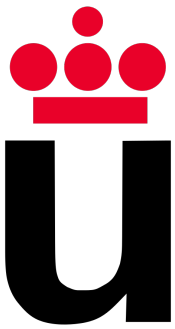


Master's thesis

An hybrid architecture for multiple people tracking

Marcos Pieras Sagardoy

Móstoles, 27 de Junio de 2016

Contents

1 Introduction

Deep learning has rised by drastic improvements over reigning approaches towards the hardest problems in Artificial intelligence (AI), massive investments from industry giants, and exponential growth in research publications. Deep learning is a tool inside the machine learning toolbox, the goal is to make machines learn.

The first incursion was made by Frank Rosenblatt, the Percetron [**?**]. Rosenblatt conceived of the Percetron as a simplified mathematical model of how the neurons in our brains operate. This model of the neuron built on the work of Mcculoch-Pitts [43], who showed that a neuron model can model the basis OR/AND/NOT functions. Which in the early days of Artificial intelligence, was great, because the predominant tought at that time was that making computers able to perform formal logical reasoning would essentially solve AI. However, the Mcculoch-Pitts model lacked a mechanism for learning, which was crucial for it to be usable for AI. This is were the Perceptron exceeled, Rosenblatt came up with a way to make such artificial neurons learn, inspirated by the Hebb's Rule. This learning method was, if the output of the perceptron was low, increase the weights, and otherwise decrease the weights if the output is too high. Also, another researchers came with ADALINE [71] learning procedure, they used the signal before the activation function to compute the derivative, how much the error changes when each weight is changed can be used to drive the error down and find the optimal weights values. Similiar as we train the networks nowadays.

Researcher were really excited about this idea of Connectionism: that networks of such simple computational units can be vastly much more powerful and solve the hard problems of AI. But in 1969, Minsky and Papert published an analysis on the limitations of perceptrons, named Perceptrons [45]. The biggest cristicm was that a pereceptron could not learn the simple boolean function XOR beacause it is not linearly separable. However, they stated that it can be learnt with multiple layers percepron but the learning procedure did not work for multiple layers. After, this book the interest on Neural networks decreased, and initlizate a period called *AI winter*, AI shifted to logic programming and commonsense reasoning.

This period lasted till 1986 when Rummelhart, Hinton, and Williams published the algorithm of backprogation [52], which specifically adressed the problems discussed by Minsky in Perceptrons, a method to train multiple layer neural nets. With this discovery in 1989, LeCun showed a real world application, recognized handwritten digits [33]. The architecture of this model was a convolutional neural network. It was inspired by the Neurocognitron [12] of Fukushima, which took ideas from studies of the brain. In particular the studies of Hubel and Wiesel, they propose that the visual cortex is formed by a hierarchical model, primarilly for simple cells who respond for simple structures and then complex cells respond to a more complicated feature. As we can observe in figure 1, in the lower layer, the network learns Gabor like features, and while going upwards, the networks learns more abstract concepts.

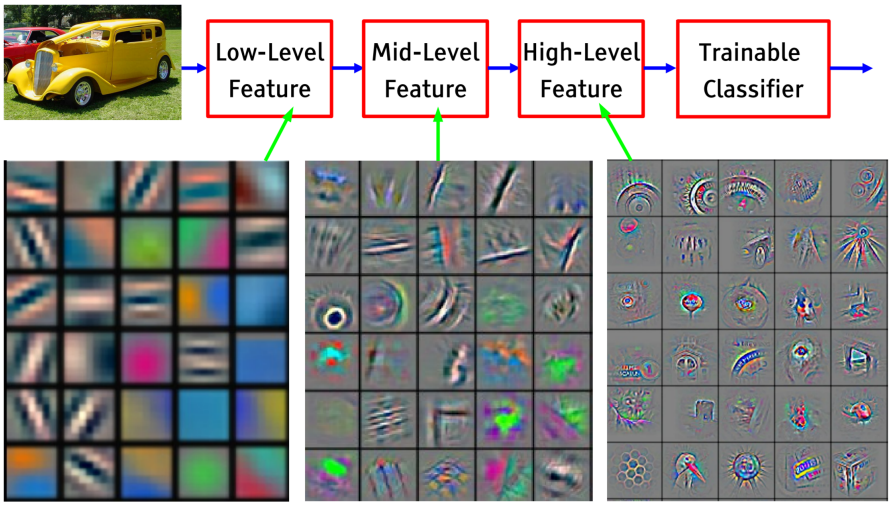


Figure 1: Representation of a Convolutional neural network.

But this approach did not scale to larger problems, the biggest source of problems was the vanishing gradients problem. When the backpropagtion gradients backpropagets trough the network, in some nodes the local gradient is very low (in the extreme of the sigmoid functions) the signal vanishes or saturates, and by the 90s other techniques became the method of choice, like support vector machine (SVM). Altough some progress was made for other kinds of problems.

- Unsupervised learning. This types of architectures are used to finding a smaller representation of some data from which the original data can be reconstructed, it is useful for compression, visualization, and classification. One example of this architecture with neural networks, is the Restricted Boltzman machine [1], developed by Hinton.

- Reinformcent learning. The goal of this type of learning is to learn how to make good decisions, it requires rewards, not labels. One example of this sort of systems, is the TD-Gammon [61], a neural network that learned to be a backgammon player.

- Robotics. This was another field separate from Machine Learning where neural nets were very useful. A major example of early neural net use for robotics came from CMU's NavLab in 1989. the neural net learned to control the vehicle using steering data recorded while a human drove [48].

- Recurrent neural networks. Plain neural networks could not process sequences due to they do not have memory, they need mechanism to remmeber the pasts ouputs. With memory, it can process sequences like audio or text. One approach to this, by Waibel [67] in 1989.

In 2006, there was a breaktroguht [21], Hinton realized that a neural network with many layers really could be trained well, if the weights are initilzaized in a clever way. The basic idea was to train each layer one by one with unsupervised training (like an autoencoder architecture) and finally stack all together and train it in a supervised way. But the big breaktrought came in 2012, when a neural network won the Large scale visual recogntition challenge.

Although, this improvements, the big breaktroguht came in 2012, when AlexNet [29] beat the state of the art in the ImageNet challenge, an image classification challenge, where the error rate was 15.3% whereas the winner of the previous year was 26.3%. In the plot 2 we can observe the advance in the state of the art of the ImageNet challenge with the inclusion of deep learning techniques.

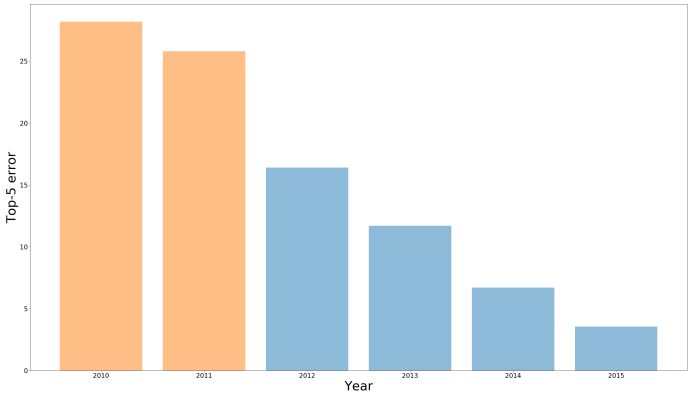


Figure 2: Classification error in ImageNet challenge.

The emergence of these techniques were the culmination of decades of research but the step forward was dued by three aspects:

- **Appearance of large and high quality dataset**, The increasing size of the dataset.
- **Parallel computation**, The increasing of computing capabilities helped train larger models in least time.
- **Optimization details**. With the discovery of the proper inizialization and activation functions nets can be trained.

1.1 Tracking with deep learning techiniques

As we said above, Deep learning techinques has increased the perfomance in all the aspects of computer vision, as well as Tracking, the main ways to apply deep learning techniques to tracking are the following [20]:

- **Online training for tracking**. Starting from the first frame of a video, a tracker will sample patches near the target object, and they are used to train a foreground-background classifier, and this classifier is used to score patches from the next frame to estimate the new location of the target object. These methods showed a state-of-the-art perfomance results. Unfortunately, neural networks are slow to train, therefore the speed of the method is reduced [46] [6].
- **Model based trackers**. These methods use a specific class classifier and there is not need to train it online. So, these methods use a neural network to extract instances of the frames and then linked with temporal restrictions.

- **Siamese based tracking**. In this approach, many candidate patches are passed through the network, and the patch with the highest matching socre is selected as the tracking output [60].
- **Tracking as regression**, these methods are an extension of object localization using neural networks, these methods given an image containing an object predict the bounding box which contain the object in every frame [20].
- **Tracking with RNN**. From the output of an object detector, these tracking algorithms model the sequence of movement of the objects using an recurrent neural network [53].

2 Objectives

Next we explain the objectives and the requirements of this thesis.

