UNIVERSITÀ DEGLI STUDI DI TRENTO

Department of Information Engineering and Computer Science

Master's Degree in

Computer Science and Technologies

FINAL DISSERTATION

Integration of multiple deep learning algorithms for real-time single person long-term tracking over complex scenarios

Professor

Luigi Palopoli

Student

Stefano Leonardi

Supervisor

Stefano Divan

Supervisor

Fabiano Zenatti

Academic year 2019/2020

Ringraziamenti

...thanks to...

Contents

1 Sommario

Sommario è un breve riassunto del lavoro svolto dove si descrive l'obiettivo, l'oggetto della tesi, le metodologie e le tecniche usate, i dati elaborati e la spiegazione delle conclusioni alle quali siete arrivati.

Il sommario dell'elaborato consiste al massimo di 3 pagine e deve contenere le seguenti informazioni:

- contesto e motivazioni
- breve riassunto del problema affrontato
- tecniche utilizzate e/o sviluppate
- risultati raggiunti, sottolineando il contributo personale del laureando/a

2 Introduction

This chapter offers an overview of the project on which the thesis is based. The goal is to explain in detail how the practical problem has been approached in order to analyze the physical constraints, ideate a software method able to solve them, and how these ideas were then implemented into a working algorithm.

2.1 Physical context

The physical component in this project is a robot. Its definition can vary a lot respect to the context in which it is used. For this project, a robot can be described as a vehicle able to move in the space. A **LIDAR (Laser Imaging Detection and Ranging)** sensor is mounted. It is used to drive in the space avoiding hitting physical obstacles during the movement. In addition, it is installed a computational device connected to a webcam that can record streams of images of the space in front of the robot itself.
The video camera is the eyes of the robot itself, and the captured video stream is used as input of the algorithm working on the computational device. This computer can be both composed of a **CPU (Central Processing Unit)** but more often it is built with a **GPU (Graphics Processing Unit)** that can speed up parallelized computation, applied on the **DNNs (Deep Neural Networks)** used as the core of the algorithm. The software does not assume one component respect to the other, the only variation is in the performances: a GPU computation speed can be much higher than a CPU.
Instead, the output of the algorithm is a position composed of X and Y coordinates respect to a single frame captured from the webcam. This location can be then elaborated and, with the use of LIDAR sensor, the robot can estimate which is the 3D position of the element tracked from the software.
Finally, it moves to reach that position, in order to follow the tracked subject not only into the virtual space but also into the real environment.

2.2 The Problem

The thesis project is based on an internship with Dolomiti Robotics[1], a company working on self-driving robots.
These vehicles are ideated to work in an industrial environment. In this scenario does not work only robots but also people, making the driving task even more complex to accomplish.

2.2.1 Robot only environment

A looking similar context is a completely automated environment, where humans cannot access. Instead, it is completely different because each vehicle has is its own logic that can be designed to fit the requirement of all the other robots working in that area.
The typical solutions to drive a vehicle in this scenario are two:

- Based on a centralized decision unit that moves all the robots simultaneously around. This unit is responsible to avoid collisions by knowing the exact position of each single moving robot
- Based on fixed rules of movement that each robot has to respect. The rules do not allow collision and these are not violated from the vehicles

Both these methods work because an automated vehicle uses a completed deterministic decision process and do not take arbitrary choices.

2.2.2 Environment shared between robots and humans

Instead, in a shared environment, there are a lot of elements that are not controlled by a deterministic rule. The changes in the scenario are random and no prediction can be done. There are both fixed

object that may have changed position due to external interaction, and also human that walk around with no defined rules.
In this scenario, in order to create an autonomous moving vehicle, it is fundamental to choose an input method that can measure all the area around. The LIDAR has been chosen. LIDAR is a technology that measures distances all around in a horizontal plane. The effect is that the robot knows in each direction which is the distance from the surrounding objects. This key idea has been used from Dolomiti Robotics, to design software able to drive robots around avoiding hit of fixed obstacles or people walking.
While a robot moves around it can construct a map of the fixed object in the environment, measured with LIDAR. Instead, the moving object, such as other robots or people, that are recognised as not fixed elements are not stored in the map as obstacles. This reconstruction allows the vehicle to move autonomously from one position to another knowing exactly which path to follow to reach the destination.

