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 tracking of a person in complex scenarios

Professor
Luigi Palopoli

Student
Stefano Leonardi

Co-Supervisor
Stefano Divan

Co-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.

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

- **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 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.

[2] A portion of the image: a rectangle.
[3] Each tracker works on a video of few seconds.

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)

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** (Section 3.3.1)

- **(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** (Section 3.3.2)

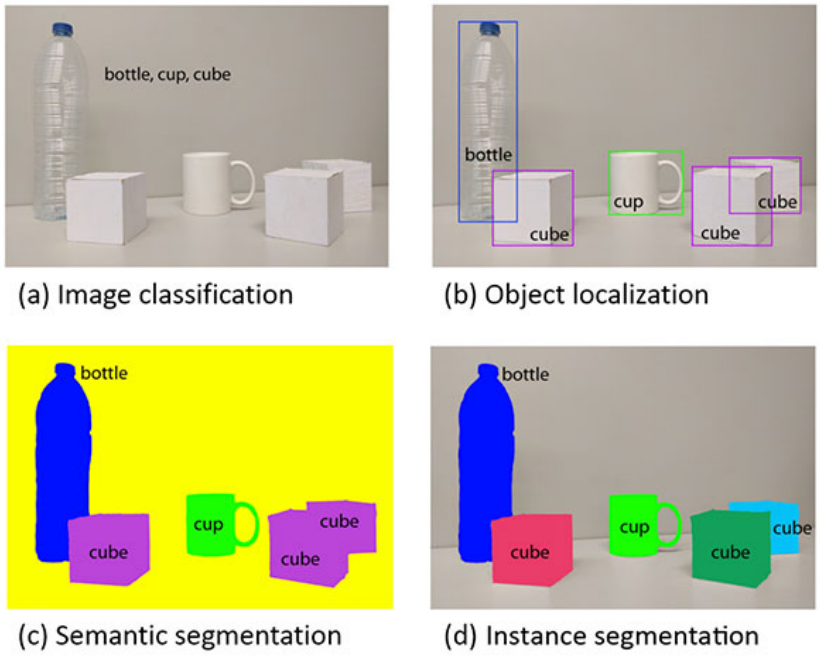


Figure 3.2: Similar problems respect to object localization/detection.

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 shown 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
- 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.

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

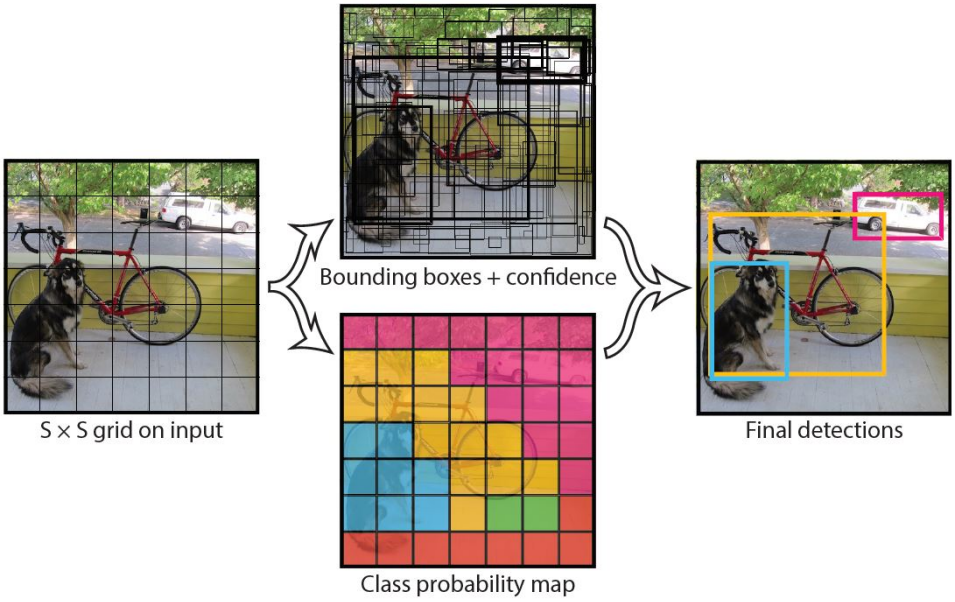
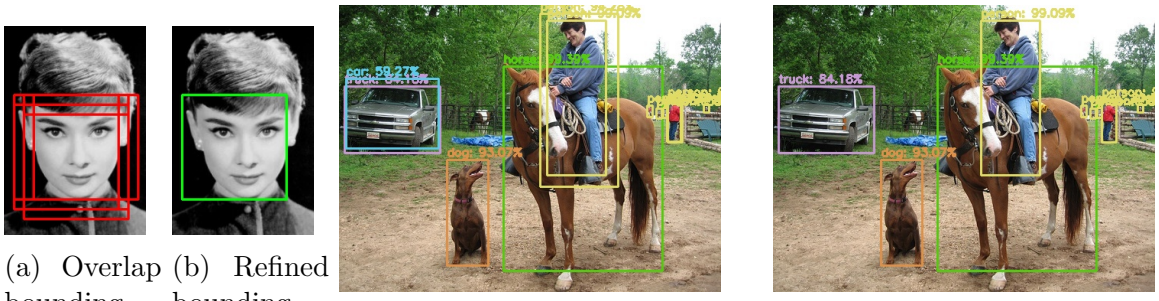


Figure 3.3: The YOLO image elaboration based on bounding box proposal and class probability map.

- 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.



(a) Overlap bounding boxes

(b) Refined bounding box

(c) YOLO generated bounding boxes

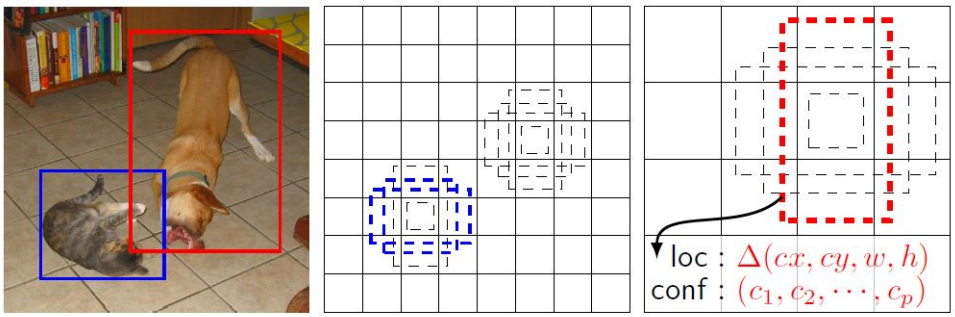
(d) Apply NMS to refine the YOLO's output

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.

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 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.