2.1 Problem description

The main objective of this thesis is the develping of an algorithm of tracking, particularly tracking multiple pedestrian on real scenes. We evaluate our solution on the Multiple Object Tracking challenge 2016, which include a variety of evaluation sequences.

- Perfomance measures, we use the measurment of the MOT [32] challenge to evaluate our algorithm, we desire to have the highest possible score.
- Speed, our solution should faster as posibles.

When we finish our algorithm we will upload the results to MOT's website and compare with the state of the art.

2.2 Requirements

Besides of the previous stated objectives, the solution must guarantee the next requirments:

- The solution will make use of the developing platform JdeRobot, which is the developing environment of the *Grupo de Robótica* of the *Universidad Rey Juan Carlos.*
- The software will run on the GNU/Linux Ubuntu 16.04 environment.

3 Theoretical basis

In this section, we will explain the fundamenal concepts which is based our thesis. We propose a multi people tracking algorithm, in order to do so, we represent each person with a bounding box, in this bounding box we extract some features and study how they move through the frames, based on the the movemenent of these features we will infer the movement of the bounding box, therefore, the movement of the person. So in the section 3.1 we will explain how to extract the bounding boxes, in section 3.2 how to extract the

movement of the bounding boxes, and finally in section 3.3 we explain how to solve people identification incongruities. We can observe the procedure in the next figure 3.

(a) Detections. (b) Features Points. (c) Optical flow. (d) Identity check.

Figure 3: Algorithm's stages.

3.1 Object detection

In object detection too, the emergence of the neural networks has supposed a turning point. As we can observe in 4, the mean average precision, has almost doubled since the appearance of deep neural networks.

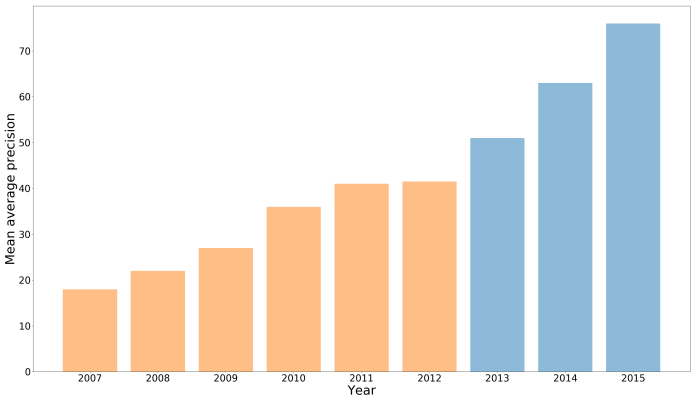


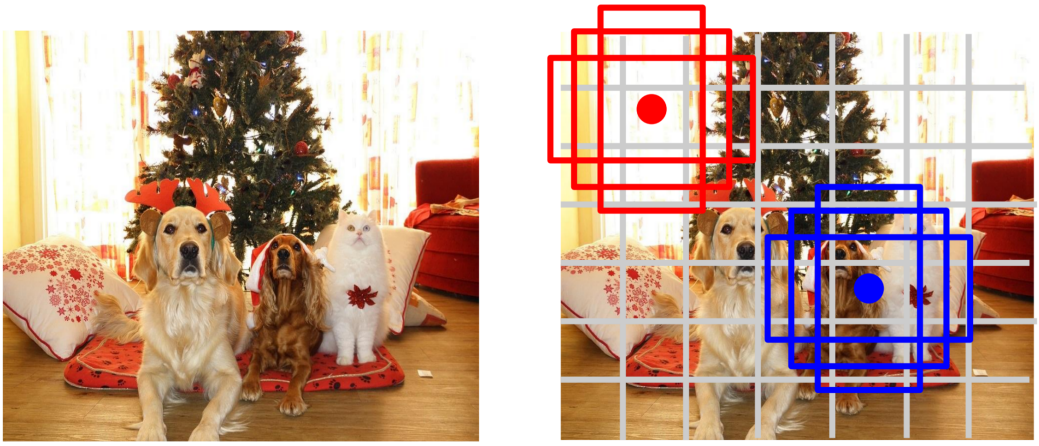
Figure 4: Mean average precion over the years in PASCAL dataset.

Present deep learning object detectors are based on three main family of archictectures [23], named by the reference algorithm of the category: FasterRCNN, SSD, and RFCN, the characteristics of these systems are:

- Faster RCNN [50], it is the last output of a tryology of detectors developed by R. Girshick and his team. Which are called Region-Based object detecors. They work as follows: Use some mechanism to extract region of an image that are probable to be an object and then classify those proposals with a CNN. The first paper to do so, was [14], and supposse a breaktrough in the field, increasing the precision of the state of the art of those days. But, it had a messy pipeline, slow and difficult to train. Later on, they developed [13], in this paper they applied the region proposal algorithm in the cnn feature map, so, they avoid to compute the features for each

proposal. They increase the speed and it could be trained much easily. Finally, they showed FasterRCNN [50], in this algorithm, they eluded the external region proposal algorithm and they implemented a CNN to compute those proposals. This CNN share parameters with the main net and they saved a lot of time. This network, has become the standard object detector with CNN. With the association of novel net architecture like ResNet [19], Inception [59], and [27] they have won all the contests.

- SSD, it stands fot Single shot multibox detector. These family of method differs from previous ones considering that these treats the problem of object detection as a regression problem. So, they are called Regression-based object detector or single shot object detector due it does not have a region proposal algorithm, they classify the image with one mechanism. The maximium exponent of these algoirthms are [49] and [39]. These work as follows, they discretize the image at the features level in a fixed grid and for each grid it predicts a class and some number of bounding boxes with different shapes and sizes. It merges all, and apply a Non-Maximum supression algorithm to obtain a set of detections. We can observe this process in 5. In addition, they apply this process in a multiresolution scheme as we can observe in 6 to deal with objects of different sizes.



(a) Input Image. (b) Divide image into grid.

Figure 5: SSD detector scheme.

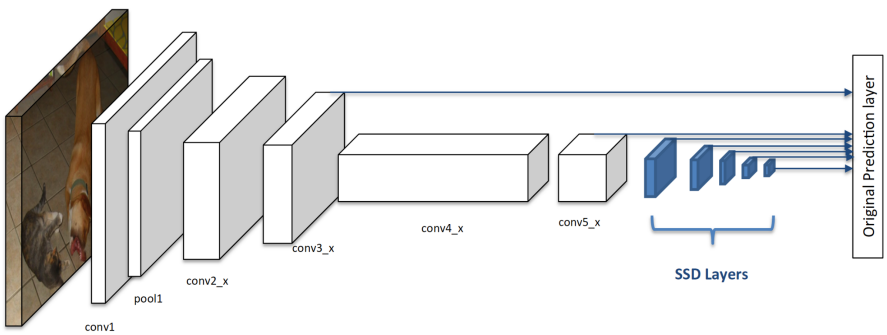


Figure 6: SSD architecture.

- RFCN [5], it stands for Region-based fully convolutional network and it was de-

veloped by the same authors of SSD. They noticed the lacks of the SSD, the SSD algorithm computes the object detector on the feature map, and at this level the features have a low spatial resolution, this involves do not detect small objects. So the authors inspired by the fully convolutional architectures, upsample those feature maps and compute the object detector like the SSD algorithm.

In the survey [23], they compared the different methods including changing the features extractors (ResNet, Inception, VGG) and they measured the precision (mean average precision) and computing time. This results are showed in 7. The conlcusion are as follows, SSD is the fastest detector, RFCN it has the best balance between speed-acurracy, and FasterRCNN, is the most acurate detector altough is slower than the other ones.

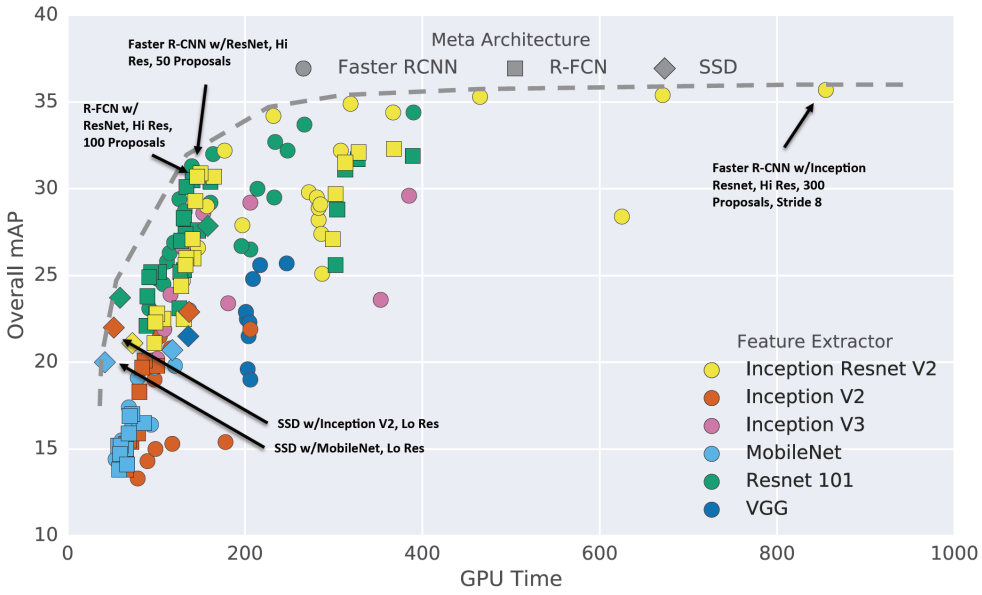


Figure 7: Comparision architectures.

