uc3m | Universidad Carlos III de Madrid

Master Degree in Telecommunication Engineering
Academic Year (e.g. 2019-2020)

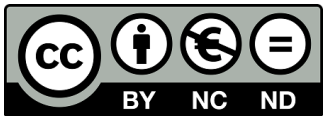
*Master Thesis*

# "Embedded solution for person identification and tracking with a robot"

---

Ignacio Condés Menchén

Fernando Díaz de María
Eduardo Perdices García
Leganés, 2020



# SUMMARY

This project describes the development process of an embedded system capable of performing a reactive following of a person. It makes use of convolutional neural networks and probabilistic tracking for processing the perception acquired by a RGB-D camera. This input is processed in a NVIDIA Jetson TX2, an embedded System-on-Module (SoM). This device is capable of performing computationally demanding tasks onboard, coping with the complexity required to run a robust tracking and following algorithm. The full design is implemented on a robotic mobile base, which receives velocity commands from the board, intended to move towards the desired person.

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

# DEDICATION

yes

# CONTENTS

# LIST OF FIGURES

# LIST OF TABLES

# 1. INTRODUCTION

## 1.1. Motivation

This work is focused on exploring the synergy between two science fields, which are outstanding nowadays: *robotics* and *deep learning*. These are combined for obtaining a robust system capable of following a certain person navigating towards it on a reactive behavioral. This behavioral is composed of two main components: the *perception block*, responsible of processing the images from an RGB-D sensor placed on the system, and the *actuation block*, which moves the robotic base accordingly to the relative position of the person to be followed.

The original idea was proposed on [1], where a neural following system was developed to be run in a standard laptop into which the camera and the robot were plugged. In the following dissertation, we will revisit this work and describe the points of interest which have allowed to enhance the previous version of this work.

The key aspects of this project can be brought in as follows:

**Embedded solution** the system is mounted on a battery-powered robot, on a *mobile base* form factor. This robot features a high-performance GPU embedded on a System-on-Module. Thus, this ensemble can work on its own, without requiring an external computer to perform the inferences or running algorithms in parallel. A remote monitoring of the behavioral is available as well, but it is not required for the system to work.

**Person identification** the proposed system runs 3 neural networks. These networks perform inferences over the images perceived by the RGB-D sensor, which is attached to the system as the *point-of-view* of the robot. The inferences are devoted to detect the different persons present in the scene, as well as to distinguish them by means of an distinguishing feature: their face.

**Tracking** the full system includes a probabilistic tracker, based on dynamic modeling. This leverages the *trajectories* followed by the persons while they wander on the visual field of the robot, as well as the relative distance to the person, obtained by the depth sensor included in the camera. As a result, we can have a gain on the robustness of the system, compared to a version governed exclusively by the neural inferences, which are sensitive to visual occlusions. Trusting just on these inferences could easily result on an unsteady behavioral. However, this can be avoided introducing the probabilistic modeling, as it will be explained later.

## 1.2. State of the art

As it was previously introduced, this work is performed to explore the synergies on robotics and deep learning. In this section, the current approaches and tools will be studied in order to outline a general framework where this work can be placed.

The problem to be addressed is to *get a robot to follow a person*. This problem can be split into several steps, where different approaches have been previously developed. The steps will be covered in the following subsections.

### 1.2.1. Person detection

Last decades have come linked to strong advances in the computer vision field. This has vastly reduced the production prices of digital cameras and high-resolution sensors, which have come into the consumer market segment: nowadays, everybody carries at least 2 cameras in their mobile phone. This, besides an increase in the hardware performance has resulted in a strong impulse into computer vision research. There are many application possibilities using cameras, however the *person detection* challenge is a recurrent one.