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**

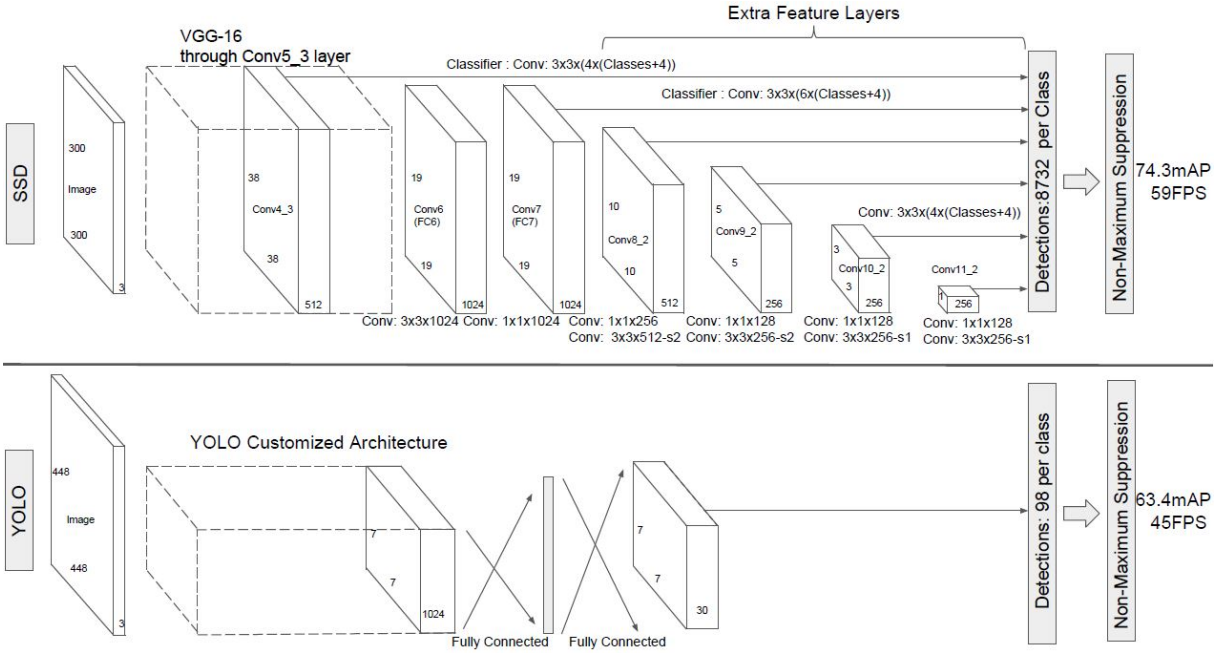


Figure 3.6: A comparison of architectures between SSD and YOLO that is designed as a compact block. Instead, SSD is modular: divided into convolution layers of different scales, combined at the end, to make the algorithm scale-independent.

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:

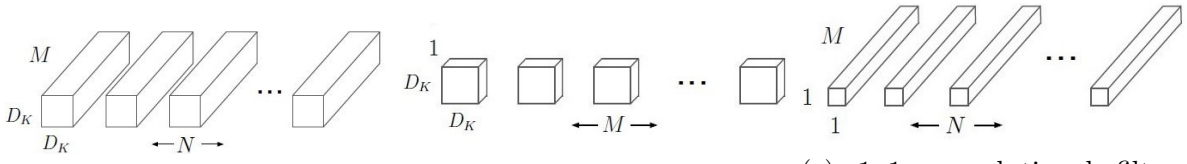
- **Depthwise convolutional filters** (Figure 3.7b): produce only one feature output at a time, repeated M times for each DxD grid.
- **Pointwise convolution filters** (Figure 3.7c): extend the output feature of the depthwise filter to N output features.

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.

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

R-CNN[10] was the first invented method of the three in this section. Differently from YOLO and SSD, it is mainly focused on performing detections with high precision despite the processing time.
This algorithm is a two steps object detector.
The workflow, shown in Figure 3.8, works as follow:

1. A region proposal algorithm is executed on the input image and it produces 2000 bounding boxes proposal.



(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.

2. Each of these proposals is elaborated independently.
   For each box, an image classifier, based on CNN, produce features from the image and then predict which classes they might contain.
   Any kind of image classifier can be used for this task resulting in an algorithm that can be easily adapted with new networks.

The main problem is that overlapping proposals are elaborated independently. The result is that feature extraction is performed on the same area of the image multiple times. These have been solved with a second version of the algorithm, called **Fast-R-CNN**[11]. The feature extraction for the full image is performed before the image classification that now works on an already generated image of features.
An incremental improvement, comes from the third version: **Faster-R-CNN**[12]. That performs the feature extraction as the first step and based on it the bounding boxes are proposed with **RPN (Region Proposal Network)**. The modified structure allows ad hoc optimizations to improve the low processing time.

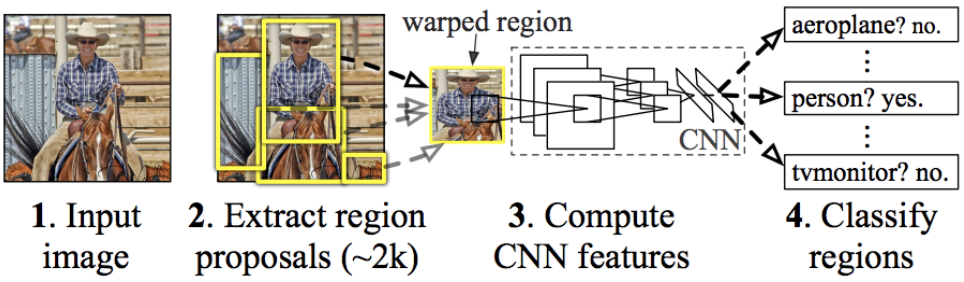


Figure 3.8: The 2-step elaboration of R-CNN model on a sample image.

3.3 Other famous algorithms

3.3.1 Mask R-CNN

A variation of R-CNN that aims to solve the instance segmentation task (Figure 3.2) is Mask R-CNN[13].
Mask R-CNN is built on top of two technologies:

- Faster R-CNN used as an object detector.
- **FCN (Fully Convolutional Network)**[14] that perform semantic segmentation.

For each bounding box, it is known the class, then FCN compute the segmentation of that class. All the shapes of elements in the image are then merged together to build the instance segmentation result. An application of this algorithm is shown in Figure 3.9.

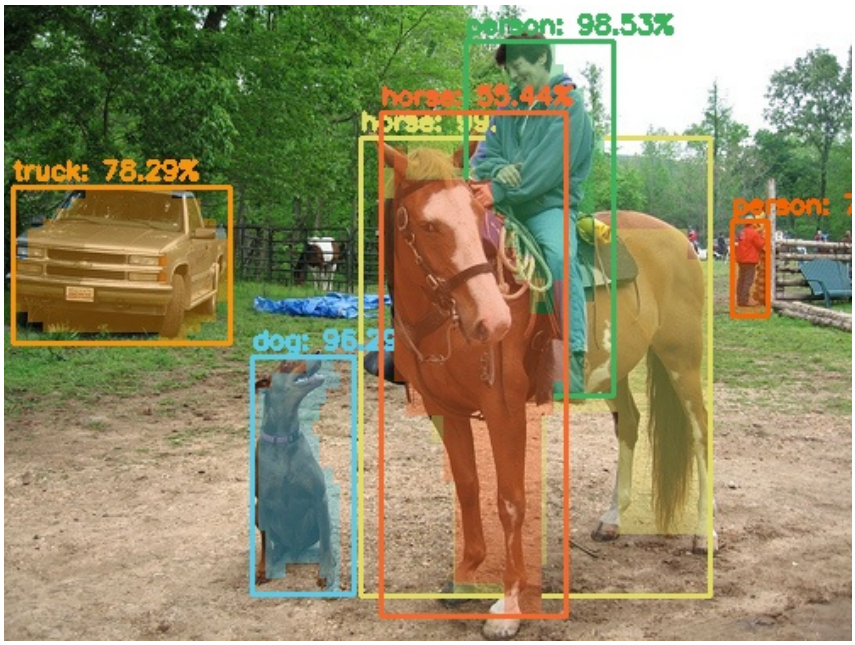


Figure 3.9: Instance segmentation of an image using mask R-CNN.

3.3.2 Open Pose

Firstly designed in 2017 OpenPose[15] aims to process an image and recognise the position of the people in it. The position is the skeleton of a person, it is the interconnection of limbs that link 15 points on the human body.
This algorithm open al lot of possibilities because until then estimating the body position was achieved with 3D cameras extremely expensive hardware that now can be easily substituted. Recently, OpenPose has been improved both in term of speed, to process frames in a video with **STAF (Spatio-Temporal Affinity Field)**[16], but it is also been extended to understand the 3D human position.
An overview of the overall procedure of the algorithm is shown in Figure 3.10, instead some output examples are shown in Figure 3.11.
The OpenPose procedure works as follow[2]:

a) Preprocessing: the input image is reshaped to match the requirements of the two-branch CNN.

b) Part Confidence Maps: the first branch of the CNN process the image to extract the location of the 15 body parts.
Each body part (i.e. left shoulder, left knee, right wrist...) is detected by an ad hoc filter. These filters do not process one person at a time but the entire image simultaneously. The result is that a filter recognises all the visible right elbow in the image. This is extremely important because by doing this the algorithm process speed is independent respect to the number of people in the image.

c) Part affinity fields: the second branch of the CNN process the image to recognise the limbs that can connect all the body points find so far.

d) Bipartite matching: has the goal to match all the elements found to reconstruct the skeleton of the entire person.