We will finish off our review with a numeric comparision of the methods, as we can observe in the table 1. This information is extracted from the original papers with their implementation, all of them are trained with the union of the training set of VOC07, VOC12, and COCO, and subsequently evaluate on VOC07 test set on a Nvidia Titan X GPU.

	mAP	**mAP_person**	**FPS**	**Proposals**
RCNN	66	64.2	0.077	2000
FastRCNN	70	69.9	6.7	2000
FasterRCNN	85.6	82.3	7	6000
SSD300	81.2	81.4	46	8732
SSD512	83.2	84.6	19	24564
YOLO	66.4	63.5	45	98
YOLOv2	78.6	81.3	40	-
RFCN	83.6	-	10	-
PVANET	84.9	-	31.3	300

Table 1: Summarize of the object detectors.

3.2 Tracking fundamentals

As we said above, we need to find features in the image and study how they move to estimate the movement of the targets.

3.2.1 Features

Our goal is to find points in an image, that can be found in other images and then compute some information, in this case, the movement. The characteristics of good features are:

- Repeatability, the same feature can be found in several images despite geometric and photometric transformations.
- Matchability, each feature has a distinctive description, thus easy to find.
- Efficiency, few features have to compact much more possible information.
- Locality, a feature occupies a relatively small area of the image, so therefore it is robust to clutter and occlusion.
- Perfomance, computation speed of features is a critical parameter.

Features points are used in all sort of operations in computer vision: Image alignment, 3D reonstruction, Motion Tracking, Object recognition, Index database retrieval, robot navigation and so on.

Looking at the figure 8, the flat patch, is a patch without texture and impossible to localize. Patches with large contrast edges are easier to localize, although straight lines segments at a single orientation suffer from the *aperture problem*, are also impossible to localize. Finally, patches with large gradients in at least two different orientations are the easiest to localize.

(a) Flat region. (b) Edge region. (c) Corner region.

Figure 8: Types of patches.

These intuitions can be formalized by looking at the simples possible matching criteron for comparing two images patches, their weighted summed square difference:

$$E(u) = \sum_i w(x_i)[I(x_i + u) - I(x_i)]^2$$

where $I(x)$ is the image, $I(x+u)$ is the shifted image, and $w(x, y)$ is a window function like a box or gaussian kernal around the pixel, and the summation i is over all the pixels in the patch. Then we are looking for points, which if we move according to u we have a change.

When performing feature detection, we do not know which other image locations the feature will end up being matched against. Therefore, we can only compute how stable this metric is with respect to small variations in positions Δu by comparing an image patch against itself:

$$E(\Delta u) = \sum_i w(x_i)[I(x_i + \Delta u) - I(x_i)]^2$$

Using a Taylor series expansion of the image function $I(x_i + \Delta u) \approx I(x_i) + \nabla I(x_i) * \Delta u$ we can approximate the expresion as follows:

$$E(\Delta u) \approx \sum_i w(x_i)[I(x_i) + \nabla I(x_i)\Delta u - I(x_i)]^2$$

$$E(\Delta u) = \sum_i w(x_i)[\nabla I(x_i)\Delta u]^2$$

With algebraic notation it transforms to:

$$E(\Delta u) = \Delta u^T M \Delta u$$

where $\nabla I(x_i) = [I_x, I_y](x_i)$ is the image gradient and M is the second moment matrix:

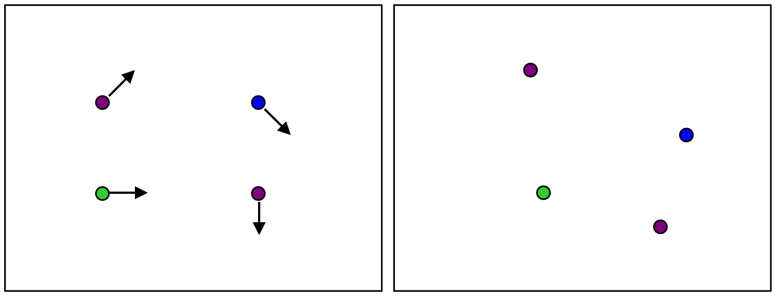
$$M = \begin{pmatrix} I_x^2 & I_{xy}^2 \\ I_{xy}^2 & I_y^2 \end{pmatrix}.$$

Computing the eigenvalue descompostion of this matrix, shows the directions of the fastest change, thus a measure of the *cornernes*. There are several algorithms that use in different ways this eigenvalues:

- Harris [18], they propose a corner detection response function. So for each pixel, they compute a matrix M and with it, they compute the function R, $R = det(M) - a\, trace^2(M)$. if R is large, that pixel is a corner, if R is negative with larger magnitude, it is a an edge, and if R is small it is a flat region. So the they a treshold to classify those pixels as a corner.
- Shi-Tomasi [55], they define the *cornerness* in another way. The image has a maximum value (e.g. 255), so λ_1, λ_2 also have an upper bound, then it is only necessary to check that $min(\lambda_1, \lambda_2)$ is large enough, this is how they define *cornerness*. This feature is called good features to track, because the authors defined a *good* features those whose motion can be estimated reliably, and they reached the same conlusions as Harris. This method is implement in the OpenCV's routine `goodFeaturesToTrack()`.

3.2.2 Motion estimation

Now, we have invariant points, we want to estimate the motion of those points. In order to do so, we compute the optical flow. This is the apparent two-dimensional motion of birightness pattern in the image. In the next figure 9 we visualized this idea.



(a) $I(x, y, t)$. (b) $I(x, y, t + 1)$.

Figure 9: Optical flow example.

So, the question is: How do we estimate pixel motion from image $I(x, y, t)$ to image $I(x, y, t + 1)$. We need to solve the pixel corresponde problem. Given a pixel in $I(x, y, t)$, look for nearby pixels of the same color in $I(x, y, t + 1)$. Solving this problem is what is referred as the optical flow problem. By nearby pixels and same colour we have two assumptions:

- Colour constancy: a point in $I(x, y, t)$ looks the same in $I(x', y', t+1)$. For grayscale images, this is called *Brightness constancy constraint*. Stated in mathematical formulation:

$$I(x, y, t) = I(x + u, y + v, t + 1)$$

$$0 = I(x + u, y + v, t + 1) - I(x, y, t)$$

- Small motion: Subsequents points do not move very far, so we can estimate the motion by Taylor expansion:

$$I(x + u, y + v) \approx I(x, y) + \frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \textit{higher order terms}$$

Then, combining these two equations, we get:

$$0 \approx I(x, y, t + 1) + I_x u + I_y v - I(x, y, t)$$

where $I_x = \dfrac{\partial I}{\partial x}$, isolating the terms we obtain:

$$0 \approx [I(x, y, t + 1) - I(x, y, t)] + I_x u + I_y v$$

$$0 \approx I_t + I_x u + I_y v$$

In the limit of t, u and v approaches zero (asumption of small motion), so it becomes, what it is called the the *brightness constancy constraint equation*:

$$0 = I_t + I_x u + I_y v$$

If we look closely, we realized that we have two unknowns u, v and one equation. This is an underdetermined system. Intuitively, this means, that locally we can only determine the component of the flow in the gradient direction, the component of the flow parallel to an edge is unknown, this is the called the aperture problem. To recover the motion we need to add some extra constraints. There are several types of constraints to solve this problem:

- **Global constraint**, adding a smooth constraint to the brightness constraint, this new constraints penalizes for changes in u and v over the images, it assumes that the motion fields vary smoothly over the image. This approach was developed by Horn and Schunk [**?**].
- **Local constraint**, locally the motion field is almost the same, so we add the neighbpur pixels to the equation. This approach was developed by Lucas and Kanade [40].

