Person Following Robot Behavior Using Deep Learning

Ignacio Condés[1] and José María Cañas[2]

[1] Universidad Rey Juan Carlos, ignacio.condes.m@gmail.com,
[2] Universidad Rey Juan Carlos, jmplaza@gsyc.urjc.es

Abstract. Human-robot interaction (HRI) is a field with growing impact as robot applications are entering into homes, supermarkets and general human environments. Person following is an interesting capability in HRI. This paper presents a new system for a robust person following behavior inside a robot. Its perception module addresses the person detection on images using a pretrained TensorFlow SSD *Convolutional Neural Network* which provides *robustness* even on tough lighting conditions. It also includes a face detector and a *FaceNet* CNN to reidentify the target person. Care has been put to allow real-time operation. The control module implements two *PID* controllers for a reactive smooth response, moving the robot towards the target person without distracting with other people around. The entire system has been experimentally validated on a real TurtleBot2 robot, with an Asus Xtion RGBD camera.

Keywords: Computer Vision, neural networks, deep learning, robotic behavior

1 Introduction

In the last years robots have become a powerful tool for humans in many areas, helping to perform all kind of tasks: hazardous explorations, personal assistance, cleaning, driving, etc.. Electronics assembly, car factories, autonomous driving (like Tesla), vaccum cleaners (like Roomba) and even logistics (like Amazon warehouses) are good examples of it. In all these fields robots are designed to perform these tasks on the most autonomous possible way, which means not requiring to be directly controlled on every action. This involves to provide robots a certain intelligence and capabilities to correctly trigger the most suitable action for each possible input stimulus.

As robots enter into offices, supermarkets or homes (like vacuum cleaners, domestic assistants, etc.), better mechanisms for human-robot interaction are also desirable. One useful capability there is the robot being able to follow a particular person.

In addition, one of the main robot sensors are cameras. They are cheap and a very powerful source of information about the robot environment. One of the most popular advances in computer vision are the deep learning techniques, using neural networks to process camera images. They can solve classification,

detection and segmentation tasks in a very robust way. Typically, neural networks have to be trained on huge datasets and they work fine with new unseen images.

In this paper, a new system for *person following behavior* inside a robot is presented. It avoids the use of laser sensors and relies on a cheap RGBD sensor and color images processing. The person *detection* subsystem uses deep learning to detect people and to identify the particular person the robot has to follow, in particular *Convolutional Neural Networks* (CNNs).

Second section reviews some related works about person following and image processing with neural networks. Third section describes the design of the proposed system, whose main modules are detailed in the fourth and fifth sections, explaining its perception side and its control side. It has been implemented and experimentally validated, as it will be seen on section 6. Finally some brief conclusions finish the paper.

2 Related works

The person following behavior is a old problem in Human-Robot-Interaction field with many existing solutions in the literature [18, 4, 16].

A nice approach [3] uses laser sensors to detect the legs, sensor fusion and a particle filter to provide robustness and continuity to the person estimation.

The topic has been extensively explored using a single camera [16]. Other interesting approaches use stereo vision. [2] combines color of the clothes and position information to detect and track multiple people around. They use Kalman filters and fuse the plain-view map information with a face detector. [17] use depth templates of person shape applied to a dense depth image and an SVM-based verifier for eliminating false positives. They use the Extended Kalman filter to continuously estimate the person positions. In addition, Histogram of Oriented Gradients are a classical and effective method for human detection [1], which is a relevant ingredient of the person following behavior.

In the last years, the availability of low cost RGBD sensors has opened the door to new approaches using them [15, 14], taking advantage of the depth information. The depth simplifies not only the robustness of the detection itself but also the control of the robot movements, which can be the developed as position based controllers. In [9] deep learning techniques on color and motion cues have been used to sucessfully detect people on RGBD videos for surveillance applications.

The re-identification of a particular person is a complementary problem to general people detection. There are image-based methods and video-based methods. Recent papers also use deep learning to identify the detected persons [13]. Other methods use features and skeleton keypoints [11]. [10] use image and range data combining color, height and gait features (step length and speed) for a robust identification, even in severe illumination environments.

[12] is an interesting work that uses neural networks to both multiple person detection and a particular person re-identification. His system works in real-time on moderate hardware using a single camera.