[2]The original demo of OpenPose by Zhe Cao at CVPR 2017 conference can be found here.

This reconstruction is a greedy approach mainly based on geometry that wants to minimize the distance of two parts that must be connected.

e) Parsing results: the output image is assembled with the full-body poses for all people in the image.

It is important to note that OpenPose is a bottom-up approach. The algorithm has no knowledge about the positioning of people in the image, it only tries to reconstruct the skeleton from small sections of the body.

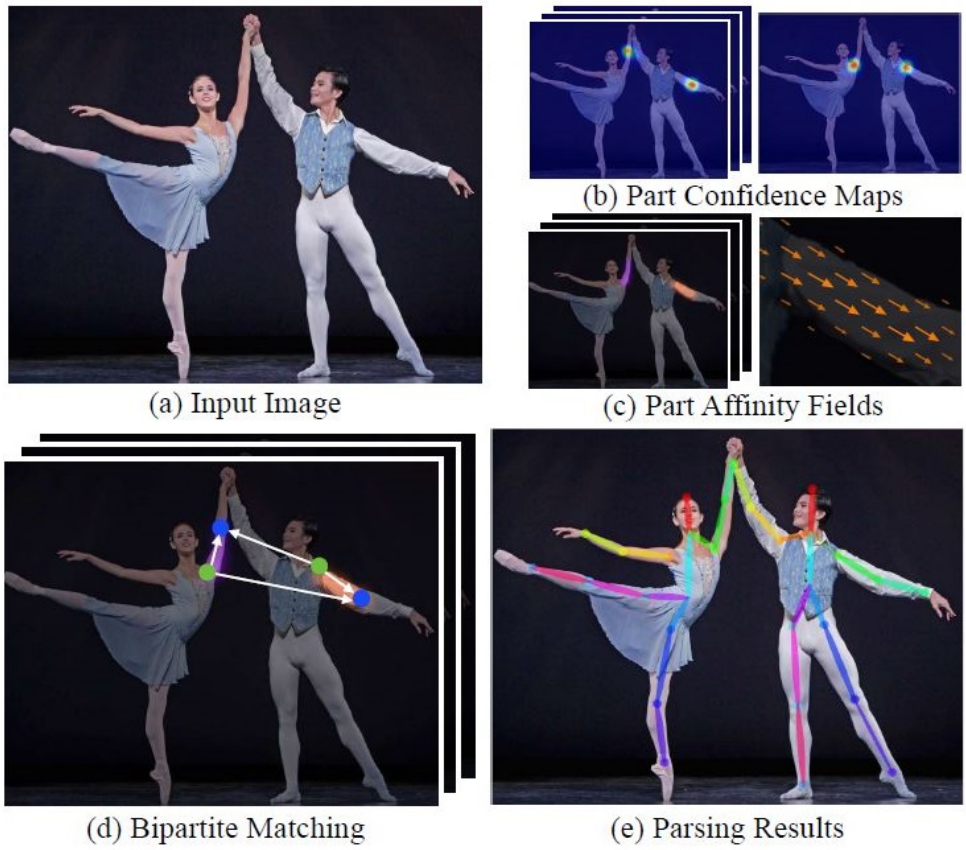


Figure 3.10: The elaboration of the OpenPose algorithm to recognise the skeleton of the two dancers in the image.

3.4 Overview of the algorithms

The algorithms presented in this chapter represent the state of the art methods for their field of application.
All those methods can be used to achieve the long-term tracking that is the goal of the thesis, but the different information generated should be used to solve the problem with different approaches. We have chosen to use the methods that perform only the detection task. This result on one hand into general information as output but on the other hand a very high processing speed that is fundamental for real-time application.

Performances comparison

A comparison of the speed measured as **Frame Per Seconds (FPS)** and of the precision measured with **mean Average Precision (mAP)** is shown in Table 3.1. The data are

Figure 3.11: Some examples, taken from the original paper, on how OpenPose works.

based on the Pascal VOC 2007 dataset[17] and comes from multiple papers[4][3].
The measures do not show YOLOv3 because respect to the other methods is more recent and a fair comparison do not exist. Instead, for Mask R-CNN and OpenPose, only the frame rate is shown because the mAP can be computed but it is completely irrelevant respect to the other presented there. These two algorithms perform different tasks and so the precision of the result cannot be compared.
Looking at the data appears evident that the single-stage algorithms (SSD and YOLO) are much faster respect to the two-stage methods (R-CNN), in fact, run around 5-10 times faster. Instead, the precision of the three detectors is almost the same. For these reasons, we have chosen to use in this project the last version of YOLO and a lighter version of SSD: mobileNet-SSD. This idea pays a few percentage points in term of mAP but implements the CNN with mush fewer parameters, resulting in a low power consumption method. This aspect is important because the robots mount not top quality hardware then the light version of SSD can be executed easier.

Output visualization

To visually shows the potentialities of these algorithms we applied all of them on the same picture. This elaboration is presented in Figure 3.12. Independently from the task that they solve appears evident that all of them recognise the 5 people in the foreground, but there are some differences:

- SSD has trouble with the player on the left. In fact, the percentage associated with them is only a 32%
- YOLO is able to detect event a sixth person in the background that none of the others has seen
- YOLO is trained to recognise a wide variety of objects respect to the other detectors, in fact, it is able to recognise even the "sports ball".
  Even SSD is adaptable and can be trained to recognise the ball. But the big advantage of the second version of YOLO, also called **YOLO9000** is that it was integrated with the **Wordnet graph**[18] to be scalable in term of the number of

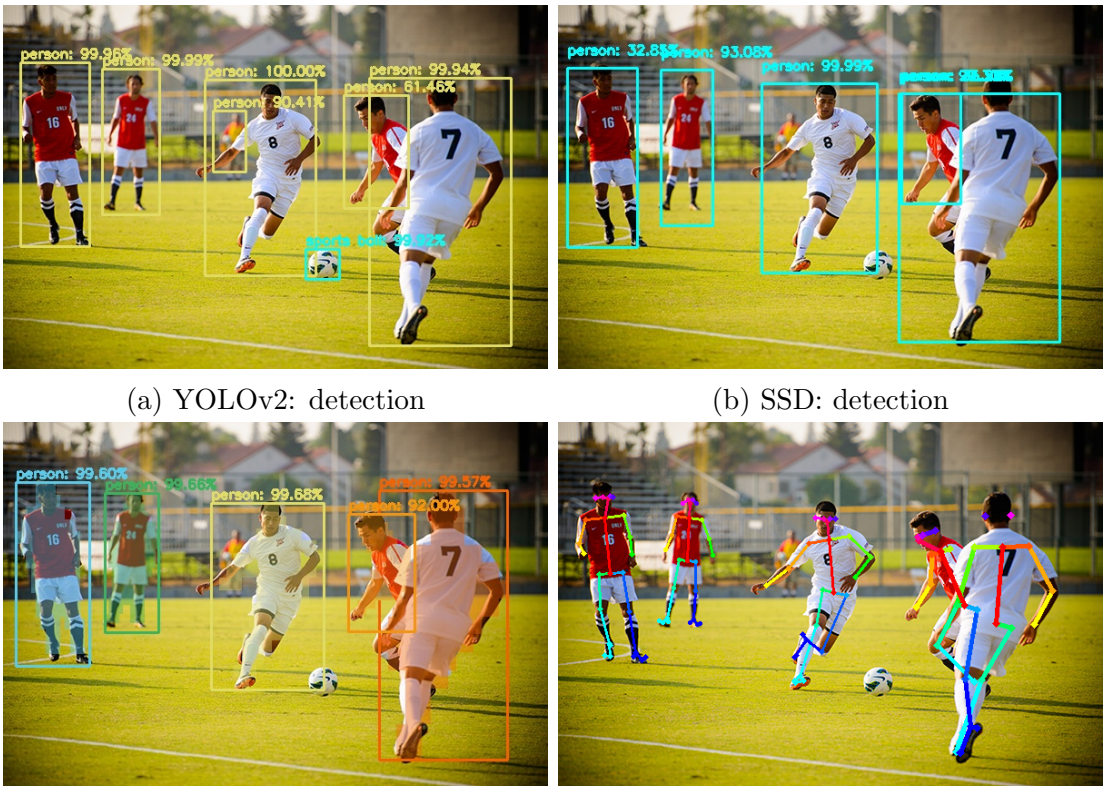
classes recognised. The result is a detector that is able to recognise up to 9000 different classes.

