uc3m | Universidad Carlos III de Madrid

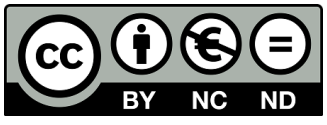
Master Degree in Telecommunication Engineering
Academic Year (e.g. 2019-2020)

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

Last decades, the production prices of digital cameras and high-resolution sensors has greatly reduced, bringing these devices into the consumer market segment: nowadays, everybody carries at least 2 cameras in their mobile phone, aside of high-quality web cameras, or even driving-assistance cameras in cars. This, beside an increase in the hardware performance has resulted in a strong impulse into computer vision research (Figure 1.1): there are many possibilities out of industrial environments for applications using cameras, such as fancy image modifications, or autonomous driving, as it can be seen on Figure 1.2.

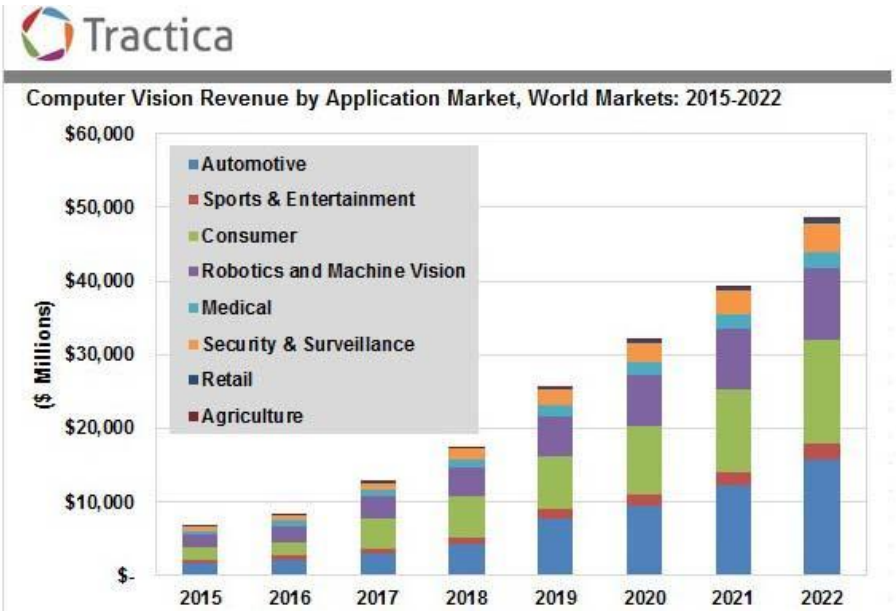


Fig. 1.1. Computer Vision revenues in the last year, and forecast for 2022 (source: [1]).

(a) Modifications of a subject on a portrait, such as apparent gender, or age.

(b) Autonomous driving on a Tesla Model X.

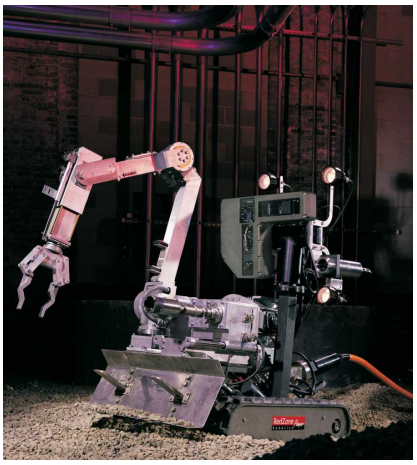
Fig. 1.2. Examples of contemporary computer-vision applications.

On the other hand, robotics applications can be really useful at daily tasks. These tasks are of greater interest when the behavioral of a robot tends to emulate the human

one[1], with the advantage of no human people exposed to a significant risk, or, in a less gloomy scenario, without human body physical limitations. This requires a polished (and somehow complex) behavioral, which is triggered by a certain input. At this point, we can find two main branches into robots, depending on the input source:

Teleoperated robots this kind of robots are capable of perform certain actions, which are *remotely controlled by a human operator*. This application is the one with most weight on the hazardousness (Figure 1.3a) [2] or precision [3] factor. Thus, some advances are made nowadays improving the teleoperation function, implementing *feedback* from the robot, such as haptic feedback [4], or VR (*Virtual Reality*) sensation, to allow that person to sense the environment as if she was in the robot position.

Autonomous robots these robots are much more complex machines, as they are distinguished for implementing a response by itself, independently of any kind of remote operator. This is seeked on certain scenarios, where there are some factors (as the time elapsed performing an action, or the cost of a control link with the robot) with a considerable weight in the design [5]. This is the kind of robots that concern us on this work: the state-of-the-art techniques try to emulate *human behavioral* (Figure 1.3b), so some actions can begin to be performed with a certain intelligence, as we will describe below.



(a) Pioneer robot, designed to perform hazardous teleoperated explorations in a deadly radioactive environment.

(b) Pepper, an autonomous humanoid capable of performing on-board processing and reacting to external stimuli intelligently.

Fig. 1.3. Example of a teleoperated and an autonomous robot.

The important advances on the last decades on the image processing and audio recognition fields have impulsed the development of assistance systems, apart from critical

[1] Some efforts are taken even into adopting the performance of human's best friend

machines as the previously described examples.

There are outstanding synergies between these two science fields, as it is explored on the system proposed in this work: these fields 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 image 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.