Local constraint

In this thesis we use the Local constraint to solve the optical flow problem. As we stated above, we add a local constraint to get more equations, this assumes that the motion field is the same in the locallity. From the brightness constraint equation:

$$0 = I_t(p_i) + \nabla I(p_i)\,[u\ \ v]$$

Adding the neighborhood equations:

$$\begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -I_t(p_1) \\ \vdots \\ -I_t(p_n) \end{bmatrix}$$

Now, there are more equation than unknows, it is an overdetermined system, we have to solve it with the least squares technique. It is based on the optimization of the function:

$$(A^T A)\, d = A^T\, b$$

Using the image notation:

$$\begin{bmatrix} \sum I_x I_x & \sum I_y I_x \\ \sum I_x I_y & \sum I_y I_y \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

The system has a solution when $A^t A$ is invertible, it will be invertible when is well conditioned, this is when the ratio of the great and the small eigenvalues of the matrix is large but no too much. The matrix $A^t A$ in terms of image formulation is the second order matrix that we stated in the section 3.2.1 developing the *cornerness*, then in order to be solvable it should have a strong gradient in both directions. After checking the invertability, we can solve the problem and extract the motion field:

$$d = (A^T A)^{-1}\, A^T\, b$$

Thus, using the image notation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum I_x^2 & \sum I_y I_x \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_x I_t \\ -\sum I_y I_t \end{bmatrix}$$

In practice motion is large, the assumption that it is small fails, consequently the approach using Taylor expansions. For two reasons, the linearity does not hold, in order to solve it, we apply an iterative refinement, which conists in compute the displacement, apply it to the pixels, and compute it again till it converges. The other one is there are local minimia and it will fail into it. To solve it, we need to utilize a coarse to fine approach, the idea is to use multiresolution to compute optical flow, the basic is that in a low resolution image the motion between pixels is very small and we can compute optical flow.

So, in order to do so, we use image pyramids, this consists in downsample these images to specific resolution, then in top level, we compute the motion field using the previous stated method, then we upsample the motion field and the images, We apply a transformation to one image according to the motion field computed in the previous level and then compute the optical flow between that transformed image and the other image, we apply this algorithm in al the resolutions

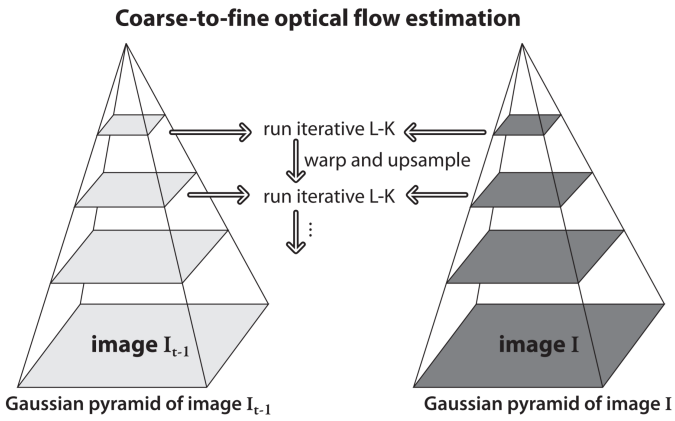


Figure 10: Grpahical .

3.3 Person re-Identification

After running the tracking module we might lose some trackets and in the detection step we might recover them. We need a module to check whether these have the same identity. In order to do so, we studied the methods to recover the identity with deep learning techniques.

Person identification is thoroughly studied in the field of Biometrics, it consists of knowing one biometric characteristic, and comparing with a claiming identity query. Specifically, the topic of pedestrian identification based on images has rised in the last years, this is due to the growth of survillance applications. Also inside the topic of tracking there is a subfield called data association which studies this problem, it consists in matching trackets with further pedestrian detections.

Let's define mathematical the problem, consider G is a gallery composed of N images, denoted as $(g_i)_{i=1}^N$. They belong to N different identities $1, 2, ..., N$. Given a query image q, its identity is determined by:

$$i^* = \underset{i \in 1,2,...,N}{\arg\max} \quad sim(q, g_i)$$

where i^* is the identity of probe q, and $sim(.,.)$ is some kind of similarity function. There are several categories of this similarity function [76]:

- **Hand-crafted systems**. This involves two components, an image descriptor and a distance metric algoritm. The most common image descriptors are those used in computer visiont too, like colour [16], texture [44], SIFT [74], bag of word [65]. The general idea of global metric learning is to keep all the vectors of the same class closer while pushing vectors of different classes further apart. The most commonly used formulation is based on the class of Mahalanobis distance function [30], [70]. Other works focus on learning discriminative subspaces [37].
- **Deep learning techniques**. Two types of CNN models are commonly employed in the community, the first is the classification model as used in image classification, the ouptut is identity label, and the second is the siamese model using image pairs

as input. The major drawback of the classification models is that they need a great quantity of training data by category, and most of the identifications datasets only provide a few examples for identity. So currently methods focus on siamese model.

The main differences between them, is that in hand-crafted methods feature representation of the data and the metric are not learned jointly, instead, deep leraning techniques jointly optimize the representation of the input data conditioned on the *similarity* measure being used [41].

3.3.1 Siamese networks

The first work with siamese architectures were developed by LeCun [3], [4] and they adressed the identification of signatures, besides the siamese networks are used in a variaty of problems like: image recovery [66], feature descriptor [56], comparing patches [73], one shot learning [28], learning visual similarity [2].

Siamese CNN topologies can be grouped under three main categories, depending on the point where the information from each input is combined:

- **Cost function**. Input patches are processed by two parallel branches featuring the same network structure and weights. Finally, the top layers of each branch are fed to a cost function.
- **In-network**. The top layers of the parallel branches processing the two different inputs are concatenated and some more layers added on top of that. Finally, the standard softmax log-loss function is employed.
- **Joint data input**. The two input patches are stacked together forming a unified input to the CNN. Again, the softmax log-loss function is used here.

While the two first approaches have yield good results and historically were dominant, the best perfomance is obtained with the joint data input strategy. As pointed out by [73] and further corroborated by [11], and [31], jointly using information form both images from the first layer tends to deliver a better perfomance.

In the field of person re-identification, the community has used these architectures, and they also, have developed their own loss function, what is called *contrastive loss*, this loss is an extension of the Hinge loss of the SVM. This loss longs for getting close similar pairs and moving away according to one defined margin, dissimilar pairs, altough, the binary cross entropy is used by the community. Also, the community has focused in the developing of the datasets, increase the size and use of different losses but there are not any landmark dataset.

There are several papers in the literature, one of the most famous is developed by Ahmed [**?**], they used In-network architecture although in order to join to the convolutional layers, they used *cross-input neighborhood differences* layer, this layer tried to increase the differences between the features of the inputs and obtain richer representation to the classification layer.

Another paper was published by Leal-Taixé [**?**], they are also the authors of the MOT challenge, their network used a cost function architecture besides they used as inputs the

two images and their optical flow. They used the network as part of a data association algorithm.

4 Software implementation

In this section, we will explain the implementation of the algorithm based on the theoretical 3 review.

4.1 System setup

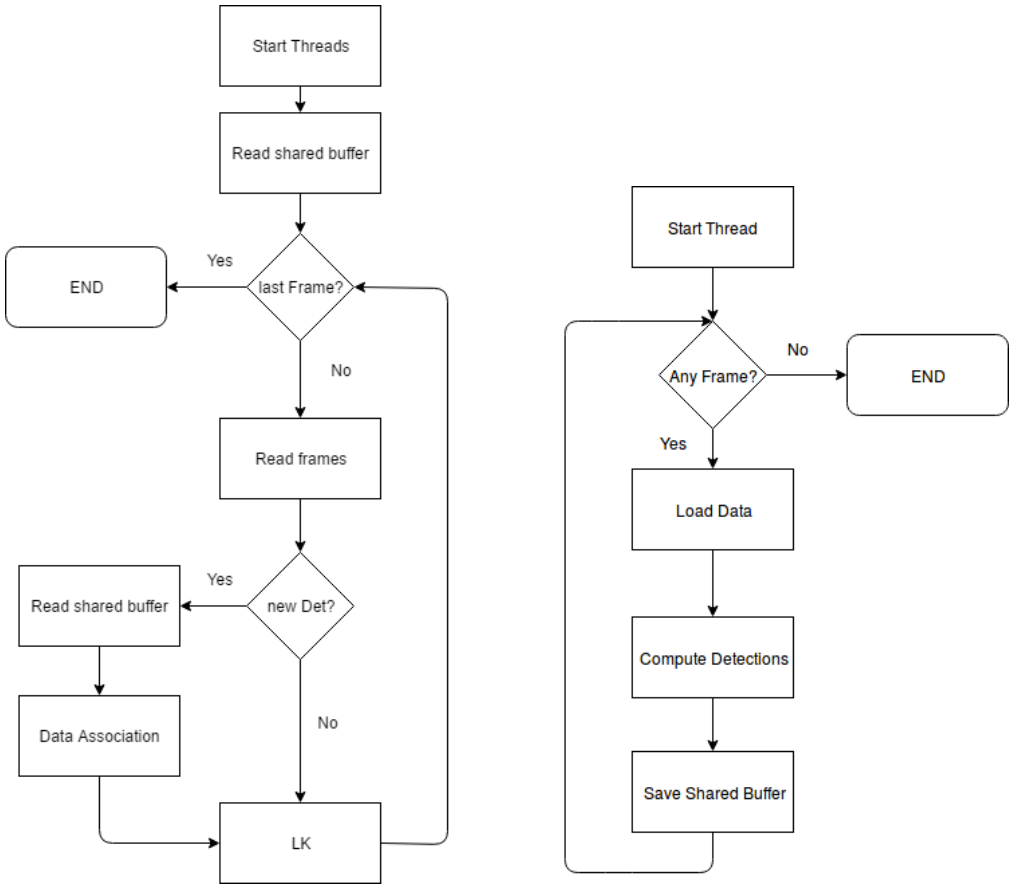
The developing and testing of the software has been carried out by a personal computer. This computer has a Intel Core i7 processor and 16 GB of RAM, whereas the operative system is Ubuntu 16.04 Xenial. The graphical processing unit consist in a AMD R7 M265, but its codecs are not available for our operative systems, therefore we can not take advantage of this hardware unit. For training our deep learning models we used the Amazon Web services platform, which instances is runned in operative system with a Nvidia Tesla K80 GPU.