- Mask R-CNN is, in this example, the most accurate algorithms. It recognises four people with a precision of 99% and the last one with 92%.
- OpenPose by estimating the body parts can even understand the orientation of the people in the soccer field. This big advantage can be used to understand where these people will move in the frame after this one.

Considered these reasons seems evident that discard OpenPose and Mask R-CNN, in favour of SSD and YOLO, is only a choice of this project. All these algorithms have potentiality that can be used to create a good object tracker.

Pascal VOC 2007	YOLO			R-CNN				
Algorithm	v1	v2	SSD	R-CNN	Fast	Faster	Mask	OpenPose
FPS	45	**67**	46	0.05	0.5	7	7	10
mAP	66	**<u>76</u>**	<u>74</u>	53	70	<u>73</u>	X	X

Table 3.1: Overview of the performances of the algorithms presented. The evaluation is based on the mAP and the processing speed. In bold the best scores and underlined the mAP of the three states of the art detectors.



(a) YOLOv2: detection (b) SSD: detection

(c) Mask R-CNN: instance segmentation (d) OpenPose: human pose estimation

Figure 3.12: An example of application based on the same image, of the four main algorithms.

4 Object Tracking

The chapter is focused on a few methods that solve the traditional task of object tracking. For each algorithm, the potentiality is shown and the advantages compared to other methods, will be discussed. In addition, an overview of similar problems is givenSection 4.1.3.

4.1 Task definition

Object tracking is a challenge that, differently from the detection, does not work with images but deal with videos. A video is technically defined as a sequence of images, called **frames**, combined together at a certain frame rate. Typically this rate is 15, 30 or 60 **Frames Per Second (FPS)**.
Despite the existing of a video, each algorithm process one frame at a time. The interconnection of subsequent images is given by the knowledge, that the method has about the previously analysed frames and uses the extracted information in order to better understand the new incoming images. Differently, an object detection algorithm does not store information about precedent elaboration because figures coming from a database are completely independent of each other. For frames of videos, this is not true.
The task baseline is defined as:

> ***The traditional tracking challenge*** *consists of tracking a single well-defined subject over a short video clip, no matter if it is a person, an animal or an inanimate object. The target of the tracking clearly fully appears into all the frames of the sequence.*

4.1.1 Interface

Structure of the tracking algorithms interface:

- One input value is composed of an image, typically the first frame of a video. In the case of real-time elaboration, the scenario in which we are interested, the image correspond to the actual view of a webcam connected to the device.
- The other input corresponds to the bounding box coordinates of the subject that need to be tracked. The bounding box can be manually generated or can be automatically identified with an object detector.
- All future steps get as input a new frame. The task is to understand, in this new image, where the subject defined by the initial bounding box is located.
  Then, if all these elaborated images, and the generated bounding boxes, are combined together the result is a video. In which the initial subject while moving is always centred into a rectangle that follows it across the entire field of view of the camera.

4.1.2 Subject of the tracking

It is important to note that the task is called "*Object tracking*" and not "*Person tracking*" this because there are no limitations about the subject that need to be tracked, it does

not need to be a human. A better name for the challenge could be "*Area tracking*" because the algorithms should follow the rectangle of coloured pixels that was located in the initial bounding box. No assumption can be done of which type of subject exist inside the bounding box.
Generally, these methods work if the initial position is centred around an object that moves consistently in the space in future frames and does not change aspect completely. Some examples that could break these methods are:

- The initial bounding box contains two or more objects that move independently from each other. The algorithm will recognise one of the two as the main subject and will track it over the video and lose the other.
  This is both a problem if accidentally the bounding box contains something that is "more important" than the "*real subject*". But also an advantage that allows to easily discard the background that is an "*object*" itself, but almost always not an important one.
  In addition, track two elements can cause inconsistent movements, not in one direction but in multiple directions simultaneously and this is not possible for a single object, as it should be.

- The subject change aspect too rapidly due to different luminosity in the environment, or elaborated video sequences that are not "*natural*". This last possibility often breaks methods designed for real situations, so it is not a problem only in this case. Instead, the change of luminosity often occur but a solution can be achieved with frame pre-processing, such as standardize the luminosity of the image to always process a figure with the same luminosity mean.

4.1.3 Deal with special conditions

The tracking task can be seen as a set of problems. This because there are a lot of conditions that can modify the scenario where the algorithms should work. By modifying the type of difficulties in the videos some methods may fail while others do not.
The goal of this thesis is to build a "person tracker", that should work under certain conditions and solve them. The problem is that the combination of a lot of requirement makes the problem always harder.
Despite the time spent for a programmer to ideate, implement and test a solution there is a trade-off to consider. Solving a hard task requires a more computational complex solution with respect to solving a similar easier task. This complexity influence the performances of the proposed algorithm. To conclude, it is fundamental to understand which are the requirements of the problem that we are dealing with in order to choose the algorithm that it is better to use, to solve the task and perform fast.

Below are listed a set of requirements that make the baseline harder (definition is in Section 4.1):

- **Changes of 2D shape**: the target due to movements might change its ratio, aspect, and shape. We are interested in what the camera see of the subject (2D space) and not the effective actual condition of the subject (3D space).

  - Partial occlusion (Figure 4.1f): it may append that during the video part of the subject is occluded, by an object. If this happens the algorithms must be designed to be robust and localize even with a small section of the entire

target.
The bounding box generated may contain only the visible part of the subject but it is also the case that an estimation of the entire subject is done and the bounding box contains both the visible and the estimated missing area of the target.

- Rotations and deformations (Figure 4.1b): the object tracked while moving can rotate and show to the camera a different side. Or if it is deformable change the shape (i.e. a person walking change the shape continuously). This might influence even colours.

- **Total occlusion** (Figure 4.1i): the subject completely disappear behind an object or out of the camera field of view.

  - Short-time occlusion: the subject is hidden for a very small number of frames (i.e. 5 or less). This often happens when the subject or an obstacle is moving and the three elements: subject, object and camera are aligned. These very short occlusions may be solved using a little memory that store the subject information for the last few frames (i.e. 10).

  - Long-time occlusion: the occlusion last for a bigger number of frames even seconds. Often occur when the subject exit the field of view of the camera and do not enter again for a while. It can also happen if the subject is behind a big obstacle such as a vehicle or a wall.
    Harder to solve respect to the previous scenario, the solution requires a long term memory associated with a recognition procedure to understand when the subject is visible back again.

- **Fast-moving object**: the 2D movement of the object respect to the position on previous frames is high. This could happen because the subject is effectively moving fast, or because it is close to the camera and even a small movement looks big.

  - Blurred subject (Figure 4.1d): due to the movement the subject is blurred this heavily change its aspect. The modified elements are the shape and the colours that are somehow faded. In addition, especially for humans, moving fast can make disappear parts of the body such as arms or legs.

  - **Proximity assumption** (Figure 4.1g): a lot of trackers are based on the assumption that the subject moves around only a little bit. If it moves fast this principle is broken.
    After knowing the exact location of the subject on the previous frame starts the estimation on the successive one.
    At this point there are two things to focus on:

    ➤ If the subject does not move is probable that the prediction will place the target in the exact same position as before. Instead, if it moves the probability that the tracked object will be located close to the previously known position is higher than to be located far away. Specifically, the likelihood can be seen as a **Multivariate Normal Distribution** centred at the previous position and stretched towards the direction where the subject is moving (representation at Figure 4.1h).

➤ If multiple similar objects exist in the frame and the tracking is following one of them is probable that with a fast movement that generates an unpredictable big shift, the target moves far away from the previous position and the tracker recognise its subject into a similar object. From now on, the tracking is broken because it follows the wrong physical element.