This work revisits the system proposed on [6], 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, this work will be revisited and describe the points of interest which have allowed to enhance the previous version of this work.

The key aspects of this solution 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. In contrast to the previous work, this assembly 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. In addition, specific optimization engines allow the system to run faster with 3 neural networks than previously with only 2 networks, on a low-consumption hardware. This will be reviewed later.

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 sensing source 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. Unlike the previous development, all the detection and identification tasks are based on neural networks, as it will be depicted later.

Tracking the full system includes a person tracker, based on optical flow. This aims to guess the trajectories followed by each person that the robot can see. As opposed to the previous work, this tracker allows to roughly follow the persons while the neural network yields a new update, as the tracker takes considerably less time to infer the person displacement. 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 as well. Trusting just on these inferences could easily result on an unsteady behavioral. However, the introduction of the

tracker softens the robot movements ensuring a more robust following, 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 depicted 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 proposed. These steps will be covered in the following subsections.

1.2.1. Person detection

One of the mostly used approaches is commonly called the *Viola-Jones* detector [7]. 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 1.4) 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 [8].

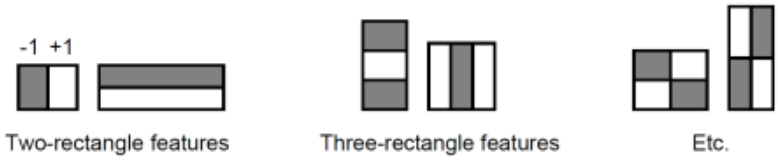


Fig. 1.4. Haar features: some examples [8].

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 pretrained models[2], 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*) [9]. This method computes local features by means of the intensity gradients across the image, and quantizes them using their angle (creating a histogram for

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

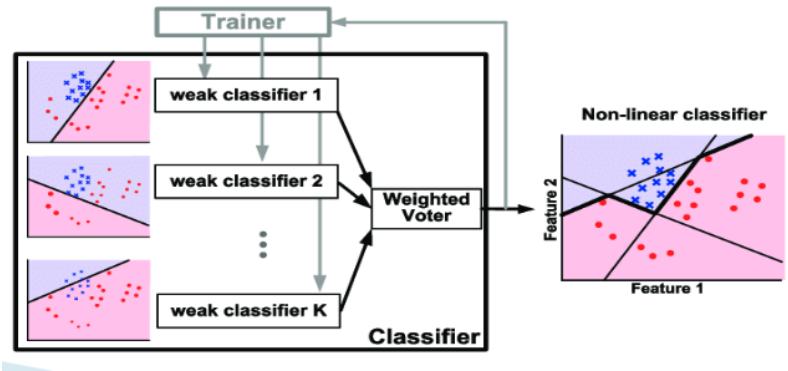


Fig. 1.5. Boosted weak classifiers [8].

the oriented gradients for that pixel), as it can be seen on Figure 1.6.

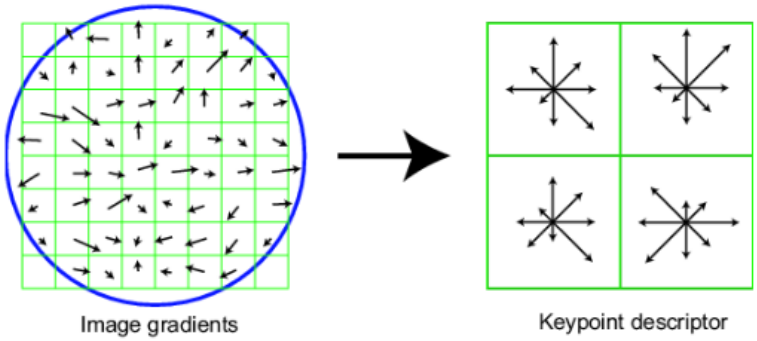


Fig. 1.6. Example of a HoG, quantized to 8 directions [10].

These gradients are collected in 64×128 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 1.7 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.

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*).

CNNs are based on standard neural networks, which combine lots of neurons or *perceptrons* organizing them into layers. These perceptrons (Figure 1.8a) implement non-linear operations, that allow to extract (after a proper training process) abstract features,

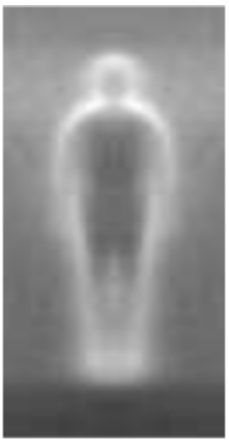
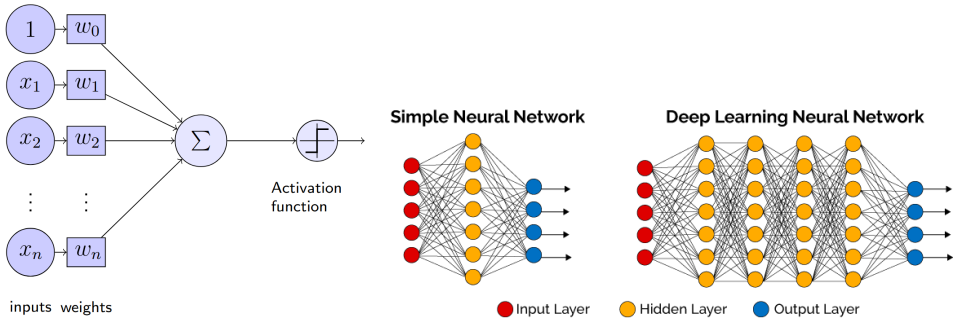


Fig. 1.7. Average gradient for person detection on [9].

which gain in complexity when the number of internal layers increase. When a neural network is composed by several *hidden* layers (in addition to the input/output ones), it is placed into the *deep learning* paradigm (Figure 1.8b).