2.2.3 Purpose of the internship

The shared environment does not offer any real human-robot exchange. The two parts only share the same spaces. The goal of this thesis project is to create a real interaction between the two.
Create a new functionality *"that allows a robot to follow a person into the real environment"*.
How does it work:

- Track/follow is the interaction of a robot and a single person (called from now on **Leader**)
- The leader starts the *"follow"* functionality standing in front of the webcam of the robot
- The robot has few seconds to recognise the person inside the camera **FOV (Field Of View)** as the leader
- Then the leader can freely move around in the space.
- In the meanwhile the algorithm is processing the webcam stream of images recognising the position of the leader and start to track it in the virtual space, while following it in the real one.
- The tracking continues for a long period, up to minutes. Until this functionality is stopped.

2.2.4 Technical problems

This "follow" functionality may be easily solvable under certain conditions. But if the general scenario should be solved it is a much harder task.
Below are listed a small collection of the principal problems that make this functionality an extremely general one, and so hard to solve.

- The tracking should be done in real-time. It is impossible to follow a person if the processing speed is too slow.
  A high **FPS (Frames Per Second)** rate should be respected.
- The robot needs to physically follow the person meaning that the webcam is not fixed.
  Due to this, also the background is not fixed and the entire captured image, subject included, might be blurred.
- The person can move freely around walking fast, slow, or stay still
- The leader is a random person, it is not known while the algorithm was designed (no parameters can be fixed in advance)
- While the leader is walking around there might be also other people that create interference to the algorithm
- The leader can be hidden from the webcam due to moving or static elements placed between the leader and the webcam itself

- The leader can exit the field of view of the webcam disappearing until the robot rotates to view it back again
- The tracking should be performed for a long period

2.3 The Solution

The problem is complex due to its generality and the necessity to cover a lot of complementary conditions. For this reason more than one solution exists. In this thesis is presented a solution based on the combination of other three. Each one is designed for solving sub problem respect to this one, and none of them alone can overcome the challenge of the general task.

2.3.1 Existing technologies

The three technologies are:

- **Object Detection (or Localization):** given an image the object detection task aim to process the image and recognise which objects exist there. The detection not only need to produce a list of all the classes[1] of objects visible in the image, but also recognise in which section of the frame every single element is.
  The output of detection is a list of: class to which the element belongs, the probability associated and the **BB (Bounding Box)** defined as the smaller rectangle that contains the entire element.
- **Object Tracking:** in this case the input is not a single image but a video stream and an initial section[2] of it. The goal is to remember this portion of the image and recognise it in all the frames after the first one. It is important to note that the tracking procedure it is not designed to follow a person, a car or other it is designed to follow a rectangle of coloured pixels, no matter what these pixels represent.
- **Object Recognition:** this is a comparison between several pictures. These often represent a bounding box of the object that needs to be recognised. The procedure has a database of images each one with a specific class, and the input value is another picture, called **query**, that do not exist in the database but it represents a subject known. The goal is to extract from the database all the images that have a subject that looks similar to the one represented in the query.
  This application is mainly used to recognise humans, often in the video surveillance context. The database is composed of the bounding box of all the person seen, i.e. in a supermarket during the last week, and when a thief is captured and it is used as a query. So, the system should return all the images containing the thief itself.