➤ A big shift can also be wrongly interpreted as total occlusion.

➤ On the other hand, reduce the tolerance to big movement allow algorithms to only search locally around the last known position resulting in a big improvement of the overall performances.

- **Low-resolution images** (Figure 4.1a): such as with many other computer vision challenges the resolution of the input image is fundamental. A low number of pixels per frame, result into a bounding box (often a small portion of the entire image) of very low resolution. Because of this, recognise the key elements that identify the subject from the rest is hard, but also faster and less computationally expensive.

- **Moving background/camera**: work on a fixed camera allow the use of a set of technique based on background subtraction. The key idea is to get prior knowledge of the background and understand which objects are there, by removing the known part pixel by pixel. A high luminosity change can make this harder, but still possible. Note that a camera that rotate always on the same section can be managed as a fixed camera, by combining all the images along with the rotation as a unified one, creating a panorama image.
  Instead, a moving camera implying a moving background makes everything harder:

  – No background subtraction can be done.

  – The subject may change aspect even if it does not move.

  – The occlusion, both partial and total, can occur more easily.

  – A zoom or rotation of the camera causes a similar effect such as the "fast-moving object"

  – To estimate the movement speed and direction of the subject it is necessary to know the movement of the camera because one is relative to the other. But almost always the displacement of the camera in the space is unknown.

- **Real-time**: design a method to run in real-time require to focus on the computational capability and respectively to the processing speed. Real-time might vary from 1 to 60 FPS according to the application. To achieve this speed often is necessary to choose a faster method instead of an accurate one. This reduces the overall precision of the entire designed algorithm.

- **Long-term video sequences**: the input frame sequence is long more than a few seconds, up to minutes. Firstly, a so long video might easily contain some of the problems listed above. In addition, try to locate the same subject over and over again without a reinitialization can fail due to the drift problem.
  The **drift problem** consists of an accumulation of small errors along all the tracking period. A tracking algorithm refers to the bounding box generated for the previous

frame. From that bounding box is extracted the subject that should be located again in the new frame. If the tracking last for a limited small number of frames the subject may look similar to the original one, instead during long processing the bounding box start to derive from the original subject. Due to partial occlusion, the box might be reduced. Due to a strong change of luminosity, the bounding box might be misaligned. Due to a fast movement, the box can be linked to the background. By continuing summing up all these little problems at the end the main subject will be recognised as a superfluous element and the tracker stop to consider it useful and the track fail.
To solve this problem the solution is to re-initialize the tracker often before reaching the limit of the drift. This can be done with different methods such as a Kalman filter or by recognising the main subject with a detection, that is the method presented in this thesis.

- **Multiple subjects** (Figure 4.1e): if there is more than a subject to follow the naive solution is to apply an object tracker to each of them. Each tracker works independent to the other but given X subjects the performances f measured in FPS are reduced to f/X.
  An efficient solution consists of considering all the tracked subject at the same time with a single algorithm instance.

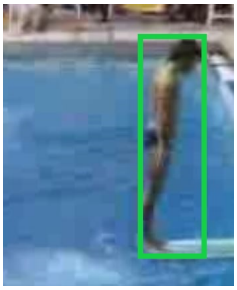
- **Type of the subject** (Figure 4.1c): the baseline problem does not assume any kind prior knowledge about which type of subject should be tracked. But sometimes ad hoc solutions are required, this often simplifies the problem. Frequent choices are people (such in this thesis), animals, vehicles or inanimate objects. For example, if vehicles are chosen the change of shape is not a problem because a car always looks the same.

4.1.4 Traditional tracking problem compared to the thesis project

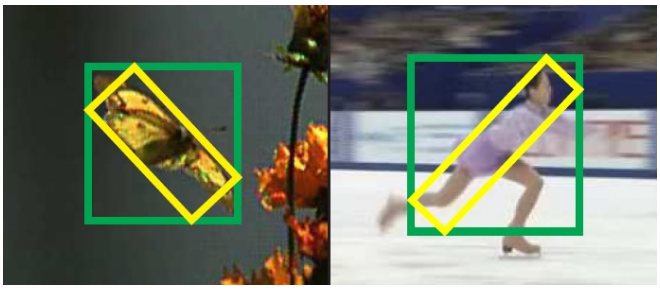
Some of the aspects explained in the previous section are extremely frequent in a lot of videos and in the databases, then are considered as a normal scenario. Others instead are often even not considered in the samples and the majority of the algorithms do not, officially, solve them.

- The conditions that are often respected in object tracking are:
  - The low-resolution images to allow the elaboration even without powerful computation devices such as a smartphone.
  - A moving camera because a great majority of videos are in movement and shaking, except for the video surveillance field where the video camera is placed fixed.
  - The changes of 2D shape and the fast-moving objects should be respected, the robustness on these two points make the algorithms more or less reliable.
  - A short-time total occlusion might sometimes be managed, but it is rarely guaranteed.

- The requirements for this thesis except the traditional ones are:



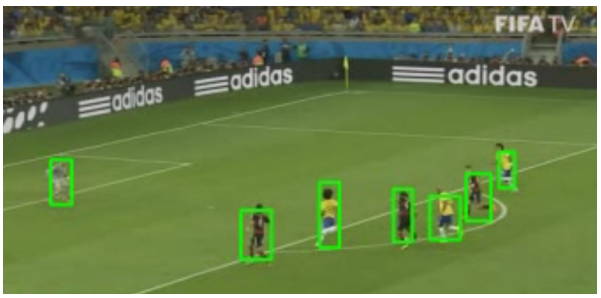
(a) Images with low resolution.



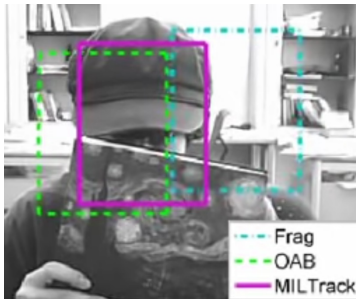
(b) Two subject that can vary their appearance very quickly.

(c) Tracking deals with objects, humans, and even animals.

(d) The limbs of Usain Bolt seems to disappear.

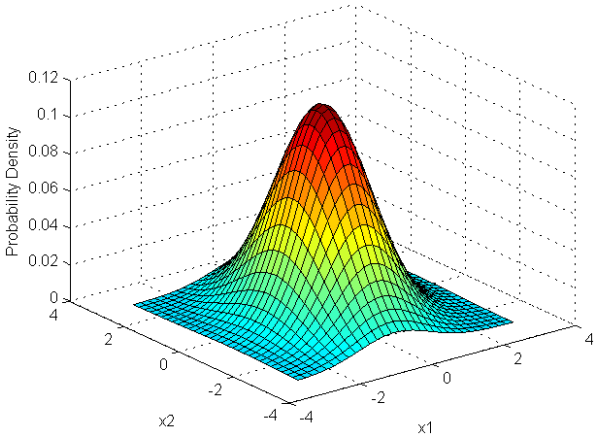


(e) A soccer match action where all the players are tracked.



(f) A face partially hidden by a hat and a book.

(g) The new location is close to the older one.



(h) The probability of where the new location is respect a multivariate normal distribution.

(i) A motorcycle running on a street, is totally hidden by a tree for multiple frames.

Figure 4.1: Some visual examples of the tracking challenge conditions.

– Long-time occlusion. In our scenario where the robot physically follows a person, the leader can disappear behind a corner and will be hidden as long as

the robot reach it.

- Real-time processing. We have decided that to understand the movements in the real environment is sufficient a processing speed of 5 FPS.
- Long-term video. The algorithm is designed to last for a very long period no explicit bounds exist, since that the drift problem has been solved.
- Subject limited to people. We are not interested in following animals or vehicles, even if extend the algorithm to them only require to change the object detectors setup and the internal database of images to train the recognition procedure.

4.2 Principal known algorithms

In this section will be presented a set of algorithms that perform well to solve the tracking task.
Differently, from object detection, the number of existing methods for tracking is much wider. This happens due to the high variability of the problem. The methods shown below represent a trade-off in term of speed and reliability.