(a) Basic unit of a neural network: the perceptron (source: [6]).

(b) Standard neural network vs. deep neural network (source: [6]).

Based on this system, and taking leverage on the *spatial correlation* when the signal to process is an image, a neural network can be modified to implement a different operation on each perceptron: the *image convolution* (Figure 1.9).

As it can be seen on Figure 1.10, convolutional units can be arranged conforming a set of layers used to build *feature extraction* stages (shown in red in the figure). Several layers can be concatenated, gaining in depth and obtaining more complex feature maps. These layers are finally followed by a detection/classification ensemble of *dense* layers (shown in blue in the figure): a set of layers with standard perceptrons fully connected among them, yielding a final output, dependent on the classification structure of the network.

In the case of an object detection network (the ones involved in this work), the output varies depending on the implementation, but it is generally composed of a set of (location, probability) tuples, one for each class the network is capable of detecting. Figure 1.11 shows the activation maps of an object detection network, where the map presents higher

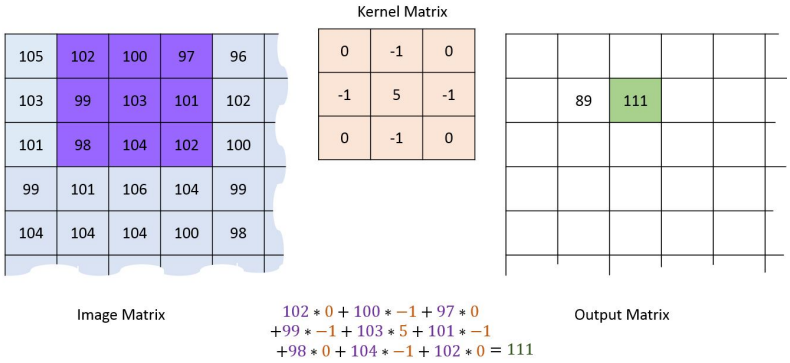


Fig. 1.9. Convolution applied on an image, applying the mask (red) on a region (purple) of the input image, storing the result on the mapping of the central pixel of the region (green). The computation is the sum weighted by the mask values (bottom) (source: [6]).

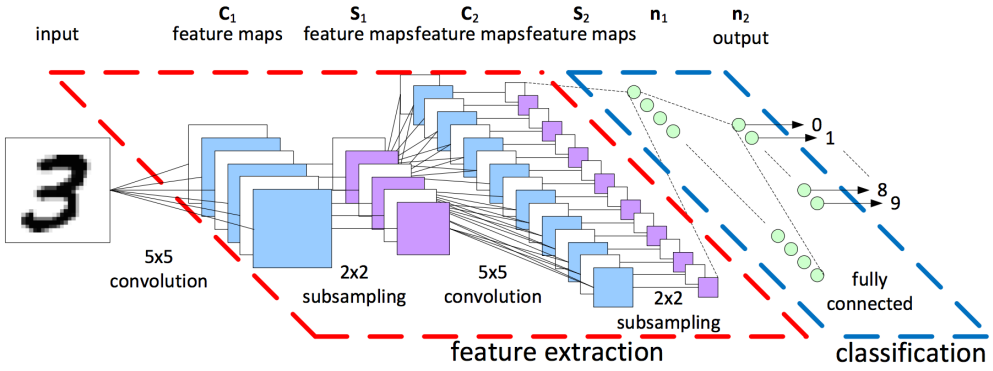


Fig. 1.10. Schematic of a digit classification CNN (source: [6]).

values in the regions with high probability of containing the object of the class it is designed for. Several of these maps compose each neuron on a CNN.

One possible application of this concept is focused on what is called *Region-based Convolutional Neural Networks* [11], which require 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 is 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 [12] [13] to reduce this computation time. However, this reduction in time brings about a reduction in precision as well.

SSD

Another of the most outstanding object detection architectures is SSD (*Single-Shot Multibox Detector*) [14]. 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

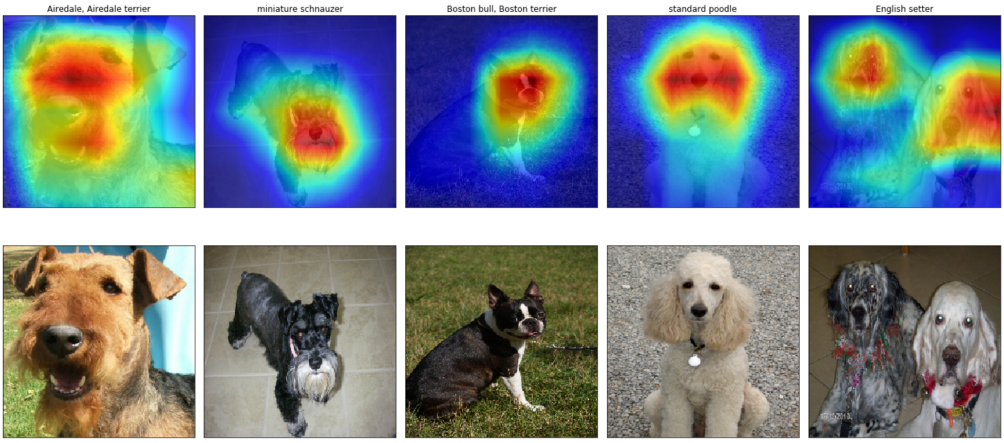


Fig. 1.11. Activation maps of a detection CNN searching for dogs on different images (source: [6]).