3 Design

The proposed system follows the structure represented in the Fig. 1 and it is composed of two modules. The *perception* module is responsible for performing the *vision tasks*. Therefore, it determines, with the maximum possible accuracy, the position of every person found inside the color image. It also discerns which of these persons is the one to be followed, in case she is seen. The *actuation* module reads that output from the perception module and sends suitable movement commands to the robot.

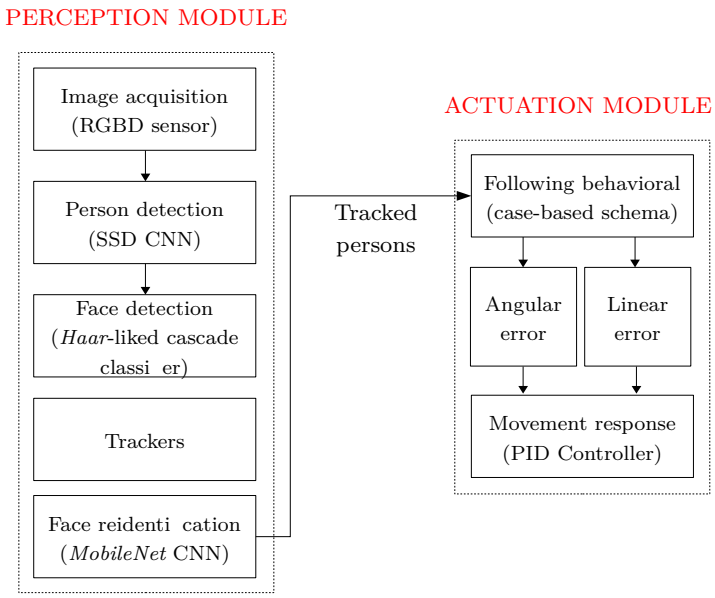

Fig. 1. General design of the proposed system.

These mentioned modules run on a computer (the specifications of our settings can be found at Section 6), connected to both the RGBD sensor driver and the robot driver. They have been implemented in Python language as concurrent functions are executed inside a single application. Several threads iterate on independent tasks at an individually defined frequency. In the case of a slow

computer the frequency can be lowered down, and so, the system is designed to run properly on different hardware specifications of the base PC.

4 Perception Module

This module is responsible for apprehending the incoming images (RGB and depth) from the sensor, and generates a significant output to be interpreted by the actuation module. Its structure follows a certain pipeline with five steps (Figure 1): image acquisition, person detection, face detection, person and face trackers, and face reidentification. This pipeline allows the system to discern if the target person is being seen right now.

4.1 Person Detection

First, the incoming RGB images are passed through an *object detection* module. It is powered by a pretrained CNN, which has a special architecture designed in [5] for this purpose: SSD (*Single-Shot Multibox Detector*). This detection model type stands out by its prediction speed, because *it performs a single feed-forward pass* of the image through the network, unlike other state-of-the-art techniques. These other approaches perform successive feed-forward passes through the network, what makes them consequently much slower. Before choosing SSD for the proposed system, several detection architectures were compared and their corresponding typical inference times were measured (Table 1).

Architecture	**Base network**	**Dataset**	**Mean inference time** (ms)
ResNet	Inception	COCO	820.71
SSD	MobileNet	COCO	107.43
ResNet	101	COCO	786.49
ResNet	50	COCO	515.28
ResNet	101	COCO	63.97
Faster-RCNN	ImageNet	ILSVRC2014	703.99
Faster-RCNN	Inception	COCO	352.20
ResNet	50	COCO	793.87
SSD	MobileNet	COCO	102.85
ResNet	101	COCO	898.59
Inception	ResNet	OID	792.42
SSD Lite	**MobileNet**	**COCO**	**68.13**
ResNet	101	Kitti	111.29
Inception	ResNet	OID	667.76

Table 1. Timing performance tests for several detection models. The selected implementation is SSD Lite.

On this module, the desired image to input has to be reshaped to 300×300 px, which is a typical shape for this type of CNN architecture. As shown on Fig.

2, it extracts a set of activation maps on its *base network*: the feature extractor (*MobileNet*[19] is used in this particular model), and finally position and class inferences are made later, on multiple image scales. Finally, a *Non Maximum Suppressor* is applied, retaining only the most confident detections, which are adjusted to a correct bounding box shape.