2.3.2 Limitation of known technologies

The challenges presented previously in Section 2.3.1, can solve a small part of the general problem but each one has a technical problemSection 2.2.4 that cannot solve:

- Object detection is a computationally expensive task, on a powerful GPU can run in real-time but that's not the case of the robots we are working with.
  In addition, the detection works frame by frame and each one is independent of the previous one. So, if a person is recognised in a frame, and in the next one, there are also some people the algorithm knows nothing about the relation between the person in the first frame and the people in the next one. Meaning that a person cannot be tracked from one frame to the next one.
- Object tracking, according to the name, seems the task that better match the requirement of the general problem.
  Despite that, the tracking does not consider that the tracked subject, the leader, cannot be hidden from the webcam. The leader should always be visible into the recorded video, and

[1] There are a set of types to which each element can be associated i.e. person, dog, car, bicycle, bottle and so on.
[2] A portion of the image: a rectangle.

that's not the case. In addition, the leader can also exit the field of view of the robot while walking around.
Lastly, all the tracker are designed to follow the subject for small periods[3], after a while the tracked rectangle of coloured pixel change and the precision of the output is not guaranteed any more. This phenomenon is known as the **drift effect**, after a while the drift is so wide that the tracked cannot be trusted any more.

- Object recognition due to its requirement was not designed at all to run in real-time. In fact, it is enough to run this procedure only when a query occurs, and that does not happen more than ones every second.
  Except that, there is a more intrinsic problem with the recognition to approach the general problem. The procedure requires a query that can be the subject at the actual frame, but then it should work on a dataset composed of old frames and these are useless to solve the actual frame.
  In addition, this algorithm cannot be independent in fact, the input values are bounding box of the person, but these bounding box can be computed only with an object detection algorithm. So, this approach cannot solve the problem independently.

This explanation shows that none of the existing proposed technologies can solve the general problem in all its parts.

2.3.3 Combine known methods to solve the general task

To solve the problem and manage all the requirements it is necessary to create a combination of known methods.
An example of integration of methods to solve a complex task was done Jiang et al, in their paper[2] that present a fusion of **YOLO9000**[3] (the second version of **YOLO**[4]) used as object detector and **SURF**[5] used as short-term object tracker.
The paper propose an innovative approach based on two thresholds that are used to understand when the drift of the tracker is too large and it is necessary to reinitialize it. So YOLO is executed to find the tracked subject back again and after the initialization the loop can start again.

The method presented in that paper is an integration of two class of methods. Instead in this thesis is presented an integration of three. The third method is necessary because an additional technical problem(Section 2.2.4) exist. Jiang et al. work sport video clips where athletes are always followed by the camera and never disappear out of the field of view. In addition, occlusion can exist but are very short and the tracker is often able to overcome them.
Instead, in our scenario we need to manage the disappear of the leader behind a corner for a relative long time. So, the object recognition method was introduce to solve this condition.
This are the main steps of the entire algorithm:

1. The detection is executed and the leader bounding box is found. For the moment assume, only one person detected in the frame, and it is the leader.

2. The tracker is initialized with the bounding box found

3. The tracker run for the next F frames

4. A new detection is executed and D people are found

5. The person recognition is used to choose if the leader is contained in the list of people found

   - If **yes**: the procedure start again from point 2 (tracking)
   - If **not**: the procedure loop again from point 4 (detection again)

[3]Each tracker works on a video of few seconds.

This flow shows how detection, tracking and recognition are combined together to build a complete algorithm, that can run in real-time due to the alternation of slow and fast methods and to manage all the problem scenarios.
The details will follow.

2.4 Structure of the thesis

The next chapters are organized as follow. This section conclude the introduction (Chapter 2).
Then follow three chapters one for each main method: object detection in Chapter 3, object tracking in Chapter 4 and object recognition in Chapter 5.
An overview of the entire algorithm and how it works together follow in Chapter 6.
In the end, the conclusion are presented in Chapter 7.

3 Object Detection

This chapter explains into the details what is the object detection task and the methods that can solve it efficiently. An overview of other methods is given, both for the not efficient one but also for problems that may look similar but are not the same.