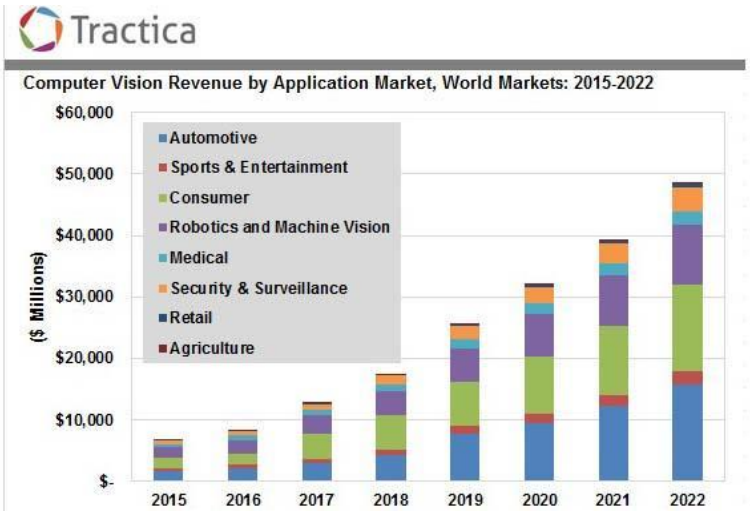



Fig. 1.1. Computer Vision revenues in the last year, and forecast for 2022 (source: [2]).

One of the mostly used approaches is commonly called the *Viola-Jones* detector [3]. This algorithm relies on a *rigid body model*, which fits a specific shape. On a grayscale image, this shape can be typically distinguished by means of the pixel intensity levels. A spatial filters called *Haar features* (Figure 1.2) are introduced: these are used across the image looking for the intensity pattern for each mask, which should resemble a part of the rigid body. As this presents a weak decision by itself, several filters (previously chosen in a training process) are combined on a *boosted cascade*. A person is detected if the weighted combination of several filters are triggered inside a certain area, which is

decided to potentially contain a person [4].

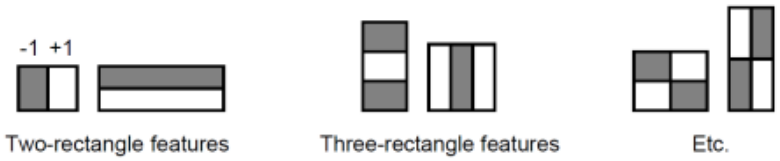



Fig. 1.2. Haar features: some examples [4].

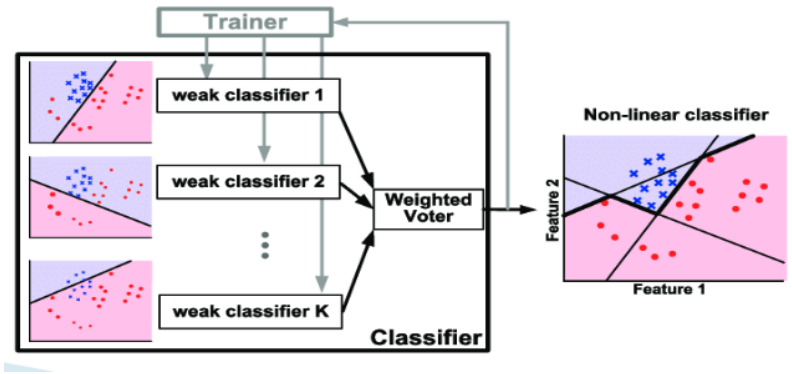



Fig. 1.3. Boosted weak classifiers [4].

Although this system was originally developed to detect faces, the rigid body model allows a generalization powerful enough to extend this to another object classes, *person* among these. The open-source standard image processing library, OpenCV, includes pre-trained models[1], which can be directly used in their Viola-Jones implementation. Scale invariance can be achieved evaluating the image at multiple scales on runtime.

Another common approach nowadays for person detection is based on HoG (*Histograms of Gradients*) [5]. This method computes local features by means of the intensity gradients across the image, and quantizes them using their angle (creating a histogram for the oriented gradients for that pixel), as it can be seen on Figure 1.4.

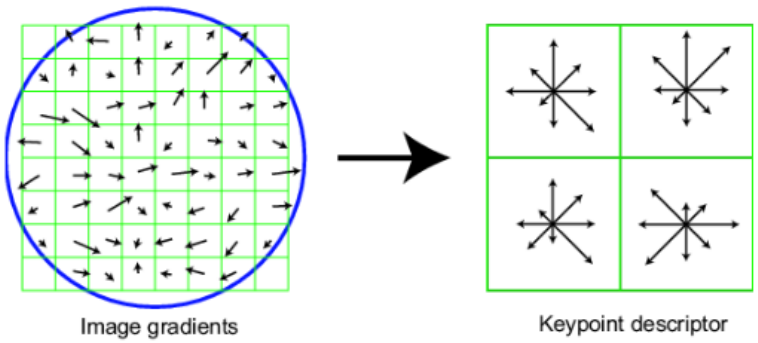



Fig. 1.4. Example of a HoG, quantized to 8 directions [6].