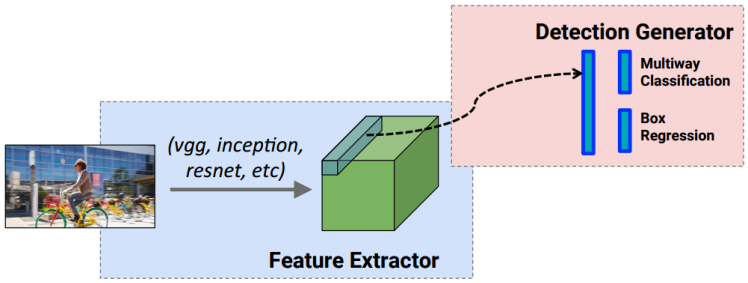

Fig. 2. Schema of the SSD architecture.

This network provides an accurate and efficient object detection, returning for each processed image:

- *Classes:* the detected classes (person, cell phone, airplane, dog ...) inferred for each detected object.
- *Scores:* the confidence $\in [0, 1]$ the network has on each object belonging to the estimated class.
- *Boxes:* the coordinates of the rectangular *bounding box* which wraps the detected object, expressed as the coordinates of two opposite corners of it.

Although the used SSD model is capable to detect up to 80 object classes, the proposed system only retains those detection corresponding to *persons*, as it is what we are interested to follow. This whole process provides a lightweight *person detection*, perfectly capable to work in real time on a regular computer.

In addition, our implementation is capable to handle different network models and architectures on a *plug and play* way, just pointing the model file (in the specific TensorFlow `.pb` format) in the configuration file of the program.

4.2 Face detection

Once the existing persons on the image have been detected, the next step is to search their faces, in order to know whether any of them corresponds to the target person to follow. In order to achieve it, the classical Viola and Jones face detection algorithm [6] has been used, as it is a simple and fast solution, suitable for a prototype that already handles a *SSD* CNN. This algorithm, which comprises *Haar* features (Fig. 3), is a simple algebraic method which takes advantage of the typical illumination pattern of a face (due to its physical shape)

to detect promising regions of the input image to contain a face. To test a *Haar* feature on a specific grayscale image, the feature is slided through it, and a simple operation is performed (black pixels subtracted to white ones). If the result is positive along all the feature, the region where it has been applied passes the test.

As the previous block detected persons inside the image, this face detection algorithm is only applied inside the instance of the detected persons, in order to speed up the system and avoid false positive detections. Each one of the detected persons is divided into *regions*, which are passed through a *cascade* of tests, where the non-compliant regions are immediately discarded. The accepted ones pass to a slightly more complex feature each time, and the regions which pass all the features are supposed to contain a face. This progressive region dismiss makes it an efficient algorithm, capable of run simultaneously with the rest of tasks. This entire process is performed with an OpenCV[3] function.

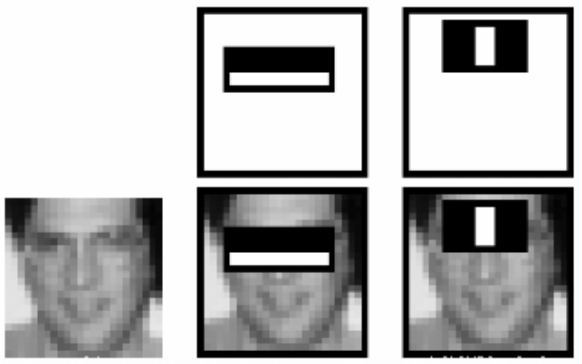

Fig. 3. *Haar* features applied on a face.

4.3 Person and face trackers

Both previous steps detect persons and faces, but although they behave in a robust way, their outputs can suffer spurious false negative or false positive detections, due to lighting or occlusions. As they can interfere with the desired following capability, the third step palliates their effect implementing time-spatial *trackers*, which filter the detection outputs (*candidate detections*). The tracker associates to each person/face its coordinates inside the image, and takes into account the number of successive frames on which it has been detected. If it surpasses a *patience* threshold, then it is considered a reliable (*tracked*) detection. Otherwise, it is taken as a spurious output and ignored. In addition, each *tracked detection* is remembered for some frames even without new updates of

[3] Open source image processing library.

its current position. This way, a partial occlusion of a person does not cause her elimination, until she has not been seen for a while.