4.2 System Overview

The system consists in two threads, one responsible of the execution of the object detector and the other one, the main thread, reasponsible of the tracker. The communication between them is unilateral, the object detector thread loads into a shared buffer the detections and the main thread incorporate them when he requires it.

So, the object detector thread, reads images, proces the forward pass of the neural network and save the detections in the shared buffer, it repeats this sequence until it has processed all the predefined list of images.

In another hand, the main tread, activates the object detector thread and waits till it gets the first detection, after this, it starts the tracking algorithm. It reads the images and computes the motion of all the regions of interest. However, at beginning of each cycle it comproves whether it has newer detection to mix in. This is the main operating mode of the algorithm and it summarizes in the figure 11. In the next sections we will develolpe in detail these parts.



(a) Main thread. (b) Object detector Thread.

Figure 11: Flow chart of the system.

4.3 Object detector

As we have stated previously, the responsible of getting the detection is the object detector thread. It only read images and computes the forward pass of the network and the results are like in the figure 12.

Figure 12: Detections of the algorithm.

For the choice of the detector we compare the detectors studied in the theoretical review on the *MOT16* dataset. In the figure 13 we can observe the ROC curve of different detectors.

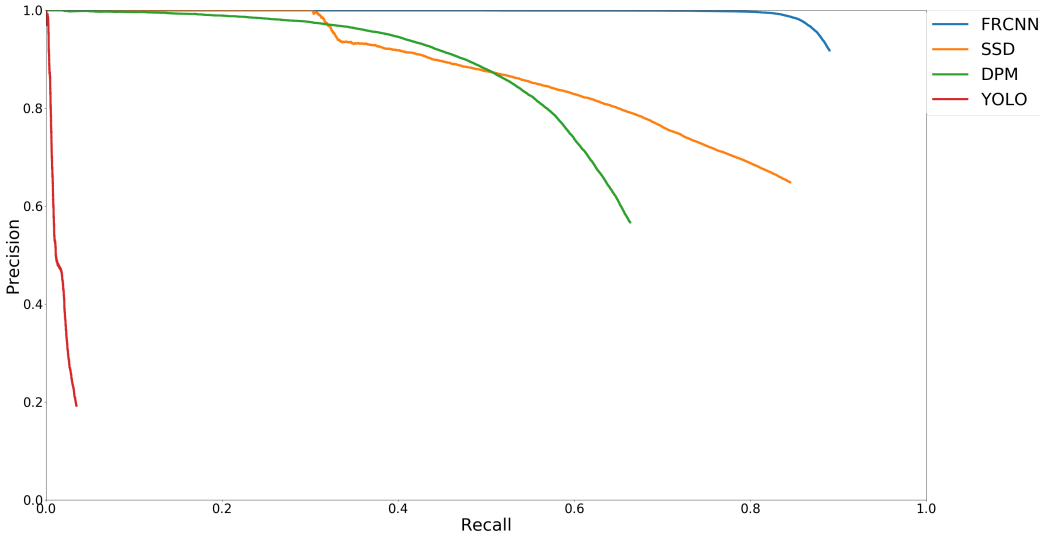


Figure 13: Detections ROC curve.

Finally, the conclusions of the characteristics are the following:

- **Faster-RCNN**, it scores 0.81 average precision on the dataset, although we could not used due a software incompabilities, it requires a Nvidia GPU to be executed.
- **R-FCN**, the code is not publicly available.
- **DPM**, it scores 0.69 average precision on the dataset, although we could not used due a software incompabilities, it requires a Nvidia GPU to be executed.
- **YOLO**, it scores 0.09 average precision on the dataset, it takes 6 seconds per image [24].
- **PVANET**, the code is not publicly available.
- **SSD**, it scores 0.73 average precision on the dataset, it takes 0.9 seconds per image. We used the tensorflow version of the detector [47] instead of the original code [69].

According to these results we chose the SSD detector, as object detector on this thesis.

4.4 Tracker

Once we have the detections, as we have stated previously, the responsible of compute the motion of the detections is this module. First, computing the features and then extract the motion from the movement of those features, we can observe this process in the figure 14.

(a) Features extraction. (b) Motion estimation.

Figure 14: Motion computation.

According to [57], our method belong to tracking using matching, considering that the algorithm performs a matching of the representation of the target model built from the previous frames. We do not build an additional extended model of the appearance and merge it with the matching, instead we will apply an object detector to correct the drift of the matching.

The tracking using matching is based on the optical flow, explained in 3, it computes the new position through gradient descent in several frames. We will assume that the motion is pure translational. As we can observe in the sequences of frames 17 of the dataset, the pedestrians move in translation way in the image plane, so this assumption is achieved.

Figure 15: Sequence of translational movement.

In contrast to the previous figure, we can observe the next figure 16 where the assumption of translation motion is not fulfilled (this sequences does not belong to the used dataset, only showed to contrast the previous idea) and a translational assumption will failed.

Figure 16: Sequence of no translational movement.

The tracking module is inspired by the well-known tracking algorithm *MedianFlow* by

Zalal et al[26] with his correspondent implementation in Python[25]. Next we explain the tracking module in details.

4.4.1 Preprocessing and feature extractor

For extracting the features, we use the OpenCV routine `goodFeaturesToTrack()`, this function determines strong corners on an image, according the Shi Tomasi method. His parameters are the following:

- `image`, input image
- `maxCorners`, maximum number of corners to return. If there are more corners than are found, the strongest of them is returned.
- `qualityLevel`, parameter characterizing the minimal accepted quality of image corners.
- `minDistance`, minimum possible Euclidean distance between the returned corners.
- `mask`, optional region of interest.
- `blockSize`, Size of an average block for computing a derivative covariation matrix over each pixel neighborhood.
- `useHarrisDetector`, Parameter indicating whether to use a Harris detector.
- `k`, Free parameter of the Harris detector.

The strength of the further processing depends on the quality and quantity of this features, so in other to improve both, we apply some prepreprocessing to the image, we tried several preprocessings like sharpening, image contrast, median filter, and equalization. We can observe the features extraction after this preprocessings in figure **??**.

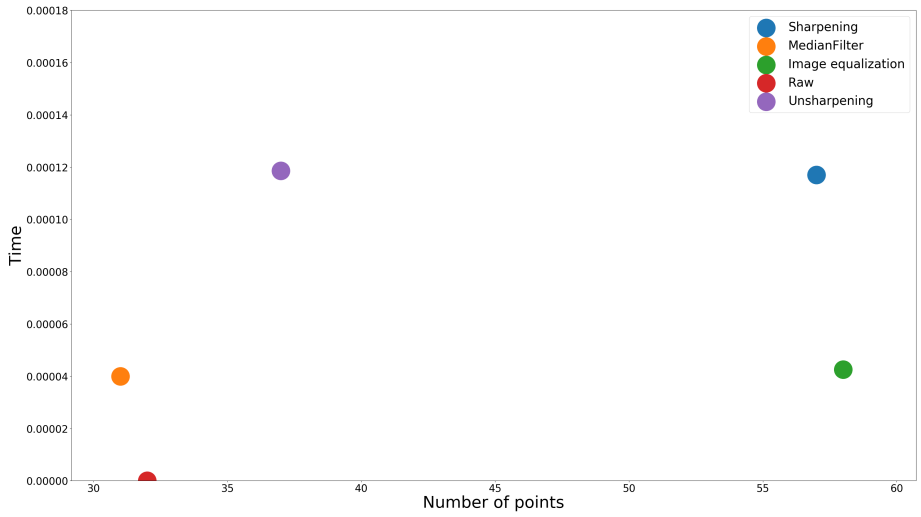


Figure 17: Preprocessing methods.

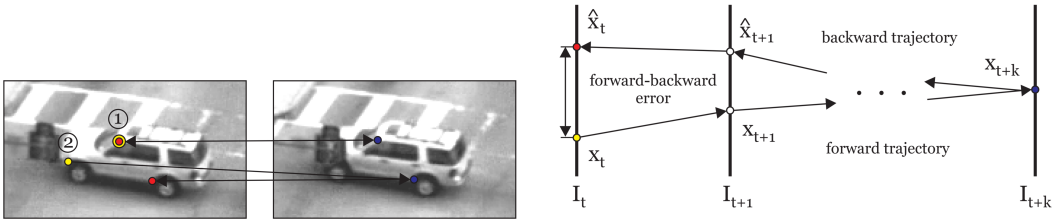
We realized that the best preprocessing is to equalize the image, in terms of speed, it only consists in equalize an histogram and transform the image, and in terms of quality and quantity, it increases over 55% the number points in comparision to apply it to the raw image. In the figure 18 we can observe the different number of features in the raw and in the equalized image.