[1] https://github.com/opencv/opencv/blob/master/data/haarcascades

These gradients are collected in $64 \times 128$ windows, and treated as features from which a linear SVM (*Support Vector Machine*) is trained in order to classify a region as *person/non-person*. Figure 1.5 shows the average gradient patch for a person (the direction of each gradient is not shown). A visual inspection immediately resembles the shape of a person standing up. Thus, this detector will yield the best performance when the person to be detected stands in that specific pose.

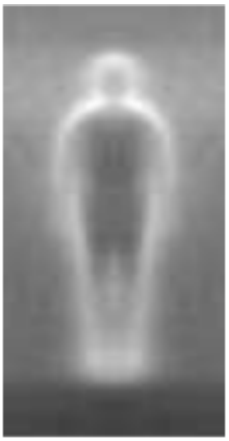


Fig. 1.5. Average gradient for person detection on [5].

These methods, among several more approaches, have been the state-of-the-art techniques: the cornerstone are the image gradients, which can be computed with a high efficiency and present decent performance. However, their main drawback is the *generalization* capability, as a successful detection is highly dependent on the person pose. However, in the latest advances, the detection frameworks have moved towards the spreading framework: *deep learning*, especially the most salient tools on image processing: CNNs (*convolutional neural networks*). The first steps on this field [7] required for a previous step on the image called *region proposal*. This step is devoted to find potential regions on the image to contain an object. This way, the challenge was to label these region according to the objects contained inside, reducing the problem to a classification task. However, the process to find these undetermined regions and iterate over them makes the process too slow for real-time requirements, which are explicitly contained in our objectives. Care has been put in posterior works [8] [9] to reduce this computation time. However, this reduction in time brings about a reduction in precision as well.

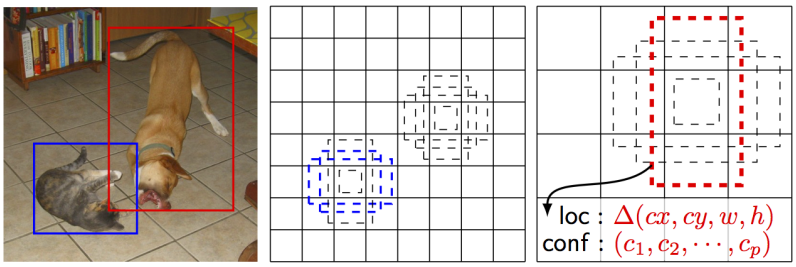
On the other hand, one of the most remarkable architectures is SSD (*Single-Shot Multibox Detector*) [10]. The main benefit from this architecture is the fact that it embeds all the required computations in a single neural network, reducing the complexity compared to other approaches requiring external region proposals, as it was depicted above. This greatly reduces the computational time when the network has to evaluate an image. As it was depicted in [1], the architecture is split into stages:

**Reshape** the posterior stages evaluate the image on a fixed tensor size of $n \times 300 \times 300 \times 3$ (being $n$ the size of the input batch). Other image sizes might be used, however this one offers a good trade-off between score and computational load.

**Base network** this first group of layers are reused from a typical image classification model, such as VGG-16 [11]. The first layer from this architecture are utilized in this design, truncated before the first classification layer. This way, the network can leverage the *feature maps* from the classification network, in order to find objects inside the input image. At the output of this network, several convolutional layers are appended, decreasing in size. This has the objective of predict detections at multiple scales. One thing to mention at this point is that the base network can be a different one rather than VGG-16, such as a MobileNet [12], which is highly optimized for running on low specifications devices. This is interesting as our embedded system will be limited in computing power, thus it will be revisited in future sections.

**Box predictors** later, for each extracted set, a dedicated operation is performed, generating a small set (typically 3 or 4) fixed-size *anchors*, with varying aspect ratios for each cell on a grid over the activation map (Figure 1.6). As these maps have different sizes, this aims to detect objects in different scales. The anchors are then convolved with small filters (one per depth channel), which output *softmaxed confidence values for each known class*, and *offsets/adjustments for the generated bounding box*. So, for each detected object (on that scale), we know the score for each class and its estimated position inside the feature map (hence, in the image as well).

