# Embedded Deep Learning Solution for Person Identification and following with a Robot

Ignacio Condés[1], José-María Cañas[2], and Eduardo Perdices[2]

[1] Universidad Carlos III de Madrid, Leganés (Madrid), Spain,
[2] Universidad Rey Juan Carlos, Fuenlabrada (Madrid), Spain

**Abstract.** This research presents a robust embedded system for following a person, making use of a pipeline of convolutional neural networks. Besides, it features an optical tracking system for supporting the inferences of the neural networks, allowing to determine the position of a person using an RGBD camera. The system is deployed using ROS, and runs in a NVIDIA Jetson TX2, an embedded SoM (*System-on-Module*), capable of performing computationally demanding tasks onboard, and coping with the complexity required to run a robust tracking and following algorithm. The board is attached to a robotic mobile base, which receives velocity commands to move the system towards the target person.

**Keywords:** deep learning, computer vision, robotics, person following

## 1 Introduction

Last decades, the production prices of digital cameras and high-resolution sensors have been greatly reduced, bringing these devices into the consumer market segment. This, beside an increase in the hardware performance, has resulted in a strong drive for the computer vision research.

In particular, *deep learning* has been massively employed for addressing high complexity tasks, such as language understanding, speech recognition and computer vision problems.

Moreover, robotics applications can be really useful at daily tasks, specially when a robot behavior tends to emulate the human one. This requires a polished (and somehow complex) behavior, triggered by a certain input. Depending on the source of the behavior, two main branches emerge in robotics: *teleoperated robots*, useful for hazardousness or high-precision environments, and *autonomous robots*, which aim to act by themselves. The decision-making and actuation capabilities rely on the robot, due to a compromise in the design, or for purpose-specific systems, such as social robots [1], [2].

The important advances on computer vision and robotics allow to explore outstanding synergies, as this work aims to take advantage of: obtaining a robot capable of following a certain person, navigating towards them on a robust reactive behavior, and using deep-learning-based visual perception. As it will

be explained later, this behavior is composed of two main components: the *perception block*, in charge of processing the images from an embedded RGBD camera, and the *actuation block*, which moves the robotic base accordingly to the relative position of the person to be followed. This application can be specially interesting on social robots, which are designed to follow a person at home or in a hospital [3].

This research improves the system proposed on [4], where a neural-network-based following system was run in a standard laptop, with a camera and a robot plugged.

## 2 Related works

The person following behavior is a old problem in Human-Robot-Interaction field with many existing solutions in the literature [5], [6]. Another approach [7] uses laser sensors to detect the legs and a particle filter to provide robustness and continuity to the person estimation.

The topic has been extensively explored using a single camera [6] or stereo vision. [8] combines clothes color and position information to detect and track multiple people around, using Kalman filters and fusing the plain-view map information with a face detector. [9] uses depth templates of person shape applied to a dense depth image and an SVM-based verifier for eliminating false positives, employing an Extended Kalman Filter to continuously estimate the person positions. In addition, Histograms of Oriented Gradients are a classical and effective method for human detection [10], which is a relevant ingredient of the person following behavior.

In the last years, other approaches make use low cost RGBD sensors [11], [12], taking advantage of their depth information. It 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 [13] deep learning techniques on color and motion cues have been used to successfully detect people on RGBD videos for surveillance applications.

The identification of a particular person is a complementary problem to general people detection. There are image-based methods [14] and video-based methods [15]. Recent papers also use deep learning to identify the detected persons [16]. Other methods use features and skeleton keypoints [17]. [18] use image and range data combining color, height and gait features (step length and speed) for a robust identification, even in severe illumination environments. [19] is an interesting work that uses neural networks to both multiple person detection and a particular person re-identification, working in real-time on moderate hardware using a single camera.

Deep learning techniques yield interesting results as well. [20] uses deep neural networks in order to extract features from an image in order to identify the person in the image. Other approaches such as [21] make use of recurrent networks and siamese architectures to reidentify a person.