3.1 Task definition

Object detection, also know as **object localization**, is an evolution of the **image classification** (Figure 3.2). In classification, an algorithm should produce a list of all the classes of objects inside the image. Instead, the detection not only calculates which object class exist but also how many occurrences are present for each class. Then the complex part, and the most interesting one for this thesis application, is localizing where those elements are placed inside the image. The position is not considered as a point but as a bounding box defined as the smaller rectangle that contains the entire element. An example of object detection is shown in Figure 3.1.

Figure 3.1: Object detection applied on a sample image.

3.1.1 Similar tasks

Object detection can be additionally improved to extract even more information from an image.
The main evolutions, shown in Figure 3.2 are:

- **Semantic segmentation:** took all the bounding boxes produced by an object detector, and for each one, it calculates the pixels that belong to the object itself and the one that not. Doing this each class has its own colour associated. As result, the algorithm knows for each pixel if it belongs to one label (semantic division) associated with the image or to the background (yellow in the image).

- **Instance segmentation:** is similar to semantic segmentation, but in this case, each instance of an object is considered as a new element. In fact, the three cubes in the figure have associated different colours.
  This task is solved by the **Mask-R-CNN algorithm** (??)

- **(Human) pose estimation:** is the more complex task between the five. Mainly applied to people, this challenge consists in the estimation of the 3D position of the body. The idea is to build up a skeleton of the person in the image and understand how its body limbs are positioned. This functionality is important to understand what a person is doing in the image.
  This task is solved by the **OpenPose estimation algorithm** (??)

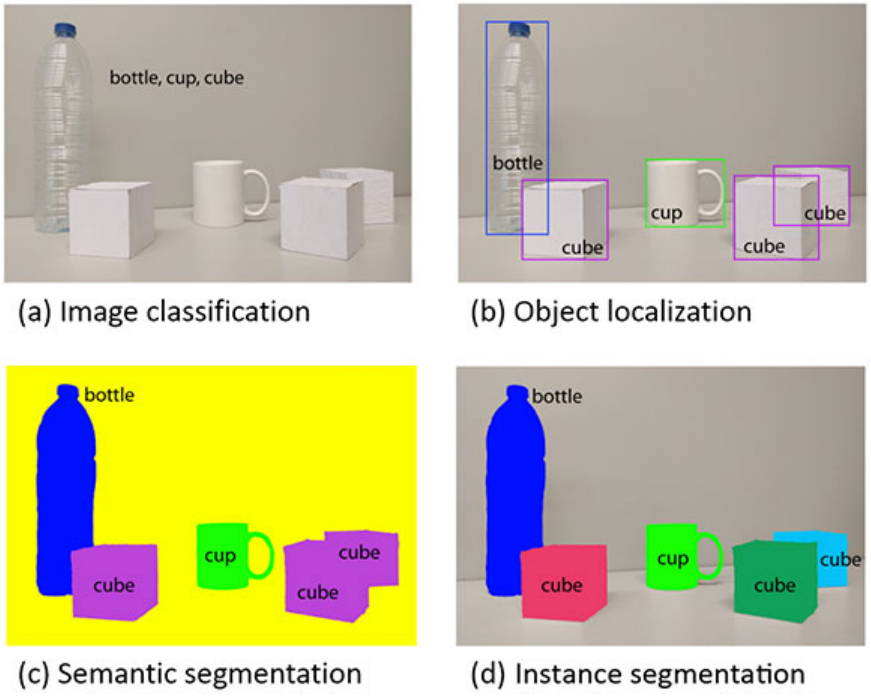


Figure 3.2: Similar problems respect to object localization/detection.

3.2 State of the art algorithms

Object detection has a lot of application both in real-time, such this one, but also into safety-critical scenarios like cars with autonomous driving. This division brings out two different metrics: precision and speed. The ideal detector is both fast and precise, but this algorithm does not exist yet. The methods can be divided into two.
The solution mainly focused on speed: YOLO (Section 3.2.1) and SSD (Section 3.2.2).
Instead, the one mainly focused on precision is R-CNN (Section 3.2.3).