*Postprocessor*: as several detections might be triggered in the same area for different classes and scales, a *Non-Maximum-Supression* operation is performed at the output of the network to retain the boxes with a larger area.




(a) Image with GT boxes (b) $8 \times 8$ feature map (c) $4 \times 4$ feature map

Fig. 1.6. A set of boxes are generated centered on each point of every feature map [10].

Another interesting approach on the neural networks field is the YOLO (*You Only Look Once*) system [13], which has been iteratively revised in two occasions so far [14] [15].

The latest implementation, YOLOv3 [15] features residual networks [16], which tackle the problem of *vanishing gradients* when the networks become deeper. The stacking of

several layers results on gradients diminishing its value up to a point the precision mode of the machine is not able to handle. The gradients are canceled, burdening the training process, as the first layers parameters take a substantially higher time to converge. The residual networks added in this revision of the design add shortcut connections across the layers, centering the backpropagation gradients on 1. As the publication says [15], the combination of these residual layers and convolutional ones allows to train much deeper architectures, capable of yielding a higher generalization. As in the SSD detectors, the YOLO architecture performs multi-scale detections, using 3 scales for splitting the feature maps into cell grids. On each of these cells, 3 anchor bounding boxes are fit. These anchors are selected by running the k-means algorithm on the COCO datasets selecting 9 clusters (3 anchor shapes $\times$ 3 scales). This aims to a better generalization as well, as in the R-CNN [7] and the SSD [10] the anchor shapes are hand-picked.

For each (anchor, cell, scale) combination, this network predicts:

- The coordinates of the object inside the anchor. Details can be visualized on Figure 1.7.
- *objectness* score, which is computed by means of a logistic regression in order to determine the probability of overlap with a ground truth bounding box more than any other prior anchor.
- 80 scores, as the original implementation is trained in the COCO dataset, which contains 80 classes. These classes might be overlapping (e.g. “woman" and “person"). Thus, these scores are computed by independent logistic classifiers and are not passed through a *softmax* operation.

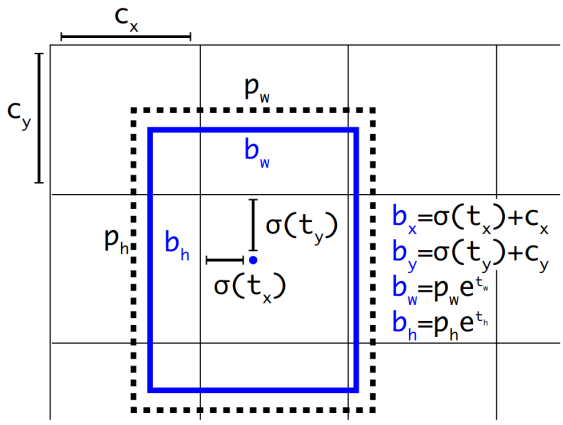



Fig. 1.7. Output on YOLO for each anchor and cell. The dashed line represents the prior anchor, while the blue line represents the detection which corrects that anchor.

### 1.2.2. Person identification

On a controlled environment, where the only present person is the one to be followed, a person detection system could be enough for following purposes. However, in a normal scenario, there might be several people inside the field of vision of the robot. This problem can be approached by means of adistinguishing feature of the person of interest, provided beforehand. One example is [17], which computes the color distribution of the person of interest, and later compares this distribution with the ones belonging to the different persons using the Bhattacharyya coefficient, which is designed to measure the similarity between the color histograms of the reference person and the detected one. However, this system can be deceived replicating the color distribution of the person of interest: wearing similar clothes helps to reduce the distance between the histogram, leaving a chance to confound another person with the one to follow.

A more robust approach is to use the *face* of the person as the distinguishing feature, as its uniqueness makes it a good reference to identify the detected person. Several approaches [18] extract facial *landmarks* from the morphology of a given face, and use them to classify the face, comparing it against a set of known faces and predicting the identity based on the distance to each known face. Some open-source libraries such as `dlib` and `OpenCV` provide the algorithms to perform these processes.