# 3 Design

The proposed system features a *Perception* module and an *Actuation* module, as shown in Figure 1. The *Perception* module is responsible for extracting pertinent information from the image: the position and identity of the person to be followed. This information serves as input to the *Actuation* module, which takes a decision about the action to be performed in order to move the robot towards the person. These movements have to be reactive, happening as soon as possible whenever the person position changes.

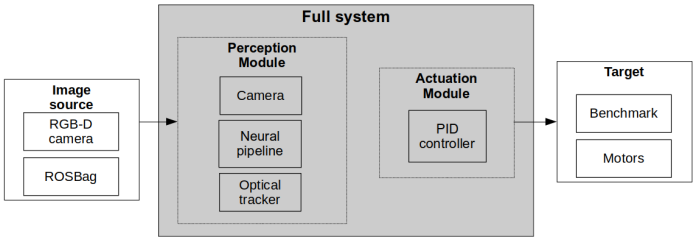



Fig. 1: Architecture of the proposed solution.

## 3.1 Perception Module

This module processes the camera images for determining the location of the person to be followed. It is composed of the components shown in Figure 1, which are described below.

**Camera:** The robot uses an RGBD device (ASUS Xtion Pro Live) as visual sensor. The system makes use of ROS topics to retrieve the latest *(RGB, depth)* pair on an asynchronous way.

It supports both an *online* source (camera) or an *offline* source (recorded ROSBag), with a transparent adaptation to the rest of the system.

**Neural pipeline:** Images are passed through a pipeline of three neural networks, which detect the persons in the scene and identify which one must be followed. The main objective of this work is to take advantage of the robustness offered by deep-learning-based techniques.

1. **Person detection:** For addressing this task, several deep models have been tested, varying the base network architecture and its depth. Since one of the objectives of the system is to work on a portable (low-power) system, only those architectures which yield a good performance with a sufficiently low

inference time are considered. The two tested models for this purpose are the SSD-MobileNet [22], [23] using a MobileNet [23] for feature extraction, and the *tiny* version[3] of YOLOv3 [24]. These models are already trained and publicly available on the TensorFlow Model Zoo [25] and on repositories hosted on GitHub[4]. In-depth tests have been conducted to compare the performance of these two models, as it will be described in Section 4.

2. **Face detection:** This problem can also be addressed using a detection neural network. Since the models previously described are not suitable for detecting faces, the adopted solution is a single-class detection system. The network trained in [26] implements a two-stage neural network capable of detecting faces. This detector is based on YOLOv2 [27], which ensures a high-speed and efficient detection based on a class-specific neural network. [26] contains a video sequence comparing the accuracy of this system against a classical Haar cascade approach [28].
3. **Face identification:** Once the face of a person has been detected, it can be used as a discriminant feature for determining their identity. For this purpose, *FaceNet* [29] has been used to perform the identification, using a publicly available implementation in TensorFlow[5]. This deep identification system transforms the image of a face is transformed into a 128-dimensional vector, known as projection or *embedding*. This transformation is learned after a triplet-loss training process, which separates different faces as much as possible, while projecting similar faces as close as possible. Ideally, it produces similar projections when two images of the same face are evaluated, despite different lighting conditions (as a channel-wise normalization step is performed before passing the image through the network).

The described neural pipeline provides *person locations, face detections and face projections* from a single image, taking advantage of the flexibility and robustness that deep learning methods offer, in order to address three different problems in an efficient way. Its functionality has been depicted in Figure 2.

Furthermore, these neural networks have been optimized using the library TensorRT[6]. This engine modifies segments from the architecture of a given network, tuning certain parameters:

**MSS (*Minimum Segment Size*):** the threshold above which a segment is selected to be replaced by the TensorRT optimization. Increasing this value makes the optimizer more selective, optimizing only the heaviest segments of the network. A low value may cause an excessively high overhead, resulting in a worse performance than using the original graph.

[3]The tiny version of YOLOv3 is used to the limited memory on the Jetson TX2 board. The full YOLOv3 model requires more memory than the available size.