This tracking step can be observed in Fig. 4, in a generic *instance* case (which comprises the same behavioral for person and face tracking). In addition, it is not necessary to detect the person/face every time, as the tracker is capable to infer that a new detection near the last position of a person corresponds to its new location, without requiring to see again her face to check whether she is the same person.

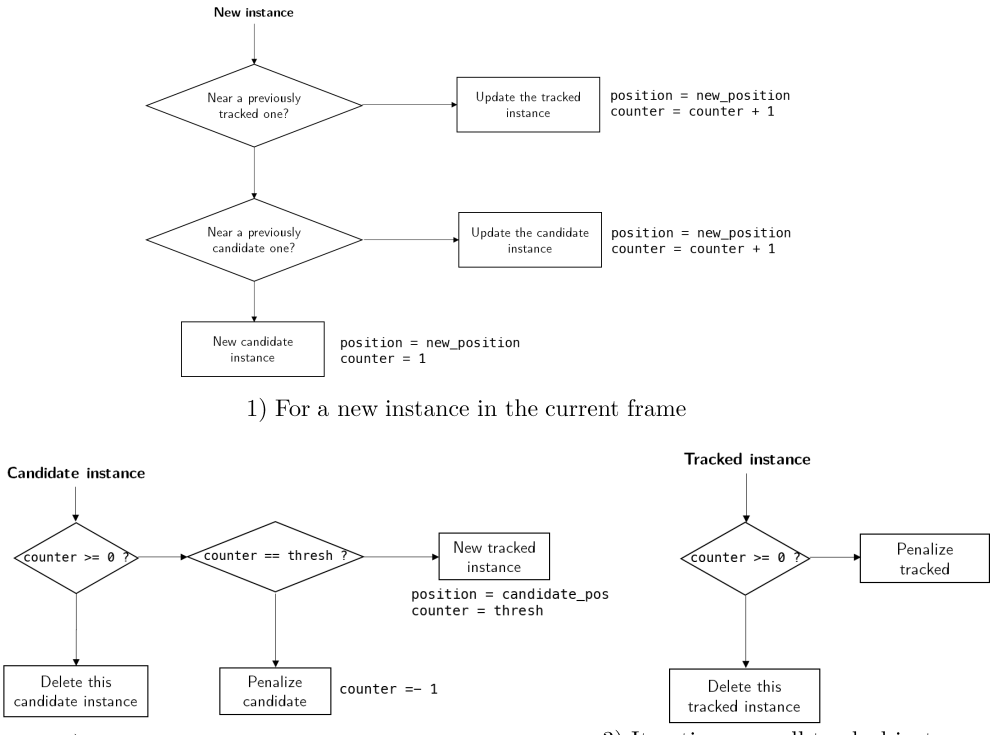

Fig. 4. Tracking process for a instance (person/face).

4.4 Face reidentification

The previous trackers turn the unfiltered output of the detection subsystems into tracked and more reliable detections. Hence, they can be used to perform *identification* tasks. For this purpose, a parallel CNN is used, the *FaceNet* [7]. This network *maps* a face image, extracting some key features, into a 128-dimensional

Euclidean space, where faces are represented by what is called *embeddings* (feature vectors). The Euclidean L^2 distance (Eq. 1) existing between two of this embeddings stands for the *face similarity* between those faces. Hence, we can consider that two embeddings belong to the same face if their distance is below a threshold, which has experimentally been set to 1.1.

$$d(\mathbf{f_1}, \mathbf{f_2}) = \sqrt{\sum_{i=1}^{128} (f_{1_i} - f_{2_i})^2} \tag{1}$$

The proposed solution makes use of this, and computes the embeddings of each detected (and tracked) face on real time. Once this has been performed, it compares the similarity between these embeddings and the one corresponding to the target person. This target embedding is computed when the program starts, from a given image file of the *person to be followed*.

To avoid penalties on similarity due to lighting conditions, a previous blurring and *prewhitening* (on Eq. 2, with x as the color channel, μ as its mean and σ as its standard deviation) are performed on each given face to the *FaceNet*.

$$x' = \frac{x - \mu}{\sigma} \tag{2}$$

In Figure 5 the same face is seen in different lighting conditions, and all of them yield embeddings with a distance lower than the threshold.