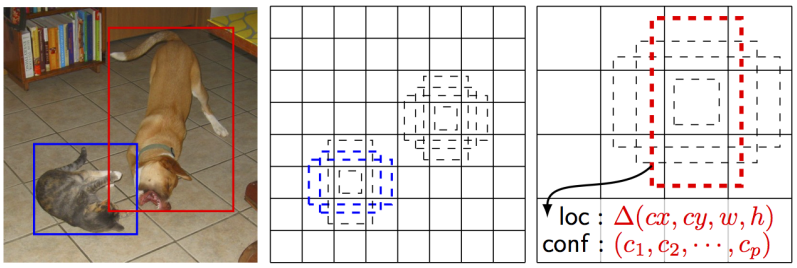
to other approaches requiring external region proposals, as it was depicted above. This greatly reduces the computational time when the network has to evaluate an image. The architecture can be seen at Figure 1.17, and can b split into stages [6]:

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 [15]. 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 [16], 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 1.12). 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* [17] operation is performed at the output of the network to retain the best boxes, under a combined criteria of detection score and IoU score (*Intersection over Union*, which measures the overlapping quality between two bounding boxes, as it can be seen in .



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Fig. 1.12. A set of boxes are generated centered on each point of every feature map [14].

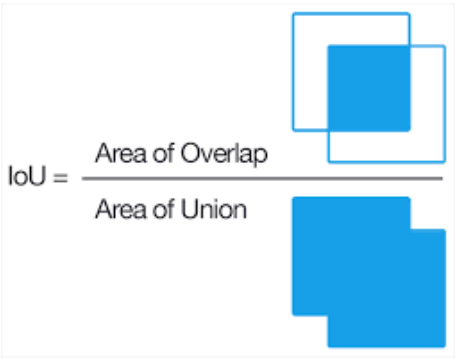


Fig. 1.13. Graphical representation of the IoU score between two bounding boxes.

YOLO (You Only Look Once)

Another interesting approach on the neural networks field is the YOLO (*You Only Look Once*) system [18]. Its main advantage is its inference speed, due to the fact that it performs a single analysis on the entire image, dividing it into a grid of cells. Each cell predicts up to 5 boxes, containing an *objectness score* (the predicted IoU of the proposal with an object, regardless its class), the coordinates of the bounding box, and a probability for the object belonging to each class. This design run faster than other methods [18], however it presents a poor performance when detecting small objects.

This design was revisited in YOLO9000 [19], introducing several improvements such as batch normalization on the input of the convolutional layers, or the concept of *anchor boxes*: the box proposals follow a fixed set of aspect ratios, chosen previously using clustering on a training set. As it can be seen on Figure 1.14, limiting the proposal shapes to 5 fixed sizes improves the performance while maintaining a high IoU metric. A visual

inspection show that the selected anchors seem like a reasonable shape for the majority of the objects the network aims to detect. Additionally, the number of deep layers was increased from 26 layers to 30, and a semantic modeling is performed on the labels across different datasets, allowing the network to be trained in different datasets under a common semantic structure called *WordTree* (Figure 1.15).

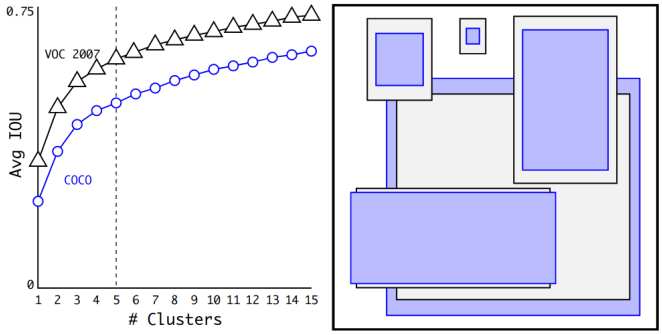


Fig. 1.14. Result of the anchor k-means clustering on VOC and COCO for YOLO9000. Using $k = 5$ anchor sizes on the right yields a good tradeoff between simplicity and improvement on the obtained IoU with respect to using $k - 1$ clusters (source: [19]).

The latest improvement of YOLO, YOLOv3 [20], features residual networks [21], 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 states [20], the combination of these residual layers and convolutional ones allows to train much deeper architectures (53 convolutional layers), 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. A similar k-means than in Figure 1.14 is performed on the COCO dataset, selecting 9 anchor sizes instead of 5, and grouping them in 3 scales. Now, on each of the cells, 9 anchor bounding boxes are fit (3 anchor shapes $\times$ 3 scales). This aims to solve the poor performance of the previous version when dealing with small objects, as well as a better generalization: in the R-CNN [11] and the SSD [14] the anchor shapes are hand-picked. These changes, with a tuning on the error function

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

- The coordinates of the object inside the anchor. Details can be visualized on Figure 1.16.
- *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.