[4]https://github.com/mystic123/tensorflow-yolo-v3

[5]https://github.com/davidsandberg/facenet

[6]https://developer.nvidia.com/tensorrt

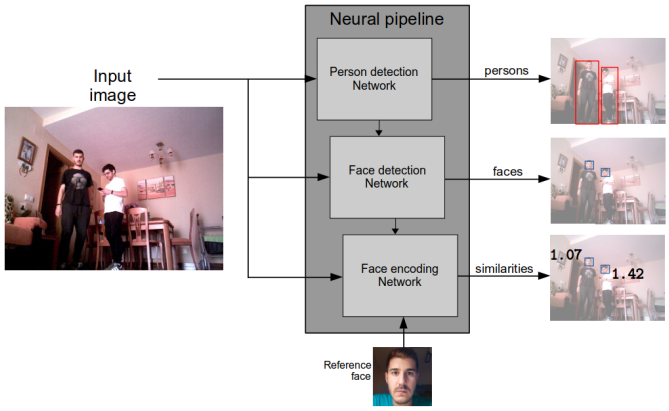



Fig. 2: Neural pipeline, showing the cascade of the three neural networks used to output persons, faces and similarities with the reference face.

**MCE (*Maximum Cached Engines*):** TensorRT keeps a cache of engines on runtime, with the purpose of reducing the time spent for loading them into the GPU. This parameter modulates the amount of engines cached.

**Precision mode:** typically, the weights and parameters of the trained neural networks are handled as 64-bit floating point numbers. A reduction in the precision to 32-bit or 16-bit achieves very similar results (as shown on Section 4), making the operations much lighter. An additional reduction to 8-bit numbers can be performed as well, mapping the weights values to 256 steps. This greatly reduces greatly the inference time.

Tuning these parameters led to optimized versions of the networks included in the neural pipeline, as Section 4 shows.

**Motion Tracker:** The previously depicted *Neural* component outputs reliable inferences with a certain refresh rate, namely $k$ frames. If $k$ is too high, the system may be affected by an important delay when the movement is performed. This may lead to unsteady movements, increasing the probability of losing the reference person. To avoid this, a *Motion Tracker* component is added to the system. Its functionality is to *estimate* the person movement along $k$ frames, while the neural pipeline is performing the next detection.

The method chosen for this purpose is a *Lucas-Kanade* visual tracker [30]. This technique estimates the *motion field* between the images taken in two time instants, addressing the problem using a differential approach [31].

However, in this use case, the objective is not to compute the entire optical flow, but limited to the pixels inside and surrounding the persons in the scene.

For this purpose, the tracking is performed on the corners inside the detected persons. These corners are found using a Shi-Tomasi corner detector [32].

### 3.2 Actuation Module

**PID Controllers:** The last block of the system is responsible of translating the location information of the reference person into velocity commands. The implemented approach is a reactive controller, which moves the robot aiming to keep the person inside *safe zones*. These safe zones (Figure 3) can be described as follows:

**Angular zone:** the reference person has to be placed at the horizontal center of the image, with a margin of ±50 pixels on the sides.

**Linear zone:** the reference person has to be placed at a distance of 1 m with respect to the robot front, with a distance margin of ±30 cm.

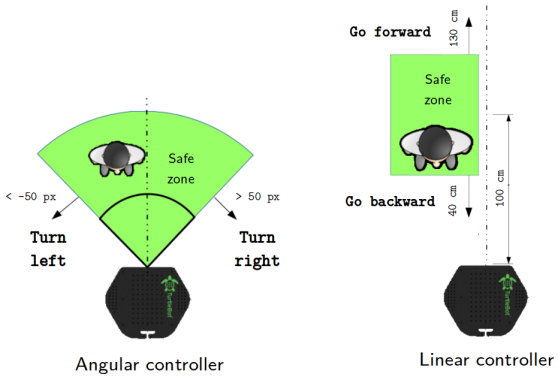



Fig. 3: Safe zones for each controller.