3.2.1 YOLO (You Only Look Once)

YOLO[4] was initially designed in 2016. At that time was the first object detector approach to use a single **CNN (Convolutional Neural Network)**. Redmon et al. goals were to create an extremely fast detector. An overview of the overall procedure is to show in Figure 3.3, and the architecture in Figure 3.6.
The image shows a two steps procedure, but these steps are solved in parallel. This is the core idea of the paper. A single CNN can be highly optimized.
The YOLO procedure works as follow[1]:

- Preprocess: The image is resized to fit the standard input dimension of the CNN

[1]The original presentation of YOLO by Redmon at the CVPR 2016 can be found here.

- Left image: Then it is divided into a grid of CxC cells
- Top image: Each cell proposes some bounding box centred on it that can match elements in the background. To each box is associated value describing the probability that it contains one of the elements of the image.
  At most one detection per box can be selected as correct. This rely on the assumption that two correct bounding boxes cannot share the centre. This is both an efficient idea but also a big limitation. Too small elements, close to each other, cannot be both detected.
- Bottom image: To each cell is associated with a probability regarding a class that represents the class that can be found in that cell if an element exists in it.
  I.e. the cyan cells means: "if there is something here, it will belong to class 'DOG' ".
- Right image: the two partial elaborations are merged. The most likely bounding boxes are chosen and classes are associated to them according to the probability for each cell in the probability map.

That was the first YOLO version, in this thesis is used the third[6]. Mainly the changes were about recognition of a wider set of classes and small implementing details to improve the overall precision of the algorithm.

The output of the CNN is generated extremely fast and it is accurate but has a big problem. Often if two classes have similar probabilities or the shape of the element is not perfect YOLO might propose more than one bounding box for each element. That's the case of Figure 3.4c where the truck is classified both as "truck" but also as "car". The same happens to the person that has been seen twice.
To solve this, it is necessary to apply a new technique: Non-Maximum Suppression

NMS (Non-Maximum Suppression)

This technique[7] is post-processing that works on the bounding boxes proposed, as output, from YOLO or other detectors. It does not consider the source image. The goal of this procedure is to refine the bounding boxes proposed and choose which subset of them is better to fit the final image prediction. Two examples of applications are shown in Figure 3.4
The main flow of the algorithm is as follow:

- The input is a list of all the boxes generated for a single image. Associated to each one there is its probability.
- The boxes are sorted in decreasing order according to the probability associated.
- Then, in order, each box is accepted or rejected according to the **IoU (Intersection Over Union)**. That is the percentage of overlapping area with an already accepted box.
  - If the IoU is above a certain threshold, meaning that the two boxes overlap too much the one with the lower probability is discarded.
  - If that's not the case, the box is accepted as a new prediction.

The input in Figure 3.4a, is processed and only one box is accepted (Figure 3.4b) because the IoU is very high. Instead, in Figure 3.4c two boxes are removed respectively from two other separated boxes (Figure 3.4d) because two different subjects are involved.

3.2.2 SSD (Single Shot Multibox detector)

The principal competitor of YOLO is SSD[8]. Both are based on the same principle: use a single Convolutional Neural Network to propose bounding boxes and associate them to classes. Then optimize this CNN, as much as possible, to improve the speed performances and eventually even the accuracy. The difference relies on how the two algorithms deal with the bounding boxes proposal. YOLO for