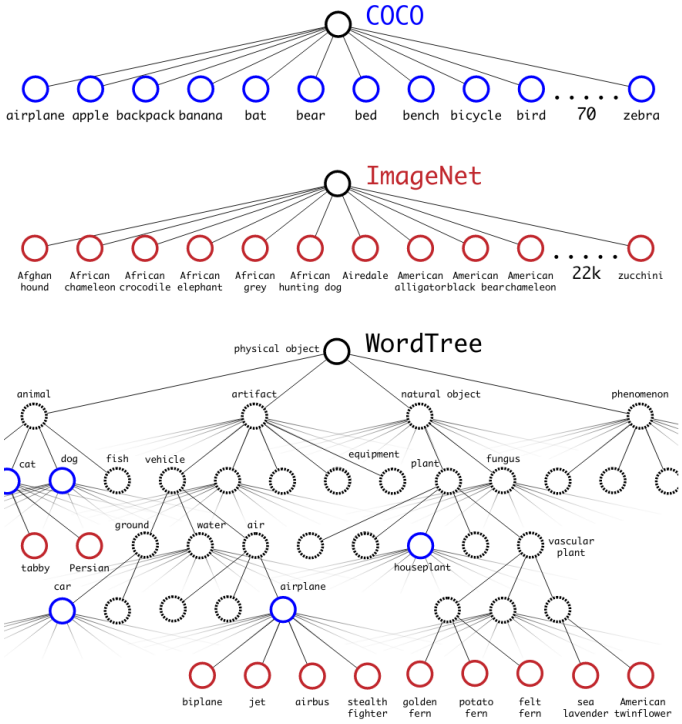


Fig. 1.15. Comparison between simple labeling structures (top) and a WordTree semantic grouping under categories. This allows to follow a dataset-agnostic training process as the labels can be combined using WordTree.

- 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.

The architecture of a YOLO-based detection network can be seen beside a SSD-based one in Figure 1.17. This allows to see the fundamental difference, in the feature extraction stage of each implementation.

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 a distinguishing feature of the person of interest, provided beforehand. One example is [22], 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 [23], a measurement of similarity between two probability distributions. This metric can be applied to measuring 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

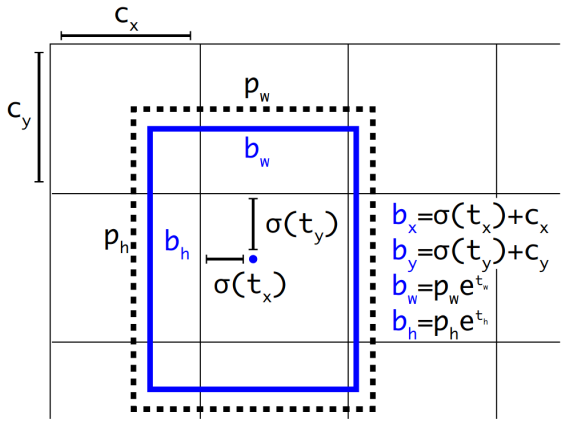


Fig. 1.16. Output on YOLO for each anchor and cell. The dashed line represents the prior anchor, while the blue line represents the detection which corrects that anchor.

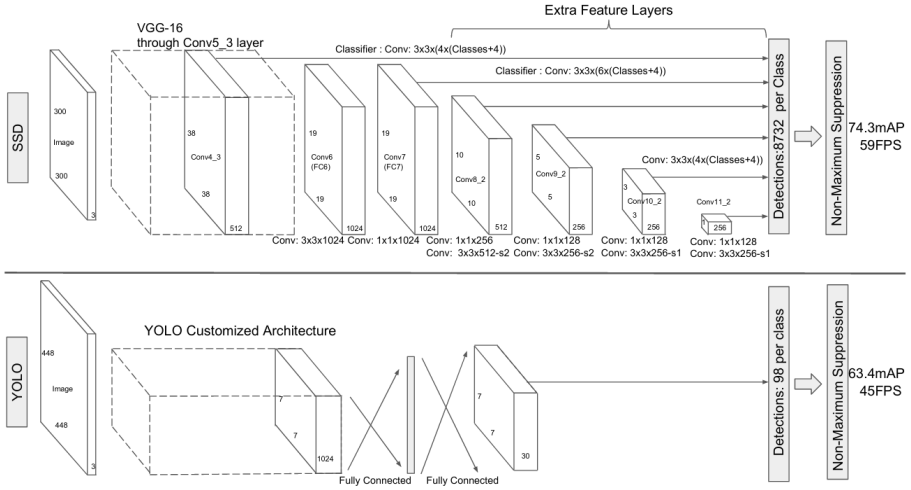


Fig. 1.17. General architecture of a SSD network (top) and a YOLO one (bottom).

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. As it is summarized in [24], several applications 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.

