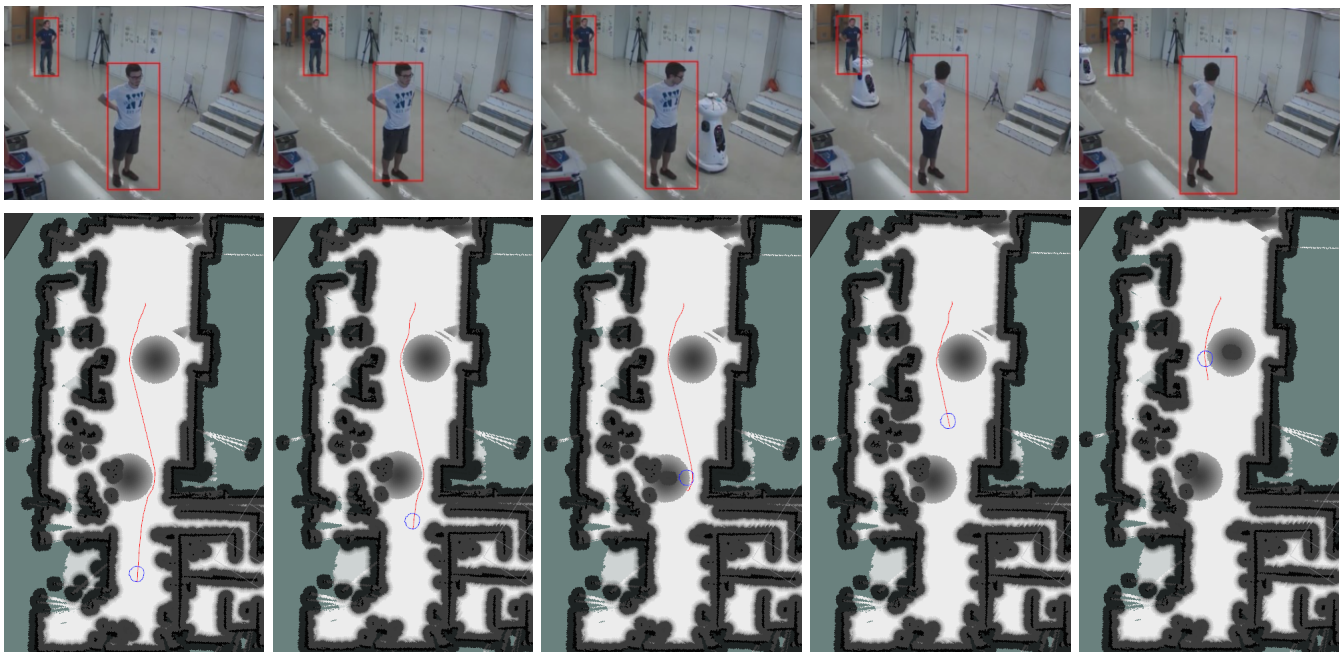
ACKNOWLEDGEMENTS

This work was partially supported by FCT[UID/EEA/50009/2013], and by the FCT grant SFRH/BPD/111495/2015.

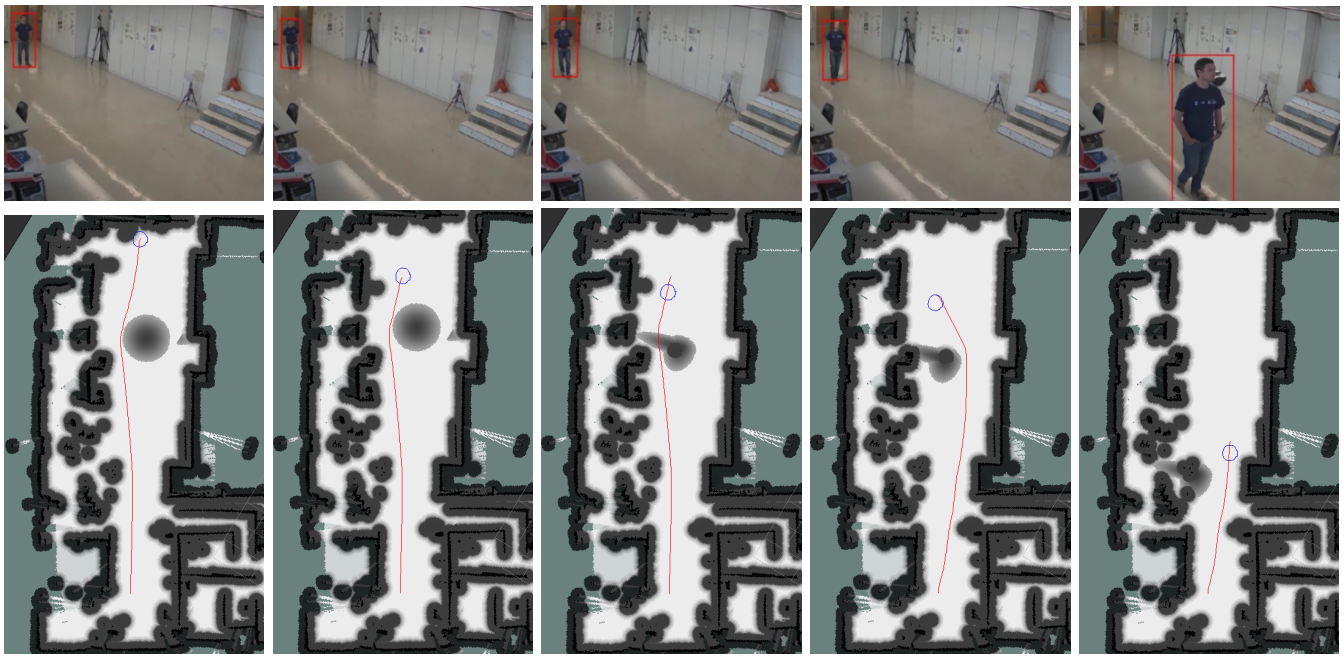
We would also like to thank to Luis Luz for his help getting the experimental results.