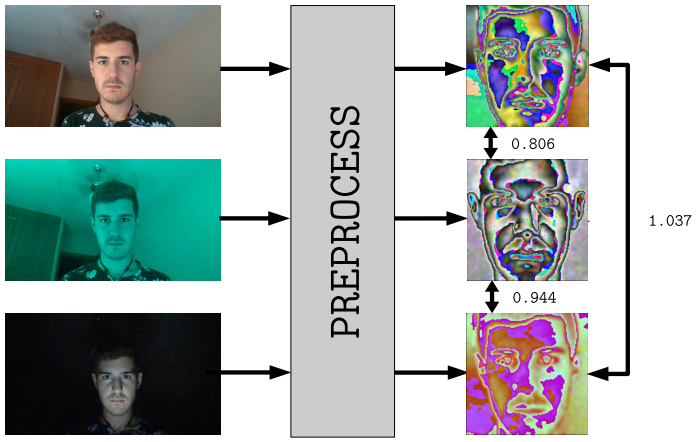

Fig. 5. Relative distances between the same face on different lighting situations.

5 Actuation module

With the *perception* module the system obtains the maximum possible certainty about the persons present in the current RGB image, as well as their condition of being or not the one to be followed. The *actuation* module is responsible of generating a suitable command to the robot motors, in order to move towards the target person, in case it is being tracked in the current frame. To do so, it follows its own pipeline, explained next.

The input for this module is the information yielded by the *perception* module: the *tracked persons* parameters (position, face, "is or is not the target person"). From this information, it has to infer the proper robot movements taking also into account the state of the system on the last iteration. This is implemented with a *case-based* behavior, which follows the *flow chart* represented on Fig. 6. The response mainly depends on *mom* (the name given to the target person), as it can be *lost* or *tracked and followed*.

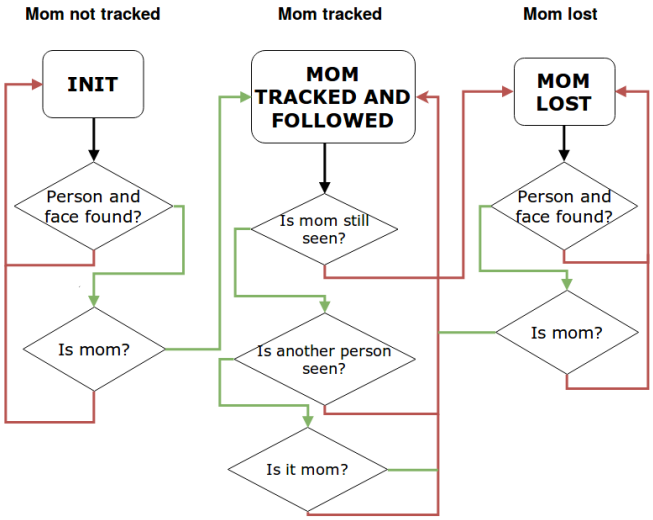

Fig. 6. *Flow chart* followed by the case-based behavioral, depending on the previous state.

If the target person is found among the tracked persons, the system follows it. It can be observed that the system does not need a continuous face feedback to follow the target person. The time-spatial continuity provided by the previously described trackers allows a stable tracking even when the person is only visible from her back side. In addition, if a new face is found and it satisfies the criteria of similarity with the reference face, then the *followed* role is switched to that new person, which starts being followed by the robot.

This actuation module has been implemented within a computer iterative thread, and its pipeline is executed *once* per thread iteration. This results on a new speed command pair (angular and linear speeds) to the motor base on each iteration.

5.1 Error computation

Once the system has recognized the target person inside the image, in case it is being seen, it proceeds to an *error computation*, in order to determine the strength of the required rotation and traslation to move towards that person. As the robot moves over the ground plane, with two degrees of freedom, the system computes two errors: angular error and distance error.