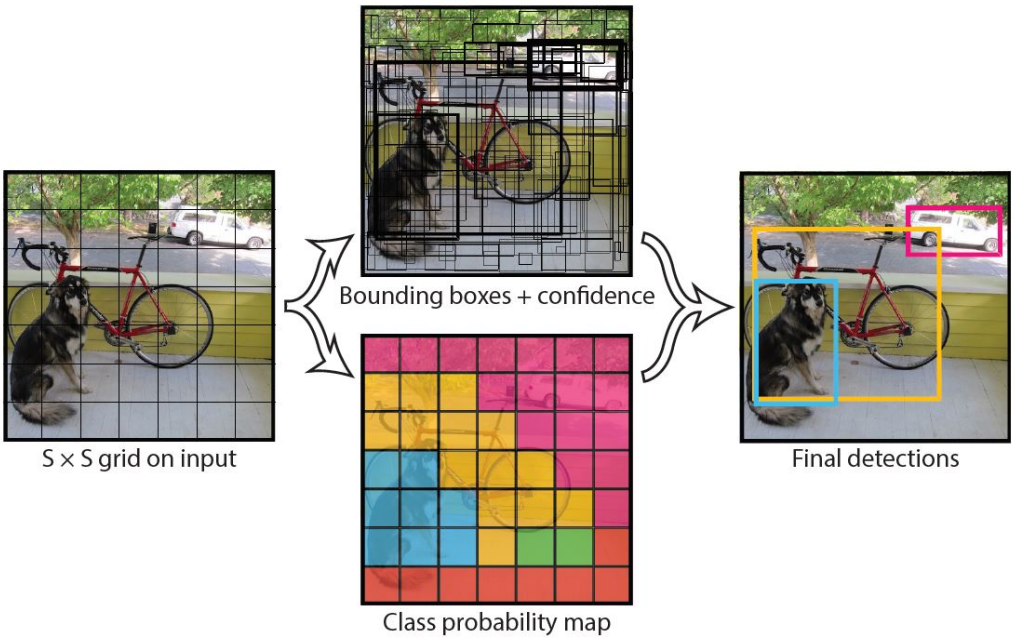
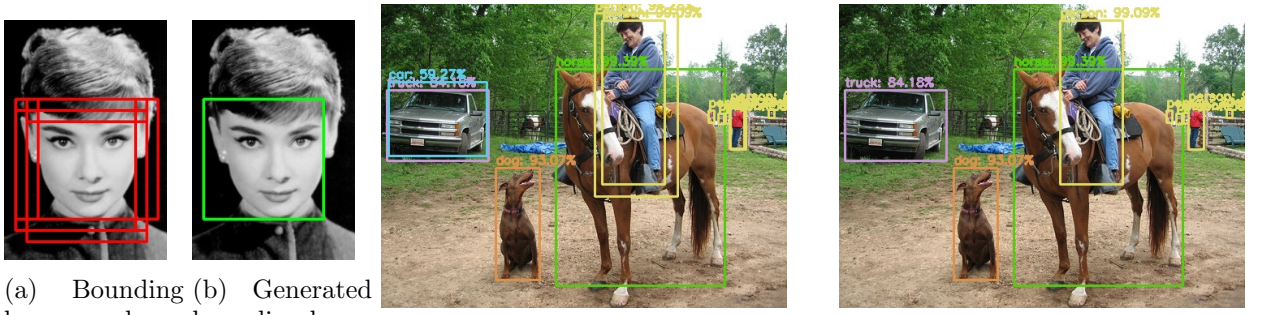


Figure 3.3: The YOLO image elaboration based on bounding box proposal and class probability map.

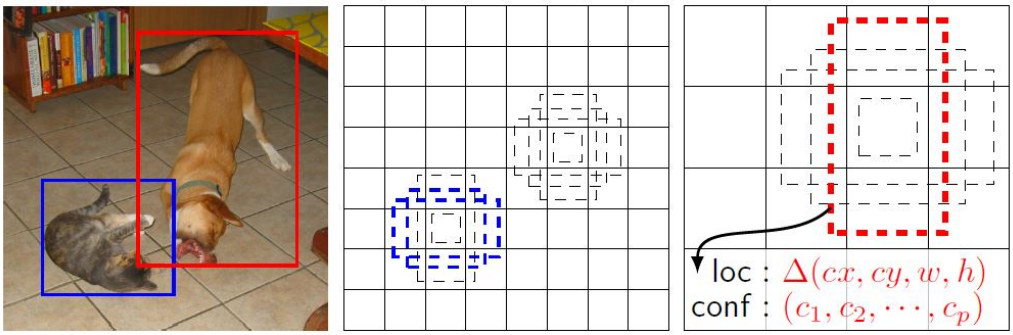


(a) Bounding boxes overlap (b) Generated bounding box (c) A YOLO generated bounding boxes (d) Apply NMS to get correct output