REFERENCES

[1] P. Dollar, C. Wojec, B. Schiele, and P. Perona, "Pedestrian detection: An evaluation of the state of the art," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2012.
[2] R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Ten years of pedestrian detection, what have we learned?" in *ECCV, CVRSUAD workshop*, 2014.
[3] P. Viola and M. J. Jones, "Robust real-time face detection," *IJCV*, vol. 57, no. 2, pp. 137–154, 2004.
[4] S. Zhang, C. Bauckhage, and A. Cremers, "Informed Haar-Like Features Improve Pedestrian Detection," in *CVPR*, 2014.
[5] 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*, 2015.
[6] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Neural Information Processing Systems (NIPS)*, 2014.
[7] P. Dollar, R. Appel, S. Belongie, and P. Perona, "Fast feature pyramids for object detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, 2014.
[8] W. Nam, P.Dollar, and J. H. Han, "Local decorrelation for improved pedestrian detection," in *Neural Information Processing Systems (NIPS)*, 2014.
[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Int'l Conf. on Learning Representations (ICLR)*, 2015.
[10] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Neural Information Processing Systems (NIPS)*, 2012.



(a) Experiment 1, where people are standing during all the robot's motion.



(b) Experiment 2, where people are standing in the beginning but, when the robot starts moving, one person will start walking in the robot's path.

Fig. 4. Results of the real experiments in realistic scenarios. A MBOT mobile robot [19] is used on a typical domestic indoor scenario, developed for benchmarking in an ERL@Home testbed. In these experiments, the robot is navigating while a person (which firstly was standing) starts walking, blocking the robot's path. At that moment, the robot must overtake the pedestrian according to the respective social rule, that says that a robot must overtake a person through the left. The robot is shown as a blue circle while the path of the robot is shown as a red line.

[11] S. Bitgood and S. Dukes, "Not Another Step! Economy of Movement and Pedestrian Choice Point Behavior in Shopping Malls," *Environment and Behavior*, 2006.
[12] N. Bellotto and H. Hu, "Computationally efficient solutions for tracking people with a mobile robot: an experimental evaluation of bayesian filters," *Autonomous Robots*, 2010.
[13] Y. Bar Shalom, F. Daum, and J. Huang, "The Probabilistic Data Association Filter," *IEEE Control Systems*, 2009.
[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Proc. Computer Vision and Pattern Recognition (CVPR)*, 2005.
[15] D. Ribeiro, J. C. Nascimento, A. Bernardino, and G. Carneiro, "Improving the performance of pedestrian detectors using convolutional learning," *Pattern Recognition*, 2016.
[16] P. Dollár, "Piotr's Computer Vision Matlab Toolbox (PMT)," http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html.
[17] A. Vedaldi and K. Lenc, "MatConvNet – Convolutional Neural Networks for MATLAB," *ACM Proc. Int'l Conf. on Multimedia*, 2015.
[18] A. Verma, R. Hebbalaguppe, L. Vig, S. Kumar, and E. Hassan, "Pedestrian detection via mixture of cnn experts and thresholded aggregated channel features," in *ICCV Workshop*, 2015.
[19] J. Messias, R. Ventura, P. Lima, J. Sequeira, P. Alvito, C. Marques, and P. Carriço, "A Robotic Platform for Edutainment Activities in a Pediatric Hospital," in *IEEE Int'l Conf. Autonomous Robot Systems and Competitions (ICARSC)*, 2014.
[20] A. Mateus, P. Miraldo, P. U. Lima, and J. Sequeira, "Human-Aware Navigation using External Omnidirectional Cameras," *Iberian Robotics Conference*, 2015.