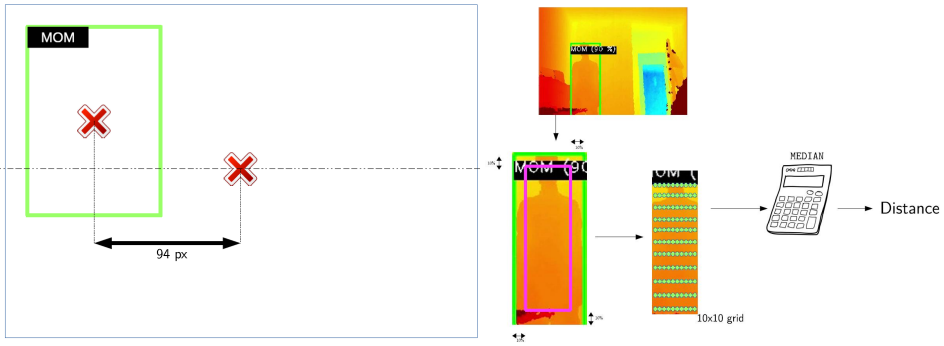

Fig. 7. Computation of the angular error (left) and distance error (right) between the system and the followed person.

The most desirable robot orientation with respect to the person is to be aligned with her. This way, the person appears right in the horizontal center of the image. Hence, we can compute the *angular error* as the subtract of the center coordinates and the center of the person's bounding box, in the horizontal dimension (Figure 7 (left)). This error gives an approximate idea of the necessary turn to align the robot and the target person.

The used sensor is a RGBD camera, which provides *depth* images. These depth images are aligned with the RGB ones, which means that the coordinates of the person inside the depth image are the same than the RGB ones. Hence, it allows to *locate* the person inside the depth map and measure the *distance error*. For the sake of robustness, this has been implemented using a 10×10 grid sampling of the depth values inside the person box (putting care on avoiding the margins, in order to measure only inside the person). This allows to collect a set of measured distances to the person. It computes the *median* of that set of measurements, taking the result as the real distance from the robot to the person, as seen on Fig. 7 (right).

5.2 Movement control

Both angular and distance errors are computed to determine the *relative position* of the robot and the target person. They are used as an input to compute the most suitable control response. As the purpose is not to move the robot literally to the position of the person, but to just maintain a following behavior, a *desirable zone* is established in each degree of freedom. When the target person is found inside these desirable zones (illustrated on Fig.8), the robot will not move towards her, as the person is considered *under control*. They are also known as dead zones as no correction from control is needed.

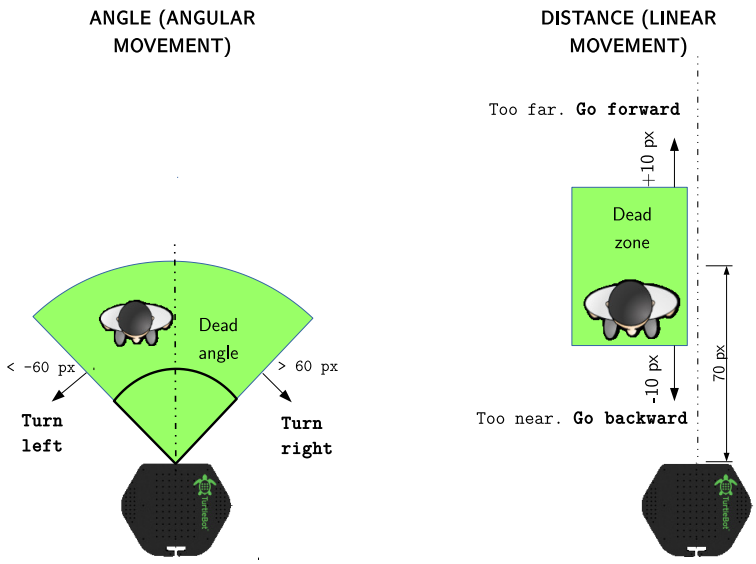

Fig. 8. Dead zones on each dimension, where the person is considered under control.

If the person is outside these dead zones, a physical correction response is required, with the purpose of *seeing* that person again inside the dead zone. For this action, each dimension implements a *PID* controller, which establishes a *closed-loop* feedback, as described on [8]. This allows to keep in mind previous responses, to achieve the optimum fitting on each iteration. This means, for example, to accelerate if the person is not going any closer, or to step hard on the brake if the person suddenly gets too close.

The implemented *PID* controllers (an angular and a linear one) have been experimentally tuned to obtain the most suitable parameters for our operation, obtaining the values in Table 2.

	Linear	**Angular**
k_p	2	7
k_d	0.1	0.5
k_i	3	10

Table 2. Optimal found values for the parameters in each PID controller.

