Person following robot behavior using Deep Learning

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

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

Abstract. The proposed following behavior combines a *perception* and a *control* module. Perception module addresses the person detection on images using a pretrained TensorFlow SSD *Convolutional Neural Network*. It provides *robustness* over traditional methods, although care has to be put on not losing real-time operation. A *person tracker filter* has been included to alleviate the effect of false positives/negatives. It also extracts faces, which are also analyzed by a *FaceNet* CNN to reidentify the tracked individual. This perception module tells which person in the image has to be followed, even on poorly lightened scenarios. It is combined with depth readings to obtain the relative position of the person. The control module implements a case-based *PID* controller for a reactive smooth response, moving the robot towards the target person. The entire system has been experimentally validated[3] on a real Turtlebot2 robot, with an Asus Xtion RGBD camera.

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

1 Introduction

In this project, we aim to establish a scope which covers two fields: *robotics* and *deep learning*.

In the last decades, robots have become a powerful allied to humans, as they are leveraged to perform all kind of tasks (hazardous explorations, personal assistance, cleaning, driving, . . .). However, we aim to design them to perform these tasks on the most autonomous way possible, which means not requiring to be controlled on every action (with exceptions, such as surgeon robots). This requires to provide robots a certain intelligence and capabilities to correctly trigger the most suitable action for each possible input stimulus. For this purpose, we can find multiple approaches for different problems (navigation, conversation, . . .).

We will focus in the *Computer Vision* field, which involves connecting cameras to a robot, and taking advantage of this on an autonomous way. Concretely,

[3] https://jderobot.org/Naxvm-tfg

we will tackle the person following challenge, which is based on a behavior governed by a *person detection* system. Many approaches are already existing, such as color filters, or disparities filtering. However, promising *Deep Learning* techniques excel on their image processing variant, as it offers high quality results, accompanied by *robustness* facing lighting issues (Fig. 1). This is the set of techniques we have used on this project, so we will describe the basis firstly.

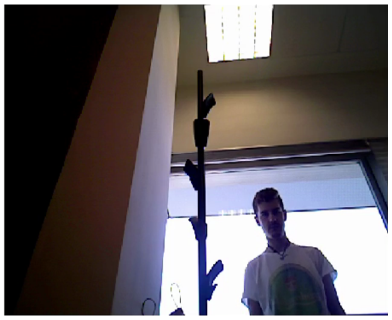

Fig. 1. Harsh lighting situation for *Computer Vision* algorithms.

1.1 Artificial Neural Networks (*ANNs*)

A Neural Network is the representation of an algebraic algorithm which implements non-linear calculus models [10 en tfg]. It is composed by successive processing *layers*, which are made up of *perceptrons* (generally called *neurons*). This is because these neural structures *emulate the human brain*, formed by a huge set of interconnected neurons, which are disposed on the already mentioned layers (Fig. 2)

1.2 Processing unit: the perceptron (neuron)

1.3 Deep Neural Networks

1.4 Convolutional Neural Networks

1.5 kaka

In the image detection quandary, we aim to determine *presence and position* of a certain object in a given image. For this, we will make use of *deep neural networks*, which are processing structures which try to emulate a biological brain.

They are composed by *perceptrons* (what are called *neurons*). These perceptrons implement an internal pipeline, which is applied to its input, what we call the *activation region.*

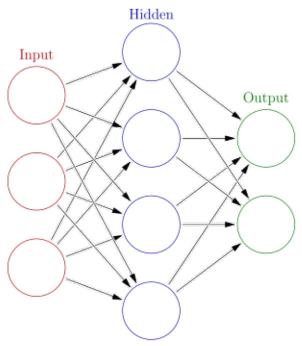

Fig. 2. Structure of a Neural Network

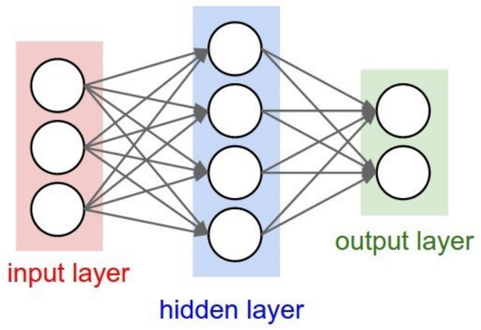

Fig. 3. Neural Network.

2 Infrastructure

On this article, we rely on a certain set of hardware/software elements:

Hardware

- *Asus Xtion Pro Live*: RGBD (RGB + Depth) sensor. It counts with a regular digital camera, and with an infrared emitter, which radiates IR beams. Their reflections are retrieved by a depth sensor, which generates a *point cloud* with the measured distance to each reflection surface. A *registration* process is performed in the driver, to project these points into the RGB pixels (due to the spacial offset between both sensors).

- *Turtlebot 2*: 2-wheeled robot, designed for didactic purposes. It counts with 2 freedom degrees on its motors: *angular* movements and *linear* movements.

Software

– *ROS*: robotics software development framework. It provides, among others, the low level drivers to get correctly communicated with the hardware devices.