(a) Raw image. (b) Equalaized image.

Figure 18: Different preprocessings.

4.4.2 Motion estimation

We used the lucas kanade method to matching the points. Also, we implement the same method used in [26], the proposed method is based on so called forward-backward consistency assumption that correct tracking should be independent of the direction of time-flow. Algorithmically, the assumption is exploited as follows. First, a tracker produces a trajectory by tracking the point *forward* in time. Second, the point location in the last frame initializes a validation trajectory. The validation trajectory is obtained by *backward* tracking from the last frame to the first one. Third, the two trajectories are compared and if they differ signicantly, the forward trajectory is considered as incorrect. 19 illustrates the method when tracking a point between two images. Point number 1 is visible in both images and the tracker is able to localize it correctly. Tracking this point forward or backward results in identilcal trajectories. On the other hand, point number 2 is not visible in the right image and the tracker localizes a different point. Tracking this point backward ends in a different location then the original one. We also implemented the forward method.



(a) Image forward-backward. (b) Scheme forward-backward.

Figure 19: Ilustration forward backward error.

To compute the matching we used the OpenCV routine `calcOpticalFlowPyrLK()`, this function implements a sparse iterative version of the Lucas-Kanade optical flow with pyramids. And his parameters are:

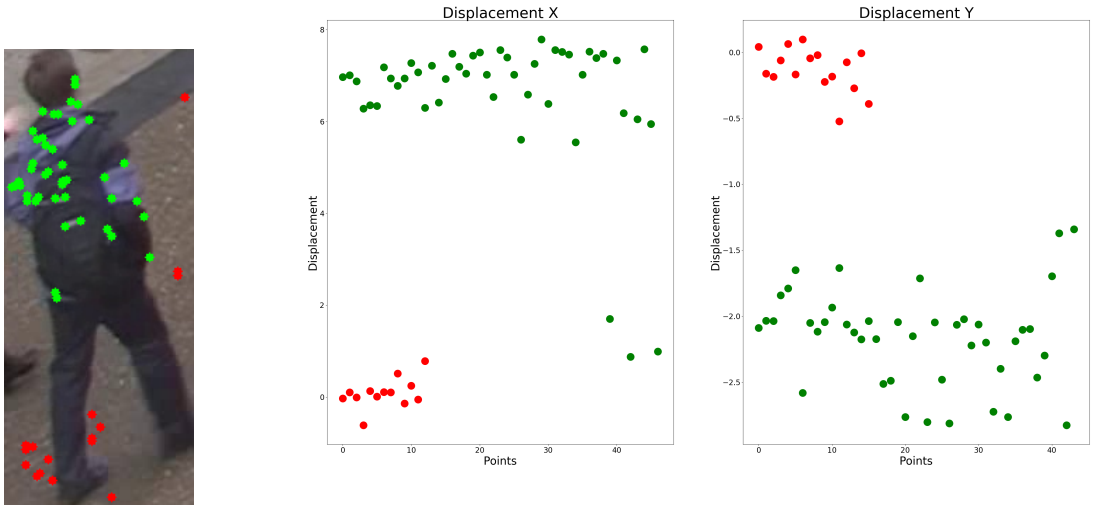
- `prevImg`, first image.
- `nextImg`, second image.
- `prevPts`, vector of 2D points for which the flow needs to be found.
- `nextPts`, output vector of 2D points containing the calculated new positions of input features in the second image.
- `status`, output status vector, it tells you whether the flow has been found.
- `err`, each element of the vector is set to an error for the corresponding feature.
- `winSize`, size of the search window at each pyramid level.
- `maxLevel`, number of pyramid levels.
- `criteria`, parameter, speifying the termination criteria of the iterative search algorithm.

In the next figure we can observe the matching of the feaetues in subsequents frames.

Figure 20: Motion estimation.

After this step we have a bunch of motion vectors, but some vectors in the bounding box do not belong to the person, and if we do not erase it will contribute to the motion

compuation. Usually this points belong to the static elements of the frames, like the floor or urban furniture, this points in terms of motion between subsequents frames will be very low or almost statics. We can observe this fact plotting the displacement of this points and drawing it in the image. So, we erase the points with a displacement smaller than a threshold.



(a) Image points.

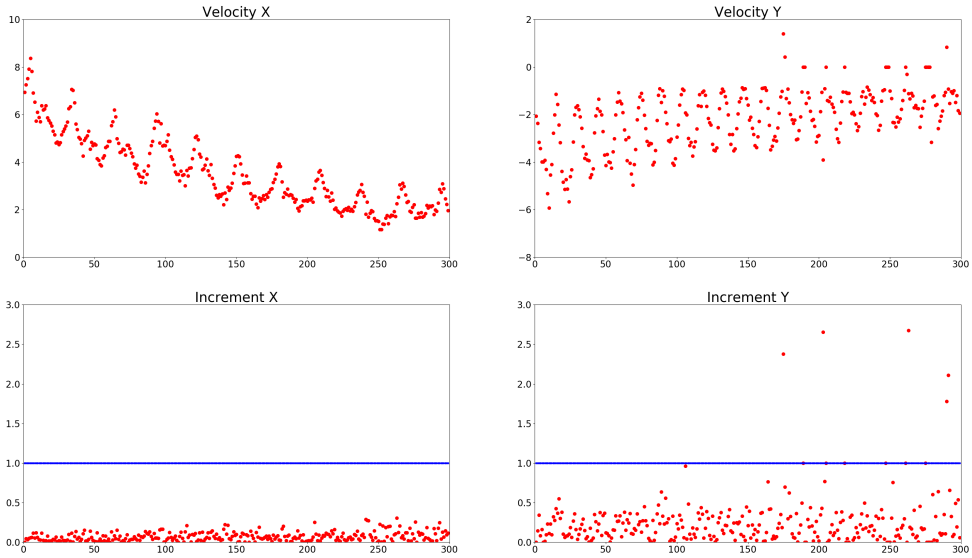
(b) Displacement graphics.

Figure 21: Different preprocessings.

Next we erased this points, we compute the motion as the median of all the contibutions displacement vectors. The scale change is computed as follows: for each pair of points, a ratio between current point distance and previous point distance is computed, bounding box scale change is defined as the median over these ratios.

At this point, we have an implementation of the tracking algorithm given a set of bounding boxes. Nevertheless, we notice running our algorithm on the dataset, that the bounding box could compute the motion of the assigned pedestrian including the points belonging to another pedestrian who appears also in the bounding box and this interaction will cause a wrong estimation of the movement of the target and eventually the pedestrian will not embedded by the bounding box. So, we need a mechanism to detect this fails, therefore we studied how the motion algorithm behaves in this situations. When it has got a trajectory without crossing with other pedestrian, the vertical and horizontal displacement behave like a damping sine wave function (or increasing sine wave) and the increment respect the previous displacment is small. We can observe this process in figure 22.

(a) Trajectory.

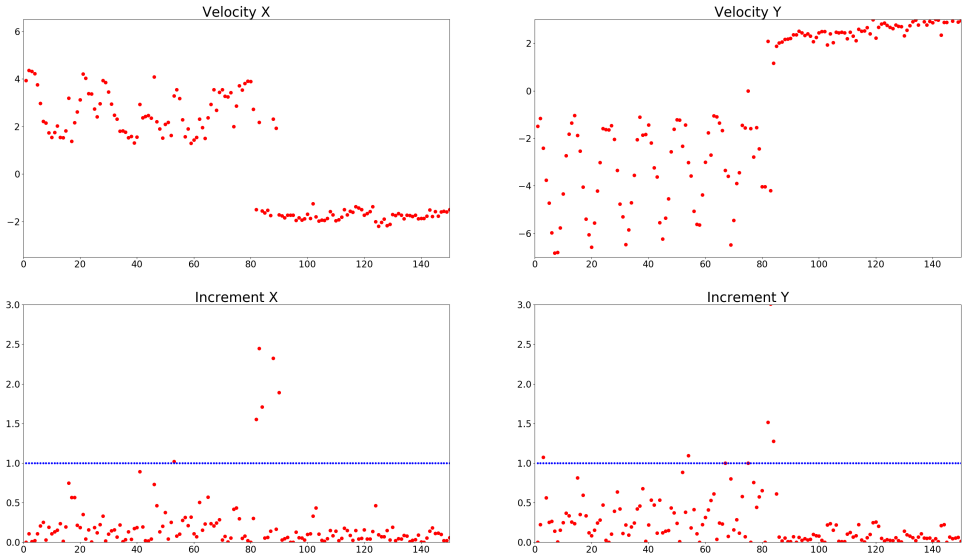


(b) Plots movement.