This way, the system can output a speed command with a tight adjustment to the values required by the situation of the current iteration. The response obtained from this value is a *reactive* one. This means that each value results in a new movement command, avoiding to perform movements longer than one iteration. For smoothness sake, what is sent to the motors is not that value, but the *mean* between it and the last sent one. This way, it results on a slightly longer convergence that helps to remove sudden movements.

a) Frontal view b) Side view

Fig. 9. Robot used for the validation experiments.

6 Experiments

The developed system has been experimentally validated on a real robot and an indoor scenario. The TurtleBot-2 robot was used on the experiments, including an Asus Xtion Pro Live RGBD sensor. The onboard computer was a regular laptop, equipped with an Intel i5-4210U CPU, a DDR3 RAM of 8GB and a Nvidia 940M GPU (Figure 9). On these hardware resources (using GPU parallelization to run the neural networks, and choosing a lightweight *SSD Lite* model), we achieved a stable detection rate of 14 fps, fairly enough to guide the robot on a seamless way.

The person detection and reidentification process can be observed on Fig. 10. The system is succesfully able to re-identify the target person (*mom*), even in challenging lightning conditions like in Figure 10.

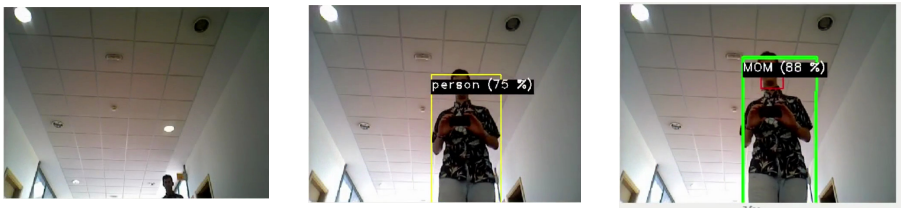

Fig. 10. Detection and face re-identification of the target person.

The complete system results show a robot finely following a particular person, even when that person does not face towards the robot. The behavior is fluent, with a refresh rate of 10 movements per second (as the SSD CNN is the lightest possible model, as seen on Table 1). This agility of the neural network for people detection helped substantially to minimize the time bottleneck. The following behavior can be observed on Fig. 11. Several videos of a typical execution are also publicly available [4].

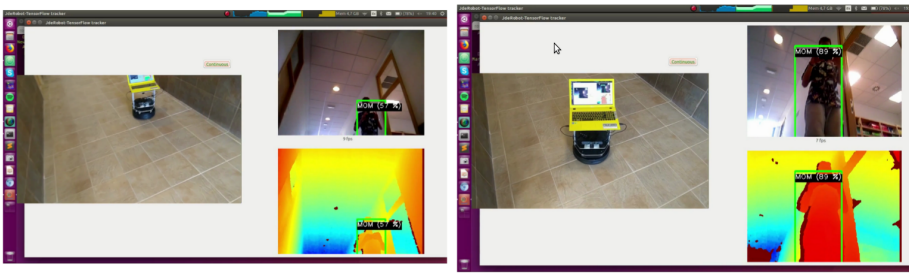

Fig. 11. Following behavior.

7 Conclusions

In this paper an algorithm for person following behavior inside a robot has been presented. The proposed system has been implemented with two concurrent modules in a Python program, one for *perception* and another one for *actuation*.

[4] Full video test available on https://www.youtube.com/watch?v=oKMR_QCT7EE

For perception it uses a pretrained people detection network, a regular face detector based on Haar features, some filters to alleviate false positives and false negatives, and a FaceNet neural network for person re-identification. The visual information is combined with depth data from the RGBD sensor to estimate the relative position of the target person from the robot. The person detection network was carefully selected to meet the real time operation requirement, typical in robotics. All this leads into a fast and efficient following system, capable of keeping the track on a particular individual even without a continuous checking of her identity or detection of her face.

For actuation it uses two PID controllers which are based on angular error and distance error between the robot and the target detected person.

All these functionalities have been combined into a software node that runs on real time. This allows a simple robot to perform a successful tracking of a person, just knowing her face beforehand. The source code of the entire system is available online [5].

Several experiments with a real robot have been performed and are also publicly available. They validate the successful operation of the proposed system. The use of deep learning neural networks provide high robustness to the image based person detection, which works even on tough lighting conditions. This has been the main design restriction developing this system, which can serve as a first prototype of a powerful people following platform.