– *JdeRobot*: research and educational robotics framework. The taken benefit from this framework is the higher level interfaces to communicate our developed program with the drivers created by ROS.

– *TensorFlow*: *deep learning* library recently developed by Google, and already widely adopted by big companies. It operates with *tensors*[4] on an optimized low-level way. It allows GPU parallelization of the operations, which leads to a very efficient implementation of *deep learning* algorithms.

– *Python*: high-level interpreted programming language. It is widely used on *machine learning* purposes due to its focus on easiness and *Object Orientation*. It counts with implementations of the previously described libraries and frameworks.

3 Perception Module

This is the first component of the developed system. As it name indicates, it is responsible for apprehend the incoming images from the sensor (RGB and depth) images, and generate a significant output to be interpreted by the actuation module. Its structure follows a certain pipeline along the images, which will be analyzed next.

3.1 Object Detection

On first place, the incoming RGB images are passed through an *object detection* module. This module is powered by a CNN, which has a special architecture designed for this purpose: SSD (*Single-Shot Multibox Detector*). This kind of detection technique stands out by its prediction speed, because *it performs a single feed-forward pass* of the image through the network, unlike the rest of *state-of-the-art* techniques (which are consequently much slower, AS CAN BE SEEN ON [tabla experimentos object detector]).

IMAGEN ESTRUCTURA RED (5.6 tfg) This neural architecture is formed putting together a succession of blocks:

1. *Preprocessing:* as this kind of neural networks work with a fixed image shape (300×300 px in this case, the most typical one among other SSD implementations).

[4] Generic data structure with a specific number of dimensions.

2. *Feature Extraction:* a first group of layers (*base network*) deals the *feature extraction* part. As it is composed by a concatenation of *convolutional* layers, the resulting activation maps are gradually smaller and deeper. So, 6 intermediate sets, composed by activation maps of a certain depth are extracted from this concatenation, with progressive sizes (suitable for different objects shapes). This way, we obtain the 6 activation maps reflected on the Table TABLA PAG 55 TFG.
3. *Box Prediction:* later, on a parallel way for each set, the activation maps are traversed, generating *prior* boxes on each point of them, with different aspect ratios. These *priors* are convolved with small filters, which output *adjustments* for each box, and *softmaxed* confidences for each of the classes that the network knows. IMAGEN PERRO GATO
4. *Postprocessing:* it combines the detections of each of the 6 sets, only retaining the most confident detections (*Non-Maximum Supression*), and adjusting and scaling the obtained bounding boxes to the original image shape.

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

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

However, our system only retains those detection corresponding to *persons*, as it is what we are interested to follow.

Hence, this results on a light *person detection*, perfectly capable to work on a real-time operation on a standard level hardware. Our implementation is capable to handle different network models and architectures on a *plug and play* model, just with the model file (in the specific TensorFlow `.pb` format).

3.2 Face detection

Once we have detected the existing persons on the image, we will look for their faces, in order to know whether any of them corresponds to the person to follow. To do so, we will perform *face reidentification*, which needs in first place to detect each person face. To do so, we implement the classical Viola and Jones [31 tfg] face detection algorithm. This algorithm, which comprises *Haar* features (Fig. –), is a simple algebraic method which takes advantage of the typical illumination pattern of a face (due to its physical shape) to detect promising regions of the input image to contain a face. This image is divided into *regions*, which are passed through a *cascade* of tests, where the non-compliant regions are immediately discarded. The accepted ones pass to a slightly more complex feature each time, and the regions which pass all the features are supposed to contain a face. This progressive region dismiss makes it an efficient algorithm, capable of run simultaneously with the rest of processes. This entire process is performed with an OpenCV[5] method.

As we already knew where the detected persons are in the image, this face detection process is only performed inside the persons bounding boxes.

3.3 Face reidentification (*FaceNet*)

If it is achieved to find

References

1. Smith, T.F., Waterman, M.S.: Identification of common molecular subsequences. J. Mol. Biol. 147, 195?197 (1981). doi:10.1016/0022-2836(81)90087-5
2. May, P., Ehrlich, H.-C., Steinke, T.: ZIB structure prediction pipeline: composing a complex biological workflow through web services. In: Nagel, W.E., Walter, W.V., Lehner, W. (eds.) Euro-Par 2006. LNCS, vol. 4128, pp. 1148?1158. Springer, Heidelberg (2006). doi:10.1007/11823285_121
3. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, San Francisco (1999)
4. Czajkowski, K., Fitzgerald, S., Foster, I., Kesselman, C.: Grid information services for distributed resource sharing. In: 10th IEEE International Symposium on High Performance Distributed Computing, pp. 181?184. IEEE Press, New York (2001). doi:10.1109/HPDC.2001.945188
5. Foster, I., Kesselman, C., Nick, J., Tuecke, S.: The physiology of the grid: an open grid services architecture for distributed systems integration. Technical report, Global Grid Forum (2002)
6. National Center for Biotechnology Information. http://www.ncbi.nlm.nih.gov

[5] Open source image processing library.