Figure 3.4: Two scenarios of application of Non-maximum suppression algorithm. First: choose which of the 6 manual generated bounding boxes, on Audrey Hepburn face, should be considered the correct one. Second: refinement of the YOLO prediction output, by removing the "car" and "person" prediction.

each cell of the grid choose a couple of options and at most one can be chosen.
On the other hand, SSD works as follow (Figure 3.5):

- The image is divided into a grid of CxC cells, called **feature map**.
- Each cell can propose a set of default boxes that has a size measured in cells (i.e. 3 cells high and 2 wide).
  Each of these proposes comes with a confidence probability and an offset respect to the ground truth elements in the image.
- The process is repeated many times varying the value of C: the **granularity of the grid**.
  This guarantee that the algorithm is scale-independent matching both big and small subjects.
  In Figure 3.6 is shown how the convolution layers block are matched together only at the end.
- All the proposes are merged together to produce the final proposals.
- SSD internally performs NMS, to remove unnecessary detections.



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

Figure 3.5: The SSD image processing and how the bounding box proposal is elaborated.

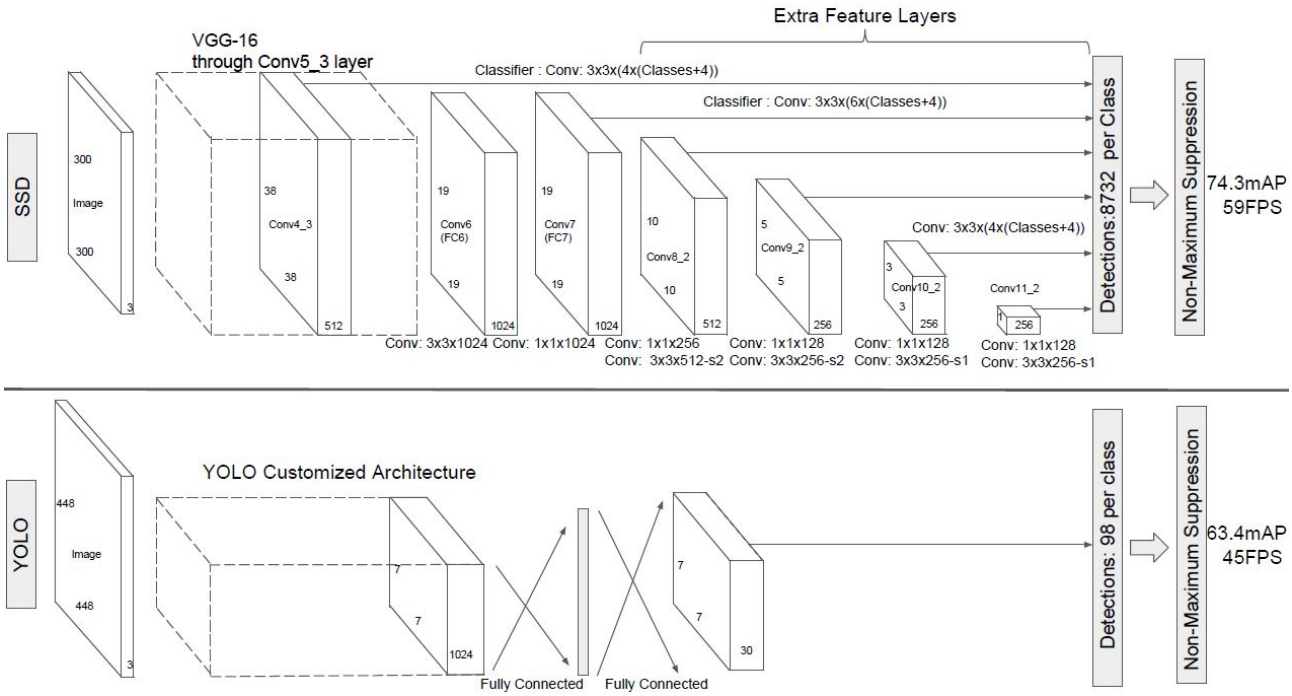


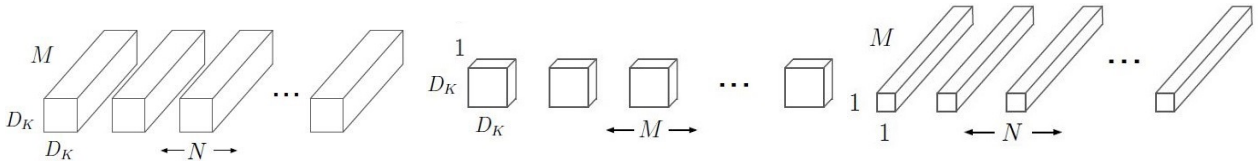
Figure 3.6: A comparison of architectures between SSD and YOLO that is designed as a compact block. Instead, SSD is modular: divided into convolution layers of different scales, combined at the end, to make the algorithm scale-independent.

MobileNet

The implementation of the project does not use a traditional version of SSD, but a lighter one. This model is a combination of SSD and mobileNet[9] and was first proposed in that paper.
MobileNet is a methodology that approaches Convolutional Neural Networks to transform the architecture structure to build a much lither version of the model. The concept was first ideated to allow low power devices, such as smartphones to run computational expensive algorithms based on CNN.
The principle is to replace each standard convolution (Figure 3.7a) with a **Depthwise separable filter**. A standard convolution works on a grid of DxD pixels and for each one produce output features of depth M. This operation can be repeated N times for each source feature.
The operation is broken into two other simpler convolutions:

- **Depthwise convolutional filters** (Figure 3.7b): produce only one feature output at a time, repeated M times for each DxD grid.
- **Pointwise convolution filters** (Figure 3.7c): extend the output feature of the depthwise filter to N output features.

The original paper demonstrate how these two operations stacked in a row can produce results close to the correct ones.
The computational cost determined by the number of parameters, used by depthwise and pointwise filters, can be further reduced by randomly remove a percentage of these parameters. According to the portion of parameters removed (25%, 50%, 75%), the algorithm precision is affected.



(a) Standard convolution filters (b) Depthwise convolutional filters (c) 1x1 convolutional filters called Pointwise convolution filters

Figure 3.7: The novelty of mobileNet is that it converts a traditional convolution (A), into a combination of two lighter convolutions(B-C), that produce almost the same output.

3.2.3 R-CNN (Region-based Convolutional Neural Networks)

[10][11][12][13]

3.3 Other famous algorithms

3.3.1 Mask R-CNN

[13]

3.3.2 Open Pose

[14][15]

3.4 Which algorithm for our scenario

better title: Algorithms performances

4 Tracking

5 Recognition

6 Solution

7 Conclusion

Bibliography

[1] Dolomiti robotics. `https://dolomitirobotics.it/`.

[2] Sicong Jiang, Jianing Zhang, Yunzhou Zhang, Feng Qiu, Dongdong Wang, and Xiaobo Liu. Long-term tracking algorithm with the combination of multi-feature fusion and yolo. In *Chinese Conference on Image and Graphics Technologies*, pages 390–402. Springer, 2018.

[3] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[4] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[6] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[7] A. Neubeck and L. Van Gool. Efficient non-maximum suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855, 2006.

[8] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[9] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[11] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[13] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[14] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *arXiv preprint arXiv:1812.08008*, 2018.

[15] Yaadhav Raaj, Haroon Idrees, Gines Hidalgo, and Yaser Sheikh. Efficient online multi-person 2d pose tracking with recurrent spatio-temporal affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4620–4628, 2019.