This work can be extended in several lines. For instance using another neural network for face detection instead of Haar features. In addition, the use of faster networks for people detection, like YOLO DarkNet ones, is being explored. For this purpose, a rewriting of the node software in C++ would be required.

References

1. Dalal, Navneet and Triggs, Bill. Histograms of oriented gradients for human detection, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005.
2. Muñoz-Salinas, Rafael and Aguirre, Eugenio and García-Silvente, Miguel. People detection and tracking using stereo vision and color *Image and Vision Computing* 25(6), Elsevier, 2007.
3. Aguirre, Eugenio and García-Silvente, Miguel and Pascual, Daniel, A multisensor based approach using supervised learning and particle filtering for people detection and tracking, *Robot 2015: Second Iberian Robotics Conference*, Springer, 2016
4. R.Calvo, J.M.Cañas, L.García-Pérez. Person following behavior generated with JDE schema hierarchy *Poster in ICINCO 2nd Int. Conf. on Informatics in Control, Automation and Robotics.* Barcelona (Spain), sep 14-17, 2005. INSTICC Press, pp 463-466, 2005. ISBN: 972-8865-30-9
5. Lui, W., Anguelov, D., Erhan, D., et al. SSD: Single-Shot Multibox Detector. *CoRR*, abs/1512.02325, 2015.
6. Viola, P., Jones, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer*

[5] https://github.com/roboticsurjc-students/2017-tfg-nacho_condes

Vision and Pattern Recognition. CVPR 2001, volume 1, pages I-511-I-518 vol.1, 2001.
7. Schroff, F., Kalenichenko, D., Philbin, J. *FaceNet*: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.038032, 2015.
8. Åström, KJ., Murray, RM. *Feedback Systems: An Introduction for Scientists and Engineers*, 2004.
9. Xue, Hongyang and Liu, Yao and Cai, Deng and He, Xiaofei. Tracking people in RGBD videos using deep learning and motion clues. *Neurocomputing* 204, pages 70-76, Elsevier, 2016.
10. Koide, Kenji and Miura, Jun. Identification of a specific person using color, height, and gait features for a person following robot. *Robotics and Autonomous Systems* 84, pages 76-87, Elsevier, 2016.
11. Munaro, Matteo and Ghidoni, Stefano and Dizmen, Deniz Tartaro and Menegatti, Emanuele. *A feature-based approach to people re-identification using skeleton keypoints*. 2014 IEEE International Conference on Robotics and Automation (ICRA).
12. Welsh, John Bradford. Real-Time Pose Based Human Detection and Reidentification with a Single Camera for Robot Person Following. *PhD Thesis, University of Maryland, College Park*, 2017.
13. Yoon, Y and Yoon, H and Kim, J Person Reidentification in a Person-following Robot 25th *IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2016.
14. Shimura, Kouyou and Ando, Yoshinobu and Yoshimi, Takashi and Mizukawa, Makoto. Research on person following system based on RGB-D features by autonomous robot with multi-kinect sensor. 2014 *IEEE/SICE International Symposium on System Integration (SII)*.
15. Ilias, B and Shukor, SA Abdul and Yaacob, S and Adom, AH and Razali, MH Mohd. A nurse following robot with high speed Kinect sensor. *ARPN Journal of Engineering and Applied Sciences* 9(12), pages 2454-2459, 2014.
16. Yoshimi, Takashi and Nishiyama, Manabu and Sonoura, Takafumi and Nakamoto, Hideichi and Tokura, Seiji and Sato, Hirokazu and Ozaki, Fumio and Matsuhira, Nobuto and Mizoguchi, Hiroshi. Development of a person following robot with vision based target detection. 2006 *IEEE/RSJ International Conference on Intelligent Robots and Systems*.
17. Satake, Junji and Miura, Jun. Robust stereo-based person detection and tracking for a person following robot. *ICRA Workshop on People Detection and Tracking*, 2009.
18. Sidenbladh, Hedvig and Kragic, Danica and Christensen, Henrik I. A person following behaviour for a mobile robot *Proceedings of the IEEE International Conference on Robotics and Automation*, 1999.
19. Howard, Andrew G. and Monglong Zhu et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision *CoRR journal*, volume 1704.04861, 2017.