To place the person inside these safe zones, the robot has to move on certain directions. For determining a movement, an *error* vector $(e_x, e_w)$ is computed (Figure 4), using the tracked person coordinates:

$e_x$**:** the linear error is computed using the depth image, estimating the distance from the robot to the person. Since the Xtion sensor registers the depth image into the RGB one, the person coordinates can be used in the depth image in order to find the distance of each pixel inside the bounding box of the reference person: the *person depth map*. For avoiding accidental samples of the background of the person, the edges of the depth map are trimmed. Later, a 10x10 grid is computed to have 100 uniformly distributed samples of the depth of the person. The linear error can be computed as the median value of the samples.

$e_w$**:** the angular error is computed computing the difference on the horizontal coordinate between the image center and the center of the bounding box of the reference person.

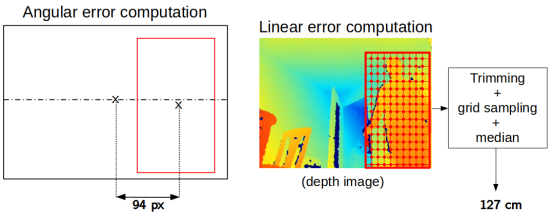



Fig. 4: Error computation on each controller.

These errors yield a reactive response, dependent on the relative distance between the person and the safe zones. In order to compute the suitable velocity commands (linear and angular responses), a PID controller is implemented for each dimension, alleviating overshooting and inertial behaviors. The PID parameters $(k_p, k_i, k_d)$ were tuned to deliver a soft response on the robot movements. Table 1 shows the chosen PID values for following a wandering person on typical indoor conditions.

| | **Linear** | **Angular** |
|---|---|---|
| $k_p$ | 0.4 | 0.005 |
| $k_d$ | 0.04 | 0.0003 |
| $k_i$ | 0.05 | 0.006 |

Table 1: Optimal found values for the parameters in each PID controller.

## 4 Experiments

Several experiments have been conducted in order to tune the parameters for improving the system performance, accuracy and robustness, evaluating the same video sequence under different configurations. A recurrent metric on these test is the IoU score, which assesses the overlapping quality between the predicted and the true bounding boxes. As this metric requires ground-truth labels on the videos, the annotation tool LabelMe [7] has been used to indicate the ground-truth position of the persons on each sequence.

[7] https://github.com/wkentaro/labelme

### 4.1 Person detection experiments

This experiment compares the performance of the two object detection architectures tested on our system: YOLO [24] and SSD[22]. In the case of YOLO, the implemented version is YOLOv3, in its *tiny* version. For an SSD-based detector, on a real-time application the most convenient variants are those which use a MobileNet [23] as a feature extraction network. The TensorFlow Model Zoo[8] offers several pre-trained models implementing this network, along which a selection has been carried out. The chosen model integrates a MobileNet v1 whose weights have been quantized [33] in order to reduce the computational cost without reducing the accuracy.

In order to quantify the different accuracy vs. inference time tradeoffs that these architectures offer, a specific test has been designed. A specific video sequence of 721 frames long has been recorded, containing a person wandering across the field of view of the camera. For every frame of the sequence, the persons are detected using YOLO and SSD respectively, and the IoU with the ground-truth labels, as well as the inference time have been measured, as it can be seen on Figure 5. Some gaps can be noticed on the detections, corresponding to frames where the person was out of the sight of the camera.

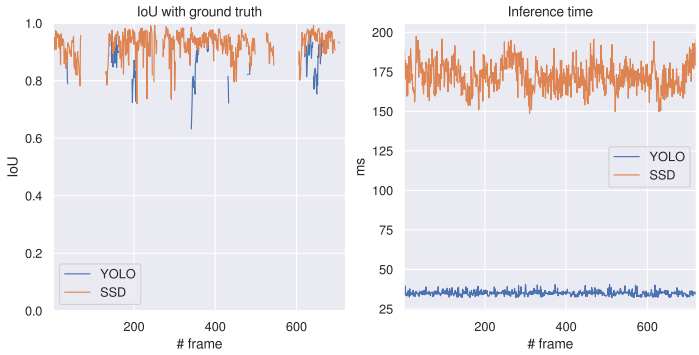



Fig. 5: Results of the person detection test: IoU score with ground truth (left) and inference time per frame (right). A discontinuity represents absence of detections.