4.2.1 MIL (Multiple Instance Learning) tracker

The MIL tracker[19][20] is an extension of the older **BOOSTING Tracker**[21], both method are based on an **online classifier**. "*Online*" means that the classifier is trained "*on the fly*" during the execution of the algorithms and not in advance. This type of training does not allow to use thousands of images but is typically done with a very few. An application of this method is presented in Figure 4.2.
The idea of the online classifier is to trust the initialization of the tracker and use the initial bounding box as the first training sample. The negative samples are then generated taking rectangles that do not overlap the positive example. The classifier learns during the execution to recognise the tracked subject as positive and the rest as negative.
The frames after the first one are elaborated similarly. The positive is searched around the last known position and the classifier assign a probability to each proposal. The box with the highest score is chosen as positive and it is used to continue the training of the classifier.
The novelty of MIL compared to BOOSTING comes here (a visual comparison is shown in Figure 4.3).
MIL instead of using only the positive sample to fit the online classifier, create a bunch of bounding boxes proposal around the positive sample, called **bags**. All these boxes should contain the subject and one could even be perfectly centred on it. The training is done with the "Multiple Instance Learning" that takes the bag of positive proposals and select the best one (the more centred one) to improve the classifier. In the end, the instance is trained with only one box that was chosen starting from a set of good alternatives, and not with all of them. The negative samples are then generated as for the first frame.

4.2.2 KCF (Kernelized Correlation Filters) tracker

The KCF tracker[24][25] is an additional extension of MIL tracker.
The key idea is that the sampled images are similar due to the subject that is repeated or thanks to the background that does not change extremely fast compared to the FPS of the video. This repetition of similar patterns can be used to optimize the operations and speed up the computation. The technical improvement comes from the application

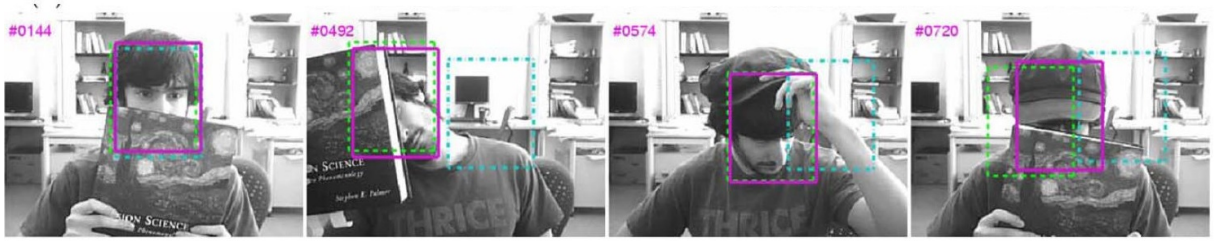


Figure 4.2: A sample of the MIL classifier tracking the face of the author while is partially occluded by a book. MIL tracker is in purple, while cyan and green are respectively FragTrack[22] and Online Ada-Boost[23].

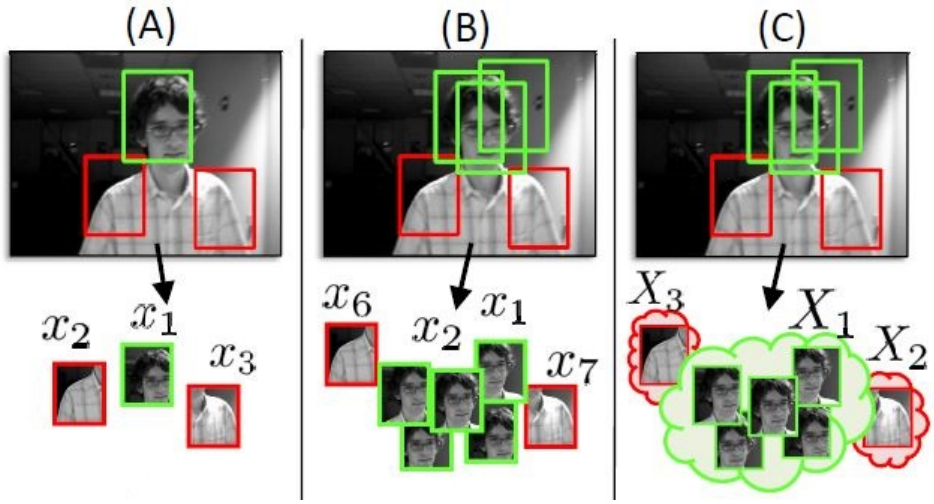


Figure 4.3: A comparison of how positive and negative samples are selected, to train the online classifier. In A the selection of BOOSTING tracker, the positive sample is used. In B a set of proposals generated around the positive are all used to train. In C the selection of MIL tracker, bags of samples are used and only one proposal is extracted and accepted from each one.

of the **Fast Fourier Transformation**. That allows applying the elaboration of images in an efficient manner.
The novelty introduced with KCF allows this algorithm to outperform both BOOSTING and more important MIL, in term of accuracy and speed. The weakness of this chain of three methods is the full occlusion, none of them is able to deal with total occlusion that always causes the tracking failure.

4.2.3 Median Flow tracker

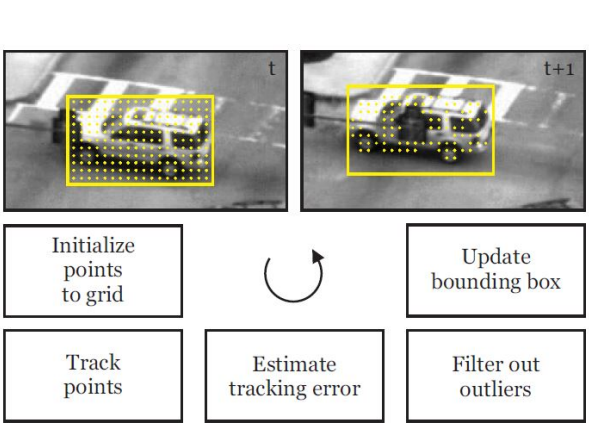
The Median Flow tracker[26] is a reliable method that locates the subject according to its trajectory. The key idea is that the algorithm recognises points in two subsequent frames, these points should be the same physical element in the real space.
The overall procedure is shown in Figure 4.4. The first step (Figure 4.4a) consist of creating a grid of points on the initial bounding box, and then locate these points in all the future frames. This connection through the frames helps to know the exact motion model of the tracked algorithm. The **motion model**(MM) is the combination of actual position (x, y) and velocity defined with the angle or direction of motion(θ) and the module of the speed(s). Essentially knowing how the subject is moving help to predict where it

will be in the near future.

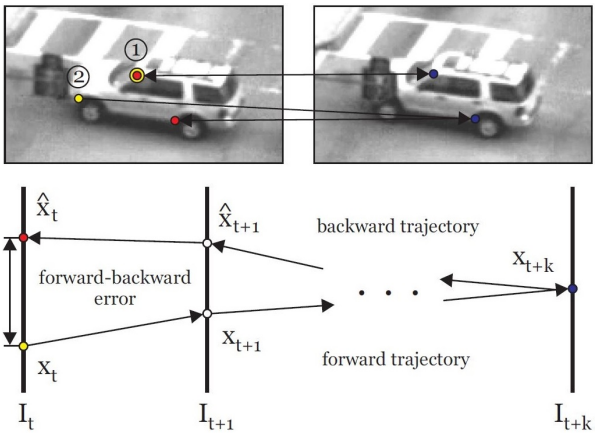
$$MM = ((x, y), (\theta, s))$$

The interaction of the frames works as follow. Every time that a new frame is added the knowledge of the motion model suggests where the subject can be located. Then, the key points are researched in this new image. Ones are found it is fundamental to check the consistency of the trajectories (Figure 4.4b). Each new point is associated with the most similar already known physical point and vice versa. If this double matching is correct and the two points refer to the same physical element (point 1 in the figure), the trajectory is confirmed otherwise (point 2 in the figure) there is a misalignment in the trajectories. In Figure 4.4b point 2 is firstly, forward pass, associated with the front wheel, in the frame after is linked to the back wheel since the other one is hidden from the street signal. The backward pass links the back wheel again on itself. This misalignment of the connection is recognised as an error and so that point cannot be trusted. If no point can be trusted the tracking fails.