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 1.19). An useful transformation when a dimensionality reduction is pursued is PCA (*Principal Component Analysis*), a linear transformation that can be implemented to

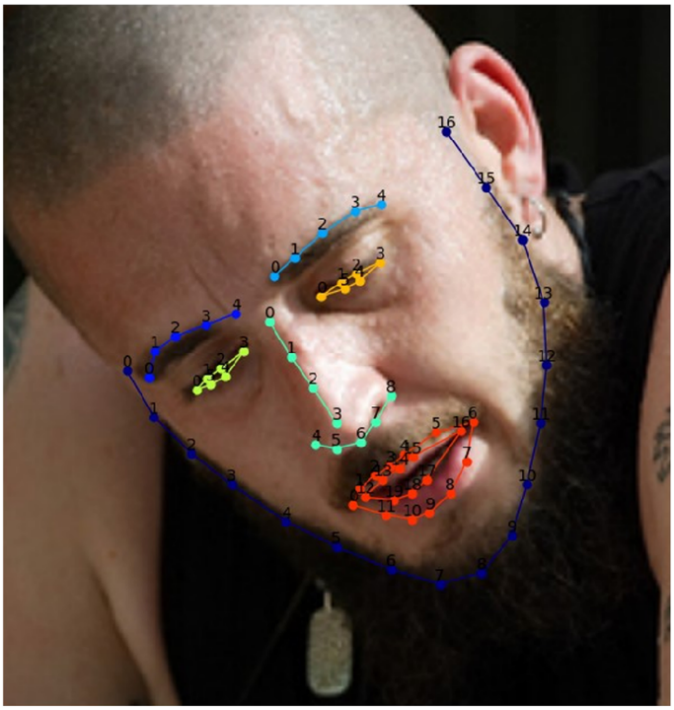


Fig. 1.18. Facial landmarks are dependent of the face shape and morphology (image from [24]).

deal with the face recognition problem [25].

Deep learning face identification: FaceNet

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 [26], implementing an architecture called *FaceNet*, which is partially based on the Inception [27] 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. The architecture can be visualized in Figure 1.20. These networks can be trained using a loss function called *triplet loss*, inspired by the work in [28]. 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 1.21).

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

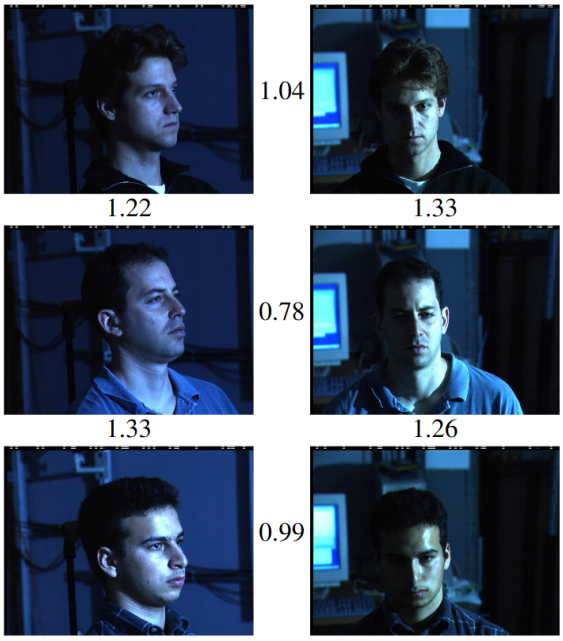


Fig. 1.19. Examples of poses and light conditions across which the face projections are desired to be consistent for the same person (image from [26]).

Fig. 1.20. Architecture of the FaceNet system (from [26]).

reduced to an *object detection* problem (detecting the class *face*, in this case). One interesting approach using this technique is *faced* [29]. 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 [18], and the second correction stage raises the precision achieved by the detector.

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, as it was done in [6].

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 [30] (Figure 1.24). 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

Fig. 1.21. 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 [26]).

Fig. 1.22. Classical Haar based face detector [7] (left) vs. *faced* (right). Image from [29].

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 these systems is that they include a high-performance GPU featuring CUDA, a low-level parallel computation library, as well as several toolkits (such as TensorRT[3]) 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 1.25, 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 total degrees of freedom. If the controllable degrees of freedom are lower than the total degrees of freedom, the robot is *non-holonomic*. This difference can be observed on Figure 1.26. In the case of a holonomic robot, the navigation process is simplified, as the robot can

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

(a) Frontal view.

(b) Side view.

Fig. 1.23. Laptop+robot deployment on [6].

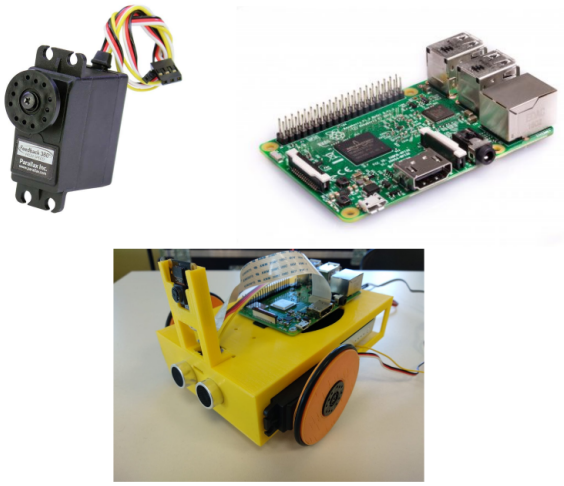


Fig. 1.24. PiBot, an open low-cost robotic platform for education (image from [30]).