The YOLO-based detector offers a slightly minor average IoU than the SSD-based one (0.858 and 0.926 respectively), while taking 5 times less average time to make inferences (35 ms vs. 172 ms). On these terms, the YOLO-based detector seems much more efficient. However, the YOLO detector is able to find the person only in 17% of the frames, whereas the SSD one finds the person in 74%. However,

[8] https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

these percentages are a relative measurement for comparing both models, since not all the frames in the sequence contained a person.
This shows that the YOLO detector is too dependent on pose and lighting conditions for the detections to be successful. On the other hand, the SSD detector yields steady predictions, only cutting on the periods where the person was truly out of the field of view. Hence, this system is much more robust for our application scenario, so YOLO is discarded in favor of SSD.

### 4.2 Face detection experiments

This experiment is devoted to compare the performance of Haar cascade classifier and the neural face detection, using the same video sequence than the previous experiment. For each frame in the sequence, the faces are extracted using each one of the described methods, and the IoU score is computed with the ground truth face bounding box. The result can be visualized in Figure 6.

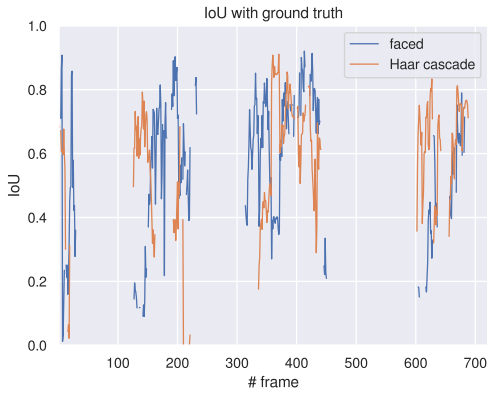



Fig. 6: IoU score with the ground truth for each one of the face detection systems.

It can be seen that both methods yield similar IoU scores and drop at the same time when the person turns their back to the camera. However, the `faced` implementation (which uses deep learning to predict the face positions) is capable of keeping a non-zero IoU at several instants where the Haar performance drops to zero. This is due to pose variances of the person, as the main drawback of the Haar cascade classifier is that it is only capable of detecting frontal faces, dropping the performance whenever the person turns the face towards a side. Both methods yield similar IoU on average, but the deep-learning approach, `faced`, detects a face in 36.89% of the frames, whereas the Haar cascade slightly drops the detection rate to 34.40%. Hence, this test validates the improvement of the face detection performance when using a specific neural network trained for that purpose.

### 4.3 Face recognition experiments

The last component of the neural pipeline is a *face recognition* neural network, devoted to confirm the identity of the reference person. This subsystem is based on a FaceNet [29] network, which projects a face into a 128-dimensional space. These projections are used by the proposed system, as their Euclidean distance to the projection of a reference face is used to determine if the input face belongs to the reference person.

This experiment is designed to assess the quality of the projection system, which should yield far points for a different face and near points for a matching face. For this proposal, a video sequence was recorded containing two persons wandering in front of the robot. The faces of each frame are labeled, separating the faces of the two persons in two different classes. For computing the distance, a reference face was set using an image, and the distance to the reference face of each one of the faces in the video was stored. The result can be observed on Figure 7.

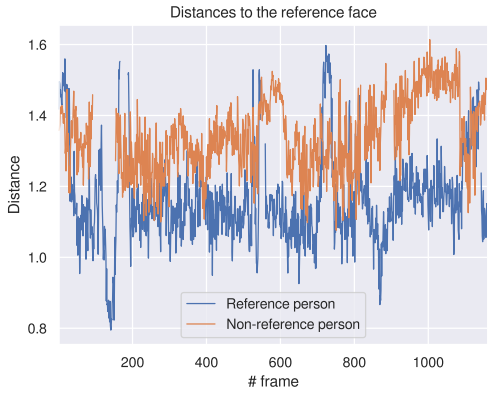



Fig. 7: Resulting distance for the two faces in the video.