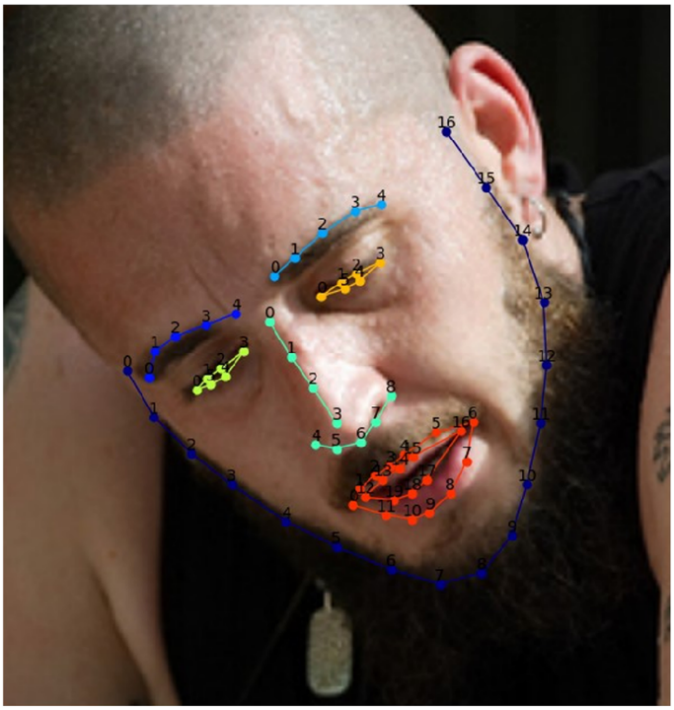


Fig. 1.8. Facial landmarks are dependent of the face shape and morphology (image from [18]).

The intuition behind these methods are to *project* the image of the face into a lower dimensional space, which allows to extract significant features from each face. These

features have to be consistent for the same face across different pose and lighting conditions (Figure 1.9). An useful transformation when a dimensionality reduction is pursued is PCA (*Principal Component Analysis*), a linear transformation that can be implemented to deal with the face recognition problem [19].

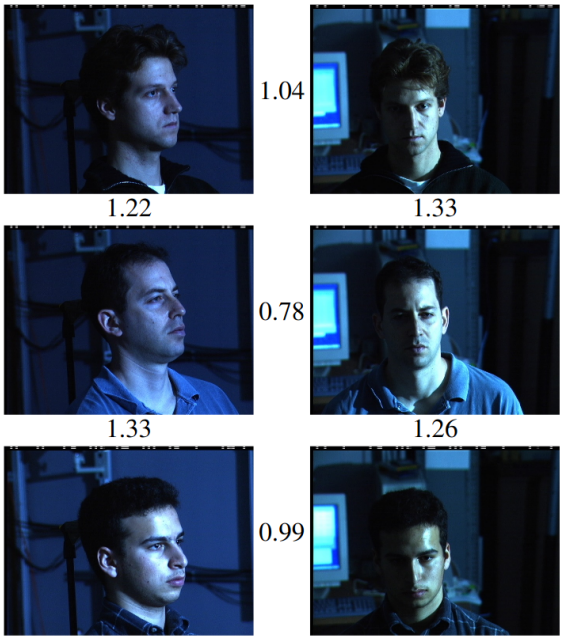



Fig. 1.9. Examples of poses and light conditions across which the face projections are desired to be consistent for the same person (image from [20]).

However, once again neural networks can be leveraged in order to achieve this and more: as the PCA is a linear operation, it could be learned by a single layer neural network. Thus, the introduction of deep networks can yield interesting results. The most relevant approach so far uses deep convolutional networks for performing this process [20], implementing an architecture called *FaceNet*, which is partially based on the Inception [21] module, designed by Google researchers in order to greatly reduce the number of parameters in a neural network. What this network computes is called an *embedding*, a projection of the input face image into a point in a 128-dimensional hypersphere. This allows to translate the identification into linear algebra terms, such as *distance* between two faces, clustering and applying unsupervised algorithms in order to determine the identity of a trivial face, among a collection of known regions. These networks can be trained using a loss function called *triplet loss*, inspired by the work in [22]. Given a training sample (*anchor*), a *positive* example (same class than the anchor) and a *negative* example (different class than the anchor) are chosen, and the network is tuned to maximize the *anchor-negative* embeddings distance, and minimize at the same time the *anchor-positive* one (Figure 1.10).