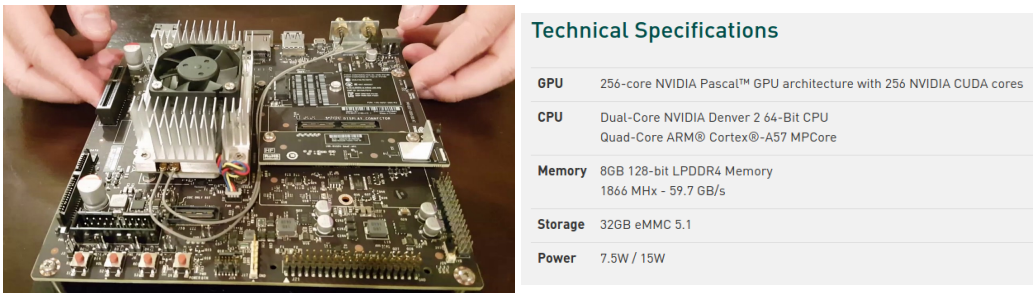
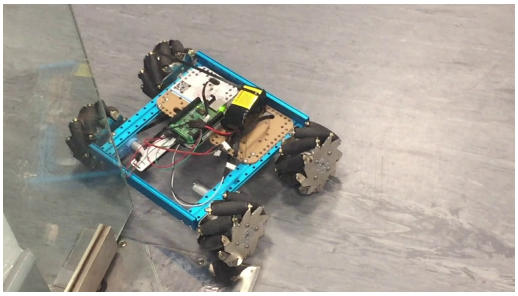


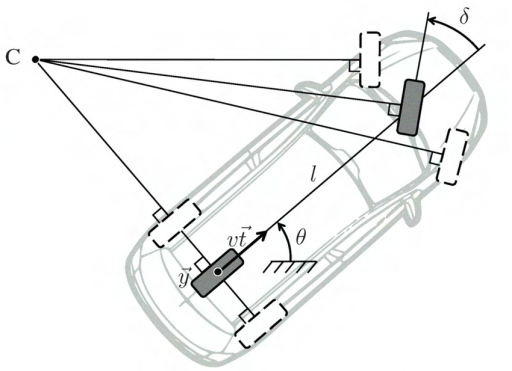
Fig. 1.25. NVIDIA Jetson TX2: an embedded high-performance device including a GPU.

instantaneously move to a desired target. However, a non-holonomic robot needs to perform maneuvers in order to move towards a point.

The summary on [31] shows an interesting classification of some existing algorithms and their applications (Figure 1.27).



(a) Holonomic robot.



(b) Schematic of the degrees of freedom of a non-holonomic vehicle (a standard car).

Fig. 1.26. Comparison of a holonomic system with a non-holonomic one.

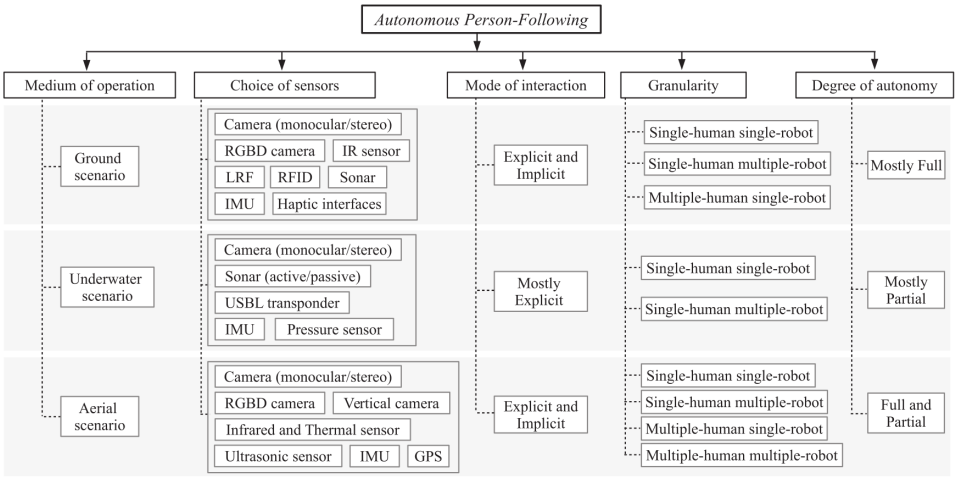


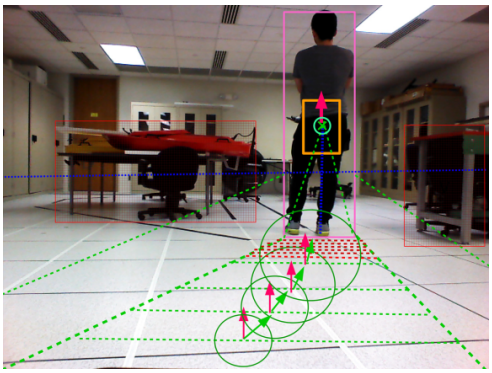
Fig. 1.27. In-depth classification of the existing person following algorithms (image from [31]).

Some approaches leverage the detected objects in order to estimate the relative homography of the orthogonal planes, which allows to partially know the environment of the robot and trace a safe path towards the person, as it can be seen on Figure 1.28a.

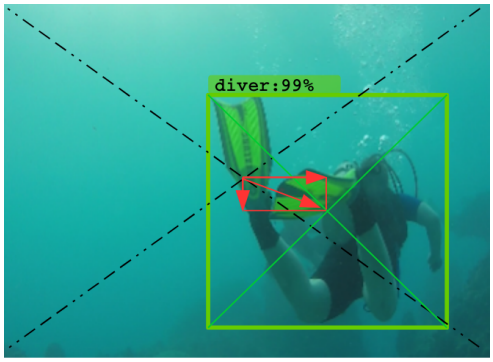
Other approaches act without a path planning component, implementing what is called a *reactive* behavioral [32], similar to the solution proposed on this work. On these approaches, the vector between the center of the image and the center of the person is used to command movements on the robot, as it can be seen on Figure 1.28b.