The results obtained on Figure 7 allow to extract two conclusions about the quality of the projections of the faces. First, the encodings of the reference person have an overall remarkable stability. On average, the obtained projections for every frame are located at an approximate distance of 1.16 (threshold chosen for accepting a person as the reference one). Exceptional rises in the distance are due to changes in the pose of the face and occlusions, that reduce the quality of the projection. Second, the encodings of a person different than the reference one have an overall higher distance (1.34) from the reference face. This is convenient for avoiding false positives while determining that a face is the reference one.

These results validate FaceNet as a robust approach to perform person recognition tasks, since the distance of a projection to the reference face has to be below the threshold for being labeled as the reference face.

### 4.4 Deep Learning optimization with TensorRT

Among the parameters that allow to configure the TensorRT optimization, there is a reduction in the floating-point precision level of the nodes. This experiment aims to compare the loss of precision this entails with the reduction in the inference time, result of an optimization process on the SSD-based object detector used in subsection 4.1.

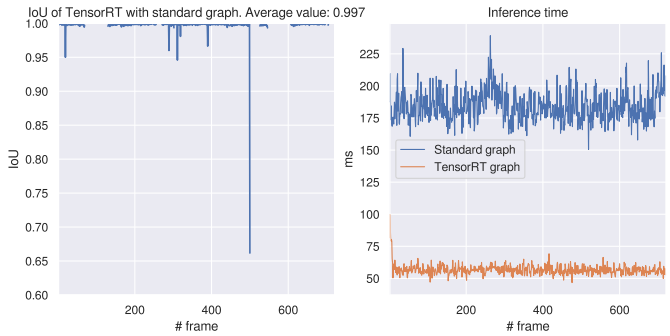



Fig. 8: IoU between the standard graph and the TensorRT graph inferences (left) and inference times for both networks (right).

Figure 8 illustrates the differences between a standard graph and an optimized one. On the one hand, this loss of precision is small, with some observable exceptions with a loss above 5% of the original performance. On the other hand, the inference time gap can be observed as well. The difference is more notorious, as the TensorRT optimized model performs the inferences 3 times faster than the original graph (56.769 ms vs. 184.477 ms). aside of more stability on the inference time. Given these results, the TensorRT optimizations are a convenient tool to greatly increase the performance of the system, allowing the slower component (the neural pipeline) to experiment an important reduction on the inference time. This greatly improves the overall performance, as reliable neural updates are performed more often.‘

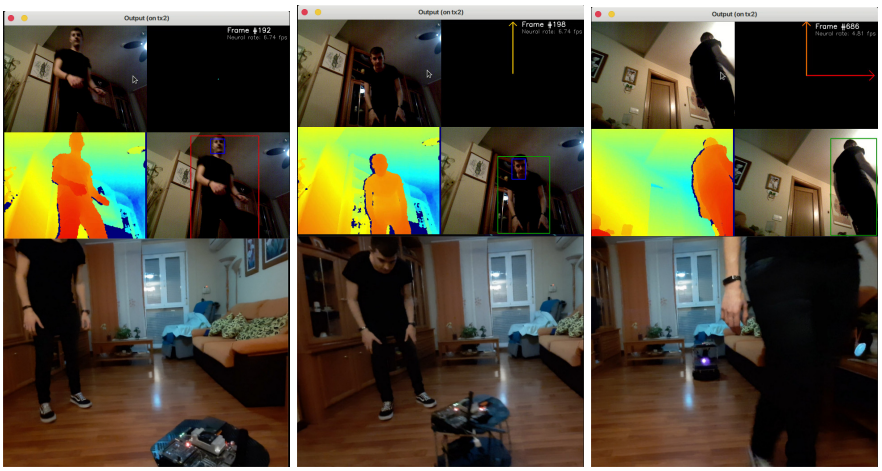
### 4.5 Typical execution

As a result, The whole system running may seen on YouTube[9]. This video presents a sequence where the reference person enters into the field of view of