Figure 22: Regular trajectory.

In contrast, when it crosses with another pedestrian, the displacement gets disrupt, then the normalized different with the previous displacement gets a high value, we can observe this process in figure 23. We set a threshold to notice this interference and delete this bounding box. We delete them from the current tracking execution, but we save the bounding box appearance for following processings.

(a) Trajectory.



(b) Plots movement.

Figure 23: Wrong trajectory.

4.5 Data association

Once we computed the trajectories, in the next iteration we might have to add a detection, so we need a module to combine these trajectories with detections. So, for each pedestrian we distinguish three situations:

- **Situation 1**, the tracket has got a nearby detection, then the detection replace the tracket bounding box. This is what is called spatio-temporal constraint.
- **Situation 2**, the tracket has not got a nearby detection, then the bounding box tracket continues.
- **Situation 3**, the pedestrian has not got a tracket but has a detection. In this case we need to classify this detection as a new detection, is the first time that the first time that the pedestrian showed in the frame or whether is a tracket lost and we need to check its identity.

We can observe this procedure in the figure 24, in green we can observe the detections and in blue the trackets. We defined nearby as the distance between the centres of the boundings boxes, this distance has to be lower than a threshold to be consider nearby.

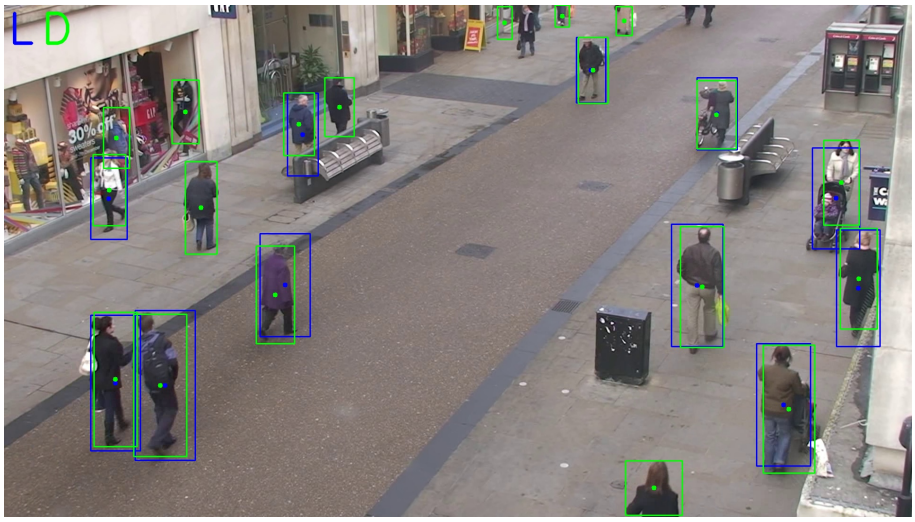


Figure 24: Spatio-temporal data association.

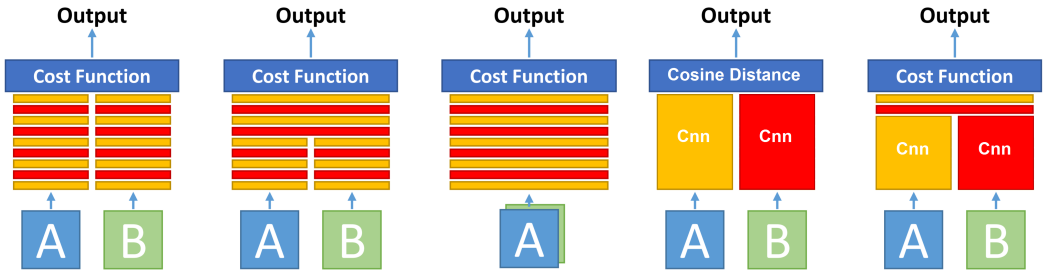
As we stated, in situation three, we might delete some tracking bounding box, like we explained in section 4.4.2 and get a new detection that maybe it belongs to the same identity, we need a mechanism to preserve the identity of the pedestrian. In the early version of this algorithm we consider those detections as new, and we assign it a new identity.

4.5.1 Siamese Network

To mantain the identity of the pedestrian we need a mehtod to compare miss trackets with no associated detections. So, we decided to solve it with deep learning techniques, the models that we considered are the following:

- **Siamese network: Joint data input**, according to the literature this architecture gives the best results compared with the other topologies. The input of the network is aa concatenation of the two images and the network proces together.
- **Siamese network: Cost function**, this is based on the idea of deep learning as feature extractor and top layers as classifier. Two branches that share parameters process the images and classify it.
- **Siamese network: In-network**, this is a mix of the previous models, where the information of the convolutional layers merges at some point before the classifier.
- **Feature extractor with cosine distance**, we used well-known architectures for image classification to extract features from the images and then compare those features with the cosine distance.
- **Famous network fine-tuned**, we extract features for each image with a well-known architecture and merge it with a fully connected layer.

We can observe this architectures in the figure 25.

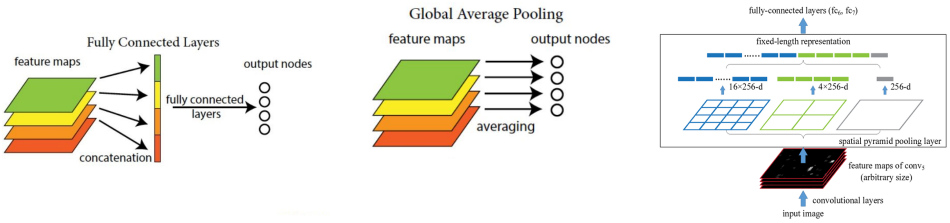


(a) Cost function. (b) In-network. (c) Input stacking. (d) Features with cosine distance. (e) Cnn fine-tuned.

Figure 25: Siamese CNN topologies.

The main characteristics of the trained networks are the following:

- **Loss**, we used the binary cross entropy as a loss, we tried with the contrastive divergence but it did not converge.
- **Optimizer**, As optimizer we used Adam, even though it has a mechanism to decrease the learning rate, we add a exponential decay, it speed up the convergence
- **Activation**, we used ReLu. Currently, there other activations functions, but ReLu has been stablished as the reference.
- **Inizialization**, To initialize the wrights we used He. inizialization, as activation function we used ReLu, in adition we inizializate the biases with the value of 0.1, in this way we avoid the dead neurons in the firsts iterations.
- **Batch normalization**, We make use of batch normalization to normailize the ouput of each layer.
- **Regularization**, we use Dropout in the fully connected layer to avoid overfitting.
- **Final layers**, In the junction between the convolutional layers and the fully connected historically, a flatten mechansihm of the tensor has been used, but it increases dramatically the number of neurons in the fully connected layer, and it shows problems to converge, from the publication of InceptionV3 it appears with a Global Average pooling, it computes the spatial average of each layer of the tensor, reducing the number of parameters. Also we used the Spatial pyramid pooling layers



(a) Flatten. (b) Global average pooling. (c) Spatial pyramid pooling.

Figure 26: Final layers.

- **Output**, We did not use softmax as ouput, we only used one neuron with sigmoid activation, in this way the output is constrainet between 0 and 1.

We developed our models in a VGG way, stacking several convolutional layers and finishing with a fully connected layers. We started with a few convolutional layers and added more till we reach a satisfaying performance.

For the dataset, it does not exist a prominent dataset in the field, so we decided to use the MOT16 ground truth as dataset to adapt the domain. In order to do so, we extract the detections with their identities, and then for each identity we selected all possible random pairs and for the negative set we selected two random identites. The negative dataset is much bigger than the positive dataset, so we limited it to have a balanced dataset. The problem with the MOT16 dataset, is that the ground truth was built with the detections of a classifier and there is not a human intervention, resulting a messy ground truth. We inspected the dataset and around the 70% of the dataset was wrong, there are a lot of oclussion in detection resulting in erronous pairs.

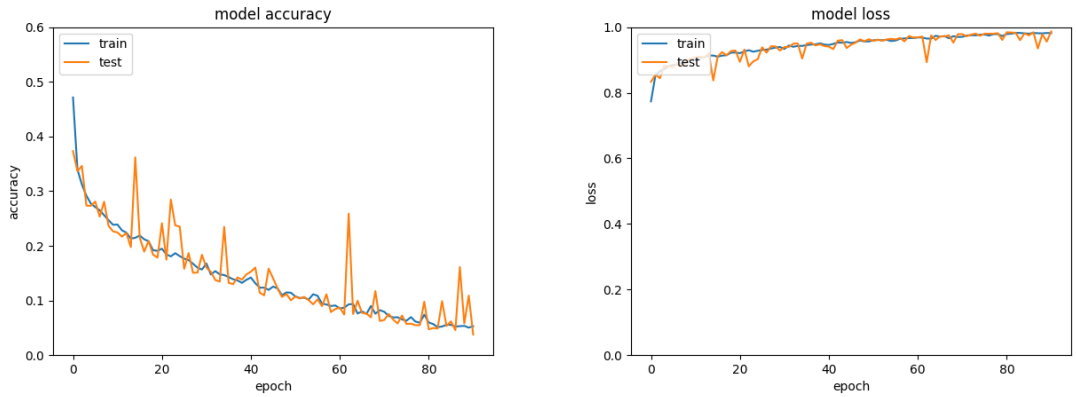
Then, we decided to discard the MOT16 dataset an use the TownCenter dataset [64] from the University of Oxford. We have got 29824 positive and negative pairs, then a dataset of 59648 image pairs. We split the dataset between training and validation set, 80% and 20% respectively. For testing we selected a set of identities of the MOT16 dataset. To regularize and enlarge our dataset we added some data augmentation to our datase like we observe in figure 27. For each pair we added one transformation, so we double our dataset. We tried to apply all the transformation for each images but the dataset was too noisy and the network did not converge.