Thanks to this double-checking procedure, the Median Flow tracker is a method that is able to well recognise when the tracking has failed to follow the subject. Unfortunately, the requirement to match key point over and over in the frames reduce the capability of the algorithm to manage scenarios where the subject appearance change too much.
For this reason, this algorithm is not good for tracking high deformable subjects such as animals and humans.



(a) The first bounding box is divided into a grid of points, and a portion of them are recognised in the next frame. Below, the main steps of the algorithm.



(b) The points found in two consecutive steps are compared to check if the forward and backward trajectories are the same and then it can be trusted.
Point 1 is accepted while point 2 is rejected.

Figure 4.4: The overall procedure of the Median Flow tracker computing forward and backward trajectories to precisely locate the target.

4.2.4 CSRT (Channel and Spatial Reliability Tracker)

The full name of the method is Discriminative Correlation Filter with Channel and Spatial Reliability (DCF-CSR)[27]. Such as KCF (Section 4.2.2) and others even this method is based on correlation filters.

A **cross correlation filter** is a technique that aims to localize into an image the exact position of another one. A representation of the cross-correlation usage done by CSRT is shown in Figure 4.5. In details, the portion of the last frame, delimited by the last known bounding box, is elaborated with multiples correlation filters, each one producing a different output. These elaborations simulate possible changes in the appearance of the subject. Design good filters are fundamental to well match the variability of the tracked subject. The filter outputs are modified images of the last bounding box cropped area.
The output of each filter (an image), is moved along the full picture pixel by pixel (learning stage: Figure 4.5 left), to check which portion of the entire camera view is more probable the subject that we are looking for. The result of this scan is a confidence map that should present a peak in correspondence of the new position of the tracked subject.
All the filters can then be summed up together (localization stage: Figure 4.5 right) to highlight the proposal of each one and comes out with the final response. This response shows exactly where the subject is placed in the new frame. A visualization of a confidence map applied to the original image is shown in Figure 4.6.

The characterization of CSRT is focused on the type of correlation filters used, with the idea of using multiple of them and combine the results at the end to produce a more reliable localization. The very high accuracy that this algorithm offers is compensated by the low FPS rate that it achieves.

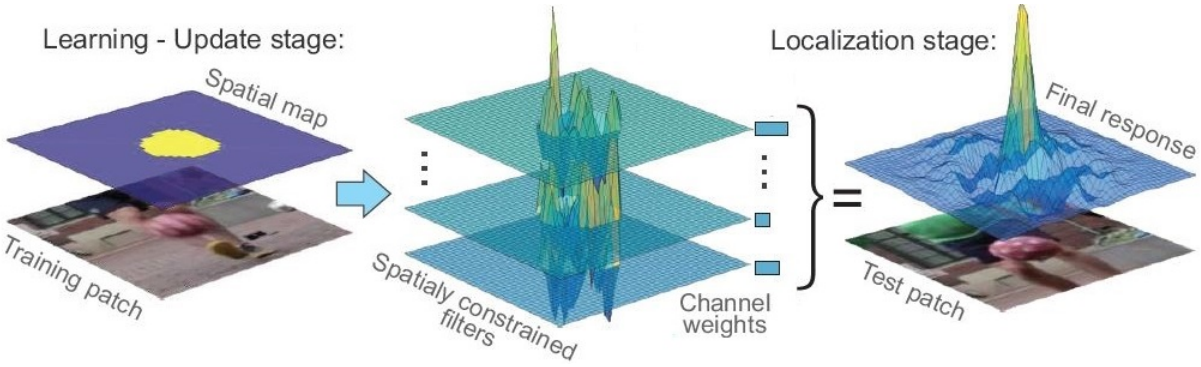


Figure 4.5: The overall procedure of the CSRT algorithm. The learning stage is composed of multiple correlation filters that are applied to the input frame. Each filter produces as output a confidence map that highlight where the subject should be. The localization stage combine all these confidence maps to produce the final response that precisely locate the subject.

4.2.5 MOSSE (Minimum Output Sum of Squared Error) tracker

The MOSSE algotihm[28] such as KCF (Section 4.2.2) is a method whose strength lies into mathematical smart choices instead of a complex high-level logic like MedianFlow (Section 4.2.3).
The novelty is introduced with a new correlation filter, called MOSSE. It can be applied to the input frames and precisely locate the variations fundamental to understand the movement of the subject.
Despite the original article boasts robustness against variations in lighting, scale, and non-rigid deformations, this algorithms is not so reliable as appears. On the other hand, the strength of this method is the extremely fast computations (FPS rate) that outperform all the other trackers presented in this section.

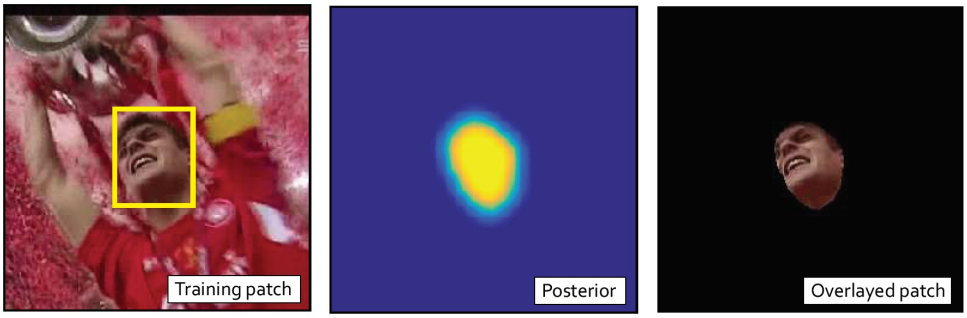


Figure 4.6: A visualization of the confidence map applied to track the face of the man in the image. Left the original image, centre the confidence map in 2D, right the cropped image according to localization.

4.2.6 GOTURN (Generic Object Tracking Using Regression Networks)

GOTURN[29] is a tracker based on neural networks. Differently from MIL and KCF (Section 4.2.2), this method is not based on an online NN, such as the online classifier, but it is based on an offline CNN.
An **offline NN** is a traditional NN that is trained on thousands of data, couples of images, in this case, producing a trained model. The process is done in advance, before the effective use, and not "on the fly" while running the tracker. The generated model is used at run time to know how to respond to an input value. The big advantage of offline methods is that the training procedure is the slower section, for this reason, an online classifier could not perform at very high FPS rates.

The general workflow of the tracker is shown in Figure 4.7. The tracker takes as input a frame at a time and compares it always to the previous frame. This choice simplifies and standardize the input to empower the potentiality of the CNN but makes the algorithm to have no chance against total occlusions even for one single frame. The CNN takes as input two squared images cropped from the frames. The crop on the previous frame is a bounding box centred around the last known position with some margins that will contain even the location in the frame after. The current frame is cropped based on the same bounding box, but in this case, the subject is not centred in the square.
Then, both squared images are processed with two independent stacks of convolutional layers, followed by three fully-connected layers. The final output is four values representing the top-left, and bottom-right corners of the bounding box, centred on the subject in the squared image of the current frame. Some examples of the application of the CNN are shown in Figure 4.8.