[9]https://www.youtube.com/watch?v=WZ0riKMwJWA

the robot, shows his face to the camera. After some consecutive frames detecting the person, it is identified using the detected face. When the face projection is close enough to the reference one, the robot starts following that person. For each frame, the linear and angular errors are computed, even if the face of the person is not seen anymore, as the system has checked previously that that is the right person. If the errors are outside the safe zones, a velocity command is computed and sent to the robot. This routine is executed until the person gets lost, causing the robot to stop waiting for the person to be seen again.

Figure 9 shows the behavior of the robot, expected to follow the person properly. The top region of the images shows several screen captures of the program output, with the RGBD images (left), and the movement commands and tracked persons (right). The bottom region of the plots show the scene recorded from a mobile phone, allowing to observe the performance externally.



(a) Person detection. (b) Person recognition and following. (c) Following without face feedback.

Fig. 9: Typical execution.

## 5 Conclusions

The proposed approach presents an embedded system that follows a reference person, relying on the robustness of deep learning for being capable of working in real environments. As it has been shown, the detection and recognition pipeline has been exclusively designed using deep neural networks, ensuring a robust performance in non-controlled environments. This robustness is crucial, given

that the camera is located at a very low position and has a vertical inclination in order to see the full body of the person. However, this also causes the camera to perceive an excessive brightness from ceiling lamps, dimming the persons on the image. Classical systems tend to fail due to this issue.

This neural pipeline is by a tracking component, improving the performance under certain issues, such as partial occlusions or a higher inference time if the power available makes the inferences less efficcient.

The main contributions of this research may be summarized as follows:

- The system is mounted on a battery-powered *mobile base*, which features a high-performance GPU embedded on a SoM. This assembly can operate autonomously, without requiring an external computer to perform the deep learning inferences or running algorithms in parallel. Remote monitoring is available, but not required for the system to work.
- Specific optimization engines allow the system to run efficiently 3 neural networks on a low-consumption hardware. These networks perform inferences over the images captured by the RGBD sensor attached to the system. The inferences are devoted to detect the different persons in the scene, as well as to distinguish them by means of a discriminant feature: their face. Using neural networks for detection and identification tasks allows the system to be robust and reliable.
- The neural system is supported also an optical person tracker. It aims to guess the trajectories followed by each person while the neural network yields a new update, as this tracker predicts the person displacement considerably faster that the neural network. Since neural inferences are also sensitive to flickering caused by occlusions, trusting just on these inferences could easily result on an unsteady behavior. Thus, the introduction of the tracker improves the system robustness and softens the robot movements.

## References

[1] P. Pennisi *et al.*, "Autism and social robotics: A systematic review," *Autism Research*, 2016. DOI: 10.1002/aur.1527. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/aur.1527. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/aur.1527.

[2] A. Cheong *et al.*, "Development of a robotic waiter system," *IFAC-PapersOnLine*, vol. 49, no. 21, pp. 681–686, 2016, 7th IFAC Symposium on Mechatronic Systems MECHATRONICS 2016. DOI: https://doi.org/10.1016/j.ifacol.2016.10.679. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2405896316322911.

[3] R. Gockley, J. Forlizzi, and R. Simmons, "Natural person-following behavior for social robots," in *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, ser. HRI '07, Arlington, Virginia, USA: Association for Computing Machinery, 2007, pp. 17–24.

[4] I. Condés and J.-M. Cañas, "Person Following Robot Behavior Using Deep Learning: Proceedings of the 19th International Workshop of Physical Agents (WAF 2018), November 22-23, 2018, Madrid, Spain," in. Jan. 2019, pp. 147–161.

[5] H. Sidenbladh, D. Kragic, and H. I. Christensen, "A person following behaviour for a mobile robot," in *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, IEEE, vol. 1, 1999, pp. 670–675.