1.3. Objectives

The main objective of this work is to design and develop an embedded system which allows a low-cost robot to follow a certain person on a robust way. 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). This objective, in turn, can be split into specific



(a) Following with path computation using homographies (image from [31]).



(b) Example of underwater reactive following (image from [32]).

requirements:

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*), as they offer robustness on detection under harsh lighting conditions, such as the ones observed in Figure 1.29.

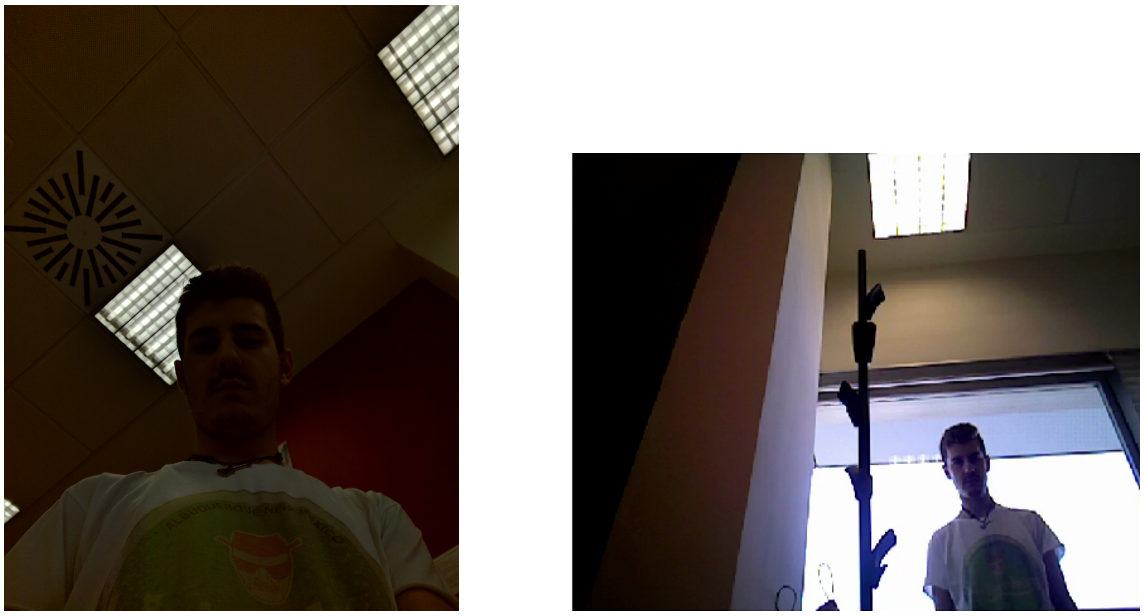


Fig. 1.29. Poor lighting situations for a low-positioned camera.

3. Combine a neural system with optical tracking to carry out a robust following 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.

BIBLIOGRAPHY

[1] *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.

[2] J. Potel, "Trial by fire: Teleoperated robot targets chernobyl," *IEEE Computer Graphics and Applications*, 1998. [Online]. Available: https://www.computer.org/csdl/mags/cg/1998/04/mcg1998040010.pdf.

[3] P. Berkelman and J. Ma, "The university of hawaii teleoperated robotic surgery system," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2007, pp. 2565–2566. DOI: 10.1109/IROS.2007.4399550.

[4] A. M. Okamura, "Methods for haptic feedback in teleoperated robot-assisted surgery," *Ind Rob*, vol. 31, no. 6, pp. 499–508, Dec. 2004, 16429611[pmid]. DOI: 10.1108/01439910410566362. [Online]. Available: http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1317565/.

[5] D. Girimonte and D. Izzo, "Artificial intelligence for space applications," in *Intelligent Computing Everywhere*, A. J. Schuster, Ed. London: Springer London, 2007, pp. 235–253. DOI: 10.1007/978-1-84628-943-9_12. [Online]. Available: https://doi.org/10.1007/978-1-84628-943-9_12.

[6] 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.

[7] 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.

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

[9] 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.

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

[11] 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].

[12] R. Girshick, *Fast R-CNN*, 2015. arXiv: 1504.08083 [cs.CV].

[13] 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.

[14] 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.

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

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

[17] J. Hosang, R. Benenson, and B. Schiele, *Learning non-maximum suppression*, 2017. arXiv: 1705.02950 [cs.CV].

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

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

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

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

[22] 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.

[23] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bull*, pp. 99–109, 1947.

[24] 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.

[25] 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.

[26] 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.

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

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

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

[30] 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.

[31] M. J. Islam, J. Hong, and J. Sattar, "Person-following by autonomous robots: A categorical overview," *The International Journal of Robotics Research*, vol. 38, no. 14, pp. 1581–1618, 2019. DOI: 10.1177/0278364919881683. eprint: https://doi.org/10.1177/0278364919881683. [Online]. Available: https://doi.org/10.1177/0278364919881683.

[32] M. J. Islam, M. Fulton, and J. Sattar, *Towards a generic diver-following algorithm: Balancing robustness and efficiency in deep visual detection*, 2018. arXiv: 1809.06849 [cs.RO].