4.2.7 TLD (Tracking-Learning-Detection)

Differently from all the other methods TLD[30] aims to be a complete tracker able to deal with extremely complex scenarios. If the previously presented trackers have no chance to manage a long-time total occlusion and a long-term tracking this method is able to overcome both the problems. A sample showing potentiality is in the Figure 4.10.
The foundation principle is that TLD is not a single algorithm but it is a combination of three. The interaction of these three parts is shown in Figure 4.9. The key idea to overcome the re-identification problem that occurs after a total occlusion, or the drift problem that happens along a large video is to re-initialize the tracker often. In fact, the

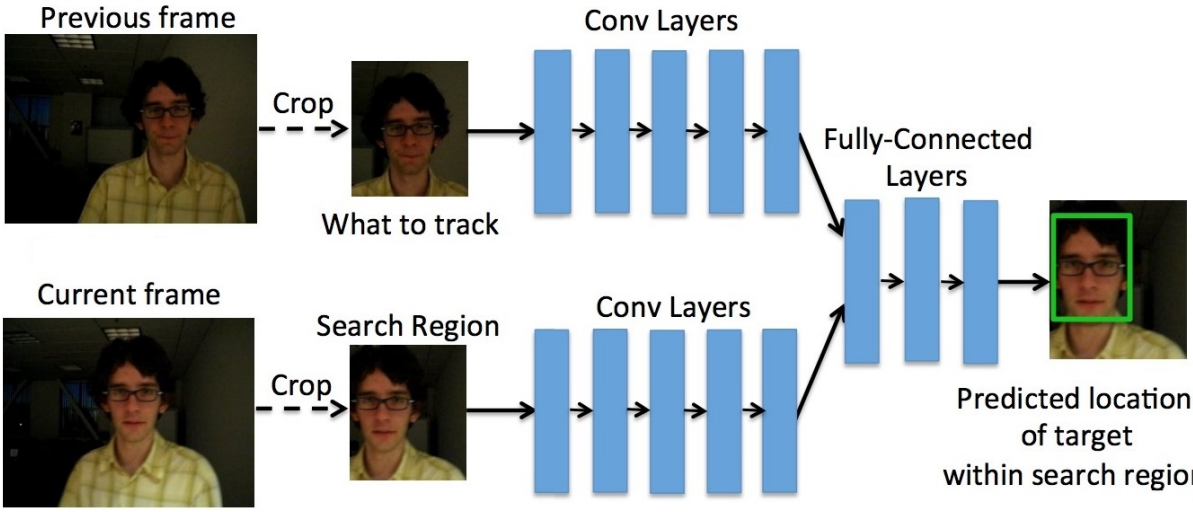


Figure 4.7: The overall procedure of the GOTURN method. On the left, the frames coming from the camera are cropped with the same square, that is centred on the subject in the previous frame (above). The images are elaborated with CNN to produce the output bounding box, that highlight where the subject is in the current frame.

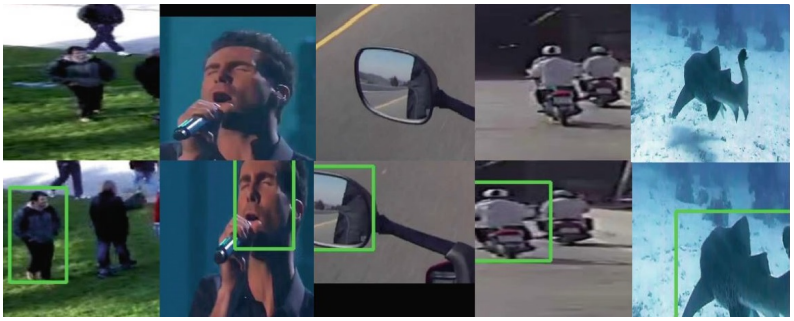


Figure 4.8: Samples of application of the CNN of GOTURN. The two squared images coming from previous (above) and current (below) frame are fed into the network. The answer is the green bounding box that locates the tracked subject on the new frame.

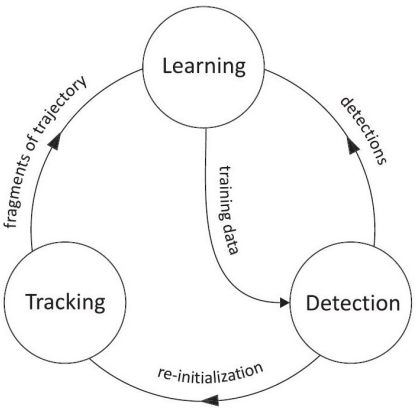


Figure 4.9: The interaction of the three foundation methods of the TLD algorithm.

tracking (**T**) of TLD aims to solve short-term video clip. When a small problem occur the detection (**D**) try to locate the subject back again. While these two situations take turns, the learning procedure (**L**), extract the key elements that recognise uniquely the subject and understand how to precisely locate it inside the frame.

The trade-off to use this extremely flexible structure is paid with a not high FPS rate. But, the bigger problem of this method is the huge quantity of false-positive predictions. The learning procedure starts with a few samples, meaning that error in the beginning phase can occur easily. The failure in the first phase causes wrong learning that will produce more and more errors later on.
Despite the good potentialities, this algorithm is not reliable for the traditional tracking task.

Figure 4.10: A sample that shows the potentialities of the TLD algorithm, dealing with a total occlusion while tracking a motorbike with dramatically change of size. The red dot means that the subject cannot be found.

4.3 Which tracker could be chosen

In the previous section, we have presented seven different tracking algorithms, each one with his own strength and weakness. These methods are the ones that were explored during the thesis work, but many others exist. But which one is more suitable for our own use.
The tracking task that we aim to solve is a long real-time sequence with long-lasting total occlusions, as explained in Section 4.1.4. But this task is solved with a combination of detection, tracking and recognition, so the internal tracking challenge is much simpler.
The requirement is to solve the baseline tracking task (Section 4.1), in addition, deal with changes of shape and partial occlusions. Considered the explained methods and known their speed performances (shown in Table 4.1), here it is what we have chosen. Note that the underlined methods are the best trade-off choices.

- **MIL tracker** is not a good choice because it runs at few FPS, and exist a newer version that outperforms both its speed and accuracy.
- **KCF tracker** is the newer version of MIL. It is one of the fastest methods and it is also reliable. It suffers the rapid change of appearance and less important for our scenario, it does not manage total occlusions.
- **Median Flow tracker** works well only on not deformable or rigid subjects. But we are dealing with humans so is a bad choice.
- **CSRT** is the most reliable method, could even manage short total occlusion. Despite the low FPS rate is a great choice, in fact, our goal is a tracker running at around 5 FPS that is way less than the speed of CSRT.
- **MOSSE** is focused on pure speed. It is not the most reliable method but can be a good choice to try to reach the highest FPS rate for the general challenge.
- **GOTURN** is a reliable method, running at a good FPS rate. It is not integrated at the moment in the project but can be a great choice for future improvements.
- **TLD** is a too complex method. It is able to solve long total occlusion but easily fails on simpler scenarios, by proposing a lot of false positive. It is not reliable at all for our purpose.
  TLD is based on a principle that is similar to the one proposed in this thesis. The integration of tracking and detection linked together with a third procedure: learning in case of TLD and person recognition for this project. Both algorithms

Algorithm	MIL	KCF	MedianFlow	CSRT	MOSSE	GOTURN	TLD
FPS	9	38	40	15	56	20	10

Table 4.1: An overview of the FPS rate of the trackers described. The performances were measured on an Intel Core i5 CPU, and on an Nvidia Jetson TX2 GPU. The speeds measured are almost the same.

aim to solve the total occlusion problem and the drift problem with a reinitialization of the tracker performed with an object detector. The key difference is the existence in this thesis of the slow start phase presented in Section??? TODO.

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] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[15] 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.

[16] 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.

[17] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[18] Wordnet graph. `https://wordnet.princeton.edu/`.

[19] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *2009 IEEE Conference on computer vision and Pattern Recognition*, pages 983–990. IEEE, 2009.

[20] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence*, 33(8):1619–1632, 2010.

[21] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *Bmvc*, volume 1, page 6, 2006.

[22] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *2006 IEEE Computer society conference on computer vision and pattern recognition (CVPR'06)*, volume 1, pages 798–805. IEEE, 2006.

[23] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *Bmvc*, volume 1, page 6, 2006.

[24] João F Henriques, Rui Caseiro, Pedro Martins, and Jorge Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European conference on computer vision*, pages 702–715. Springer, 2012.

[25] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost Van de Weijer. Adaptive color attributes for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1090–1097, 2014.

[26] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *2010 20th International Conference on Pattern Recognition*, pages 2756–2759. IEEE, 2010.

[27] Alan Lukezic, Tomas Vojir, Luka ˇCehovin Zajc, Jiri Matas, and Matej Kristan. Discriminative correlation filter with channel and spatial reliability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6309–6318, 2017.

[28] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 2544–2550. IEEE, 2010.

[29] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[30] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2011.