[6] T. Yoshimi *et al.*, "Development of a person following robot with vision based target detection," in *Intelligent Robots and Systems, 2006 IEEE/RSJ International Conference on*, IEEE, 2006, pp. 5286–5291.

[7] E. Aguirre, M. García-Silvente, and D. Pascual, "A multisensor based approach using supervised learning and particle filtering for people detection and tracking," in *Robot 2015: Second Iberian Robotics Conference*, Springer, 2016, pp. 645–657.

[8] R. Muñoz-Salinas, E. Aguirre, and M. García-Silvente, "People detection and tracking using stereo vision and color," *Image and Vision Computing*, vol. 25, no. 6, pp. 995–1007, 2007.

[9] J. Satake and J. Miura, "Robust stereo-based person detection and tracking for a person following robot," in *ICRA Workshop on People Detection and Tracking*, 2009, pp. 1–10.

[10] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, 886–893 vol. 1.

[11] B. Ilias *et al.*, "A nurse following robot with high speed Kinect sensor," *ARPN Journal of Engineering and Applied Sciences*, vol. 9, no. 12, pp. 2454–2459, 2014.

[12] K. Shimura *et al.*, "Research on person following system based on RGB-D features by autonomous robot with multi-kinect sensor," in *System Integration (SII), 2014 IEEE/SICE International Symposium on*, IEEE, 2014.

[13] H. Xue *et al.*, "Tracking people in RGBD videos using deep learning and motion clues," *Neurocomputing*, vol. 204, pp. 70–76, 2016.

[14] D. Kouno, K. Shimada, and T. Endo, "Person identification using top-view image with depth information," in *2012 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, 2012, pp. 140–145.

[15] J. You *et al.*, "Top-push video-based person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[16] Y. Yoon, H. Yoon, and J. Kim, "Person Reidentification in a Person-following Robot," in *25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, 2016.

[17] M. Munaro *et al.*, "A feature-based approach to people re-identification using skeleton keypoints," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, IEEE, 2014, pp. 5644–5651.

[18] K. Koide and J. Miura, "Identification of a specific person using color, height, and gait features for a person following robot," *Robotics and Autonomous Systems*, vol. 84, pp. 76–87, 2016.

[19] J. B. Welsh, "Real-time pose based human detection and re-identification with a single camera for robot person following," PhD thesis, University of Maryland, College Park, 2017.

[20] E. Ahmed, M. Jones, and T. K. Marks, "An improved deep learning architecture for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[21] N. McLaughlin, J. Martinez del Rincon, and P. Miller, "Recurrent convolutional network for video-based person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016.

[22] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," *Lecture Notes in Computer Science*, pp. 21–37, 2016.

[23] A. G. Howard *et al.*, *Mobilenets: Efficient convolutional neural networks for mobile vision applications*, 2017. arXiv: 1704.04861 [cs.CV].

[24] J. Redmon and A. Farhadi, *YOLOv3: An Incremental Improvement*, 2018. arXiv: 1804.02767 [cs.CV].

[25] ModelZoo, *TensorFlow Object Detection: Model Zoo*. [Online]. Available: https://modelzoo.co/model/objectdetection.

[26] I. Itzcovich, *faced: CPU Real Time face detection using Deep Learning*. [Online]. Available: https://towardsdatascience.com/faced-cpu-real-time-face-detection-using-deep-learning-1488681c1602.

[27] J. Redmon and A. Farhadi, *Yolo9000: Better, faster, stronger*, 2016. arXiv: 1612.08242 [cs.CV].

[28] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," vol. 1, Feb. 2001, pp. I–511.

[29] F. Schroff, D. Kalenichenko, and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015.

[30] I. González-Díaz, "Computer Vision: Dense Motion Estimation," University Lecture, 2020.

[31] B. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision (IJCAI)," vol. 81, Apr. 1981.

[32] Jianbo Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1994.

[33] GoogleAIBlog, *Accelerating Training and Inference with the Tensorflow Object Detection API*. [Online]. Available: https://ai.googleblog.com/2018/07/accelerated-training-and-inference-with.html.