One thing to mention about the algorithms described above is that they perform the operations on the image of a face. Thus, a face detection system is required for previously cropping the face of the person to be identified. Once again, a neural approach can be


Fig. 1.10. Triplet loss training. It minimizes the distance between an *anchor* (current example) and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity (from [20]).

reduced to an *object detection* problem (detecting the class *face*, in this case). One interesting approach using this technique is *faced* [23]. This is a custom small ensemble of two neural networks, responsible to detect faces and correct the bounding boxes found. The main objective of the system is *speed*, so the main detector architecture is based in YOLO [13], and the second correction stage raises the precision achieved by the detector.

Fig. 1.11. Classical Haar based face detector [3] (left) vs. *faced* (right). Image from [23].

### 1.2.3. Embedded deployment

One of the requirements of this work is to be integrated in an autonomous system. This imposes a power limitation on the algorithms to be deployed. Generally, the robotic systems are deployed using laptops connected to robots, at it was done in [1].

Nowadays, the mentioned increase in the interest into the real-time computer vision applications has fostered the development of specific low-power embedded devices to be integrated in mobile systems. The extending usage of devices such as Arduino or Raspberry Pi has led to embedded robotics systems, such as PiBot [24] (Figure 1.13). These robots are useful in the educational scope, as they are capable of running simple vision and navigation algorithms at a low cost. Unfortunately, the requirements on systems running more complex algorithms, such as neural networks, require of the next tier in power terms, keeping the portability nevertheless. The ideal device could be an ASIC, as the custom design would lead to a very tight optimization of the performance. However, we are aiming to run the algorithms on existing software frameworks, so we aim to general purpose computers instead. The most remarkable advance in this scope are the Jetson devices manufactured by NVIDIA. These development boards are SoM (*System-on-Module*) computers running a tailored version of Linux. The fundamental feature of

(a) Frontal view.

(b) Side view.

Fig. 1.12. Laptop+robot deployment on [1].

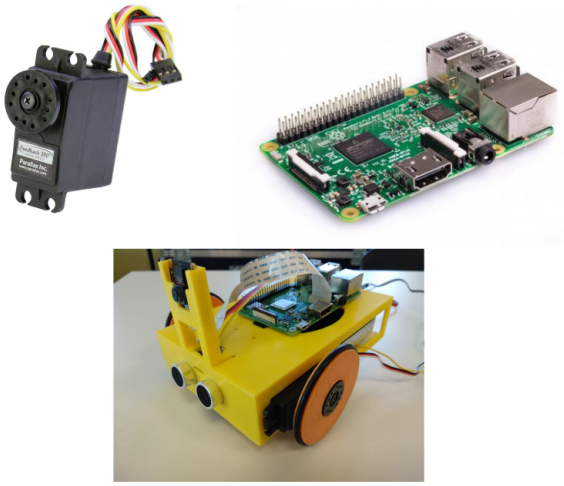


Fig. 1.13. PiBot, an open low-cost robotic platform for education (image from [24]).

these systems is that they include a high-performance GPU featuring CUDA, a low-level parallel computation library, as well as several toolkits designed to optimize as much as possible the software implementations for the plethora of possibilities to be designed on this board. As it can be seen in Figure 1.14, its size and power consumption make of this system a good choice to be included in an autonomous robotic system.

### 1.2.4. Person following

Several approaches have been developing pursuing this challenge of *following a person*. Once the perception algorithms are established, the final output of the pipeline has to be a movement command for the robot to move towards the desired point. Mobile robots can be classified according to their locomotion capabilities. A good summary can be that a robot is *holonomic* if the number of its controllable degrees of freedom is equal to its

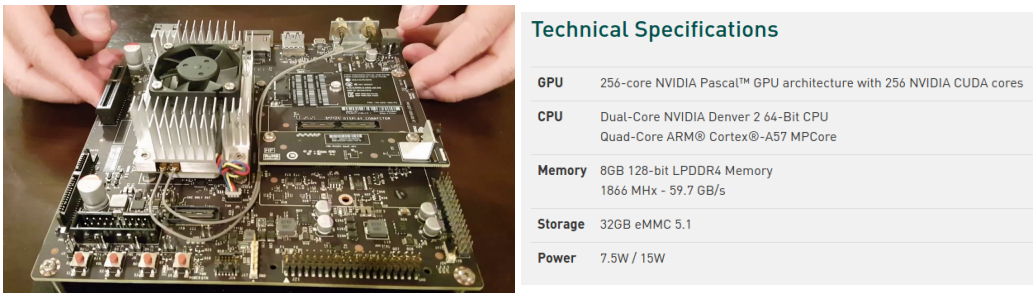



Fig. 1.14. NVIDIA Jetson TX2: an embedded high-performance device including a GPU.

total degrees of freedom. If the controllable degrees of freedom are lower than the total degrees of freedom, the robot is *non-holonomic*.

In the case of a holonomic robot, the navigation process is simplified, as the robot can instantaneously move to a desired target. However, a non-holonomic robot needs to perform maneuvers in order to move towards a point.

## 1.3. Objectives

This work has been carried out in order to fulfill certain requirements in a particular person following application:

1. Achieve a real-time following behavioral using embedded low-power hardware and a low-complexity educational robot.
2. Build the inference pipeline using exclusively concurrent CNNs (*convolutional neural networks*).
3. Combine a neural system with probabilistic filtering to carry out a robust multi-modal tracking of the persons in front of the robot. This will provide the system with extra endurance and robustness against detection losses/occlusions.

These objectives allow to summarize the starting point for the development of this project: the available materials are an educational robot equipped with a battery, an embedded *SoM* and a RGB-D sensor.

The result will be an autonomous robot which will follow a specific person, whose face has to be known beforehand (using a *reference face* image).

# BIBLIOGRAPHY


[1] I. Condés and J. 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. DOI: 10.1007/978-3-319-99885-5_11.

[2] *Computer Vision Market to Reach $ 48.6 Billion by 2022*, https://bitrefine.group/11-blog/120-establishing-your-brand-on-college-campuses, Accessed: 2020-06-07.

[3] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," vol. 1, Feb. 2001, pp. I–511. DOI: 10.1109/CVPR.2001.990517.

[4] I. González-Díaz, "Computer Vision: Image classification," University Lecture, 2020.

[5] 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)*, vol. 1, 2005, 886–893 vol. 1.

[6] I. González-Díaz, "Computer Vision: Local Invariant Features," University Lecture, 2020.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, *Rich feature hierarchies for accurate object detection and semantic segmentation*, 2013. arXiv: 1311.2524 [cs.CV].

[8] R. Girshick, *Fast r-cnn*, 2015. arXiv: 1504.08083 [cs.CV].

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *Lecture Notes in Computer Science*, pp. 346–361, 2014. DOI: 10.1007/978-3-319-10578-9_23. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10578-9_23.

[10] W. Liu *et al.*, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, pp. 21–37, 2016. DOI: 10.1007/978-3-319-46448-0_2. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[11] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2014. arXiv: 1409.1556 [cs.CV].

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

[13] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, 2015. arXiv: 1506.02640 [cs.CV].

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

[15] ——, *YOLOv3: An Incremental Improvement*, 2018. arXiv: 1804.02767 [cs.CV].

[16] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, 2015. arXiv: 1512.03385 [cs.CV].

[17] P. Li, H. Wu, and Q. Chen, "Color distinctiveness feature for person identification without face information," *Procedia Computer Science*, vol. 60, pp. 1809–1816, Dec. 2015. DOI: 10.1016/j.procs.2015.08.291.

[18] B. Johnston and P. Chazal, "A review of image-based automatic facial landmark identification techniques," *EURASIP Journal on Image and Video Processing*, vol. 2018, p. 86, Sep. 2018. DOI: 10.1186/s13640-018-0324-4.

[19] R. Gottumukkal and V. Asari, "An improved face recognition technique based on modular pca approach," *Pattern Recognition Letters*, vol. 25, pp. 429–436, Mar. 2004. DOI: 10.1016/j.patrec.2003.11.005.

[20] 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. DOI: 10.1109/cvpr.2015.7298682. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2015.7298682.

[21] C. Szegedy *et al.*, *Going deeper with convolutions*, 2014. arXiv: 1409.4842 [cs.CV].

[22] K. Q. Weinberger, J. Blitzer, and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," in *In NIPS*, MIT Press, 2006.

[23] I. Itzcovich, *faced: CPU Real Time face detection using Deep Learning*, towardsdatascience.com, Ed., [Online; consulted 9-June-2020], Sep. 2018. [Online]. Available: hhttps://towardsdatascience.com/faced-cpu-real-time-face-detection-using-deep-learning-1488681c1602.

[24] J. Vega and J. Cañas, *PiBot: An Open Low-Cost Robotic Platform With Camera for STEM Education*, Oct. 2018. DOI: 10.20944/preprints201810.0372.v1.