(a) Original image. (b) Random brightness. (c) Random image crop. (d) Vertical flip. (e) Gaussian blur. (f) Random shadow.

(g) Zoom in. (h) Rotation and translation. (i) Zoom out. (j) Gaussian noise. (k) Opposite vignetting.

Figure 27: Data augmentation.

We trained all the models and obtained a graphs like the figure 28, we observe that the network converge, it decreases the loss and increases the accuracy, also we observe that tests plots are a little bit noisy, but we considered that we applied enough regularization techniques.

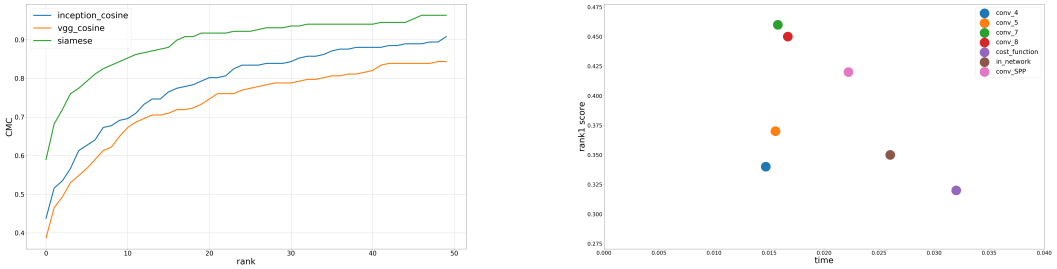


(a) Losses. (b) Accuracy.

Figure 28: Results training.

Finally the comparision using the CMC in the figure 29, we can observe that the siamese with less layers than bigger models like VGG or Incpetion perform better, this remarks

the idea of training jointly the feature extractor and the classifier and the need of task specific networks.



(a) CMC plot.

(b) Performance-timing comparision.

Figure 29: Comparision of the systems.

5 Datasets and evaluation procedures

Next we explain the datasets and measures utilized in this thesis.

5.1 Datasets for object detection

This section describes the most common datasets used in object detection tasks. Throughout the history of computer vision research datasets have played a critical role. They not only provide a means to train and evaluate algorithms, they drive research in new and more challenging directions. In order to accomplish this, they provide:

- a collection of challenging images and high quality annotation.
- an standard evaluation methodology, so the performance of the algorithms can be compared.

In the next subsections, we will explain the main datasets for object detection task.

5.1.1 Pascal Visual Objects Classes

The Pascal Visual Object Classes (VOC) challenge [9] is a benchmark in visual object category recognition and detection. Organised annually from 2005 to 2012, the challenge and its associated dataset has become accepted as one of the most benchmark for object detection. All the images are from the flickr consumer photographs website and annotated with the Mechanical turk tool.

The most popular editions of the challenge for object detection are those from years 2007 and 2012.

5.1.2 VOC07

The challenge of the year 2007, it contains 10 thousand images in the trainval and test sets, with almost 12 thousand objects. This was one the first datasets for object detection before the era deep learning. Also, it is very useful for researchers, due it has 2.5 mean object per image and it is very challenging. In the figure 30 we can observe the distribution of images and objects instances.

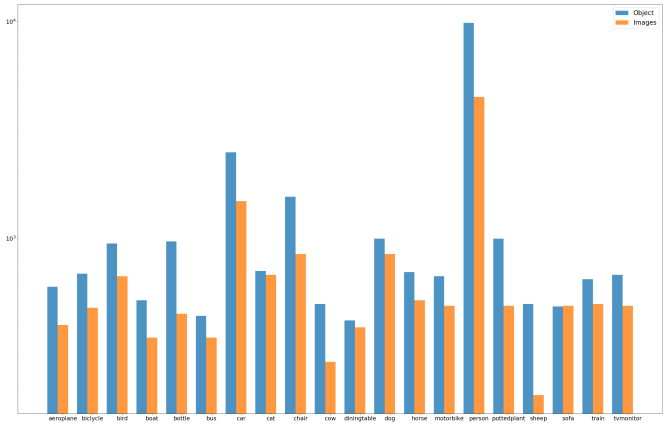


Figure 30: Distribution of VOC07 dataset.

5.1.3 VOC12

The 2012's edition is also, one of the most used dataset in object detection tasks. It increases the volume of images of the 2007 edition up to 10 thousand images on trainval and test set and similar quantity of instances per image.

5.1.4 ImageNet

ImageNet project [8] with the challenge ImageNet Large Scale Visual Recognition Challenge [ILSVRC] was the first large-scale database, temporally developed to supply the deep learning techniques, eager of feed with tons of images. ImageNet aims to populate the majority of the 80000 synsets of WordNet with an average of 500-1000 clean and full resolution images. The collection was based on the query of that words on several image search engines and human refined on the Amazon Mechanical Turk platform.

In 2016, the project collects more than 10 million of annotated images with 1000 classes. Although its main purpose is image classification, it has an object detection challenge with 200 categories with over a 1 million images with objects annotated.

5.1.5 COCO

The Microsoft Common Objects in Context also known as COCO dataset [38], is a dataset that address the three core research problems in scene understanding:

- detecting non-iconic views of objects, for many dataset most of the objects have an iconic representation, they appear unobstructed , near the center of the photo and with their canonical shape. So in this dataset, they included images to struggle the object recognition task, like objects in the background, partially occluded, amid clutter. Therefore, reflecting the composition of actual everyday scenes.

- contextual reasoning between objects, nowadays natural images contain multiple objects, and their identity can only be solved using context, due to small size or ambiguos appearance in the image, so in this dataset, images contain scenes rather isolated objects.

- the precise 2D localization of objects, also the detailed spatial understanding of object layout will be a core component of an image understanding system, so this dataset struggle to do so.

So, the three main tasks of this challenge are object classification, object detection and semantic scene labelling. This dataset contains 91 object categories, with 2.5 million labelled object instances in 328 thousand images, labeled with the Amazon Mechanical Turk tool.

5.1.6 Dataset comparison

The datasets from Pascal challenge are very useful to test object detection algorithms, their quantity is very handy (a few thousands of images) and contains a challenging quantity of objects per image, very interesting for the algorithms. But its little amount of images does not permit to train a network on this dataset, although it can be used to finetune the network.

The datasets for the ImageNet challenge are not used to much in object detection tasks, it contains a few instances per image, this not encourage researcher to use it. Although, it is very utilize to extract features and then finetune your net.

The COCO datasets, is the most recent one, is the one focus on object recognition, and the detection suppose a challenge due the objects are in common places and are very challenging to detect. And it is very interesting to due of the quantity of instances per image.

The COCO challenge contains 91 object categories with 82 of them having more than 5 thousand labeled instances. In total the dataset has 2.5 million labeled instances in 328 thousand images. In contrast to ImageNet dataset, COCO has fewer categories but more instances per category. Also, it has more instances per category than the VOC dataset. We can observe that difference in the chart 31. This fact aid in learning detailed object models capable to chance the variability and also its 2D location.

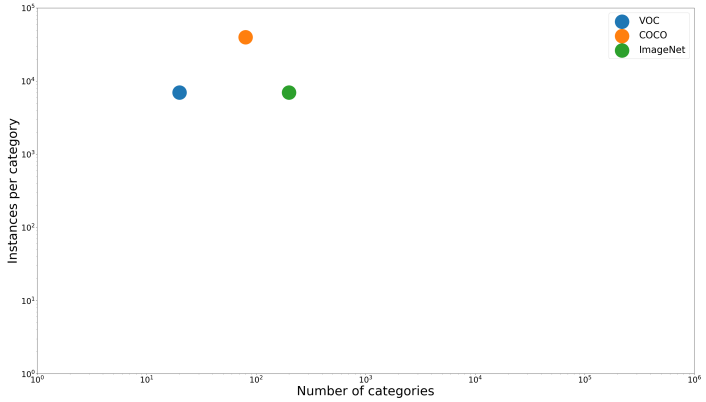


Figure 31: Distribution of pascal.

In addition, another prominent feature of the COCO over the other two, is the number of labelled instances per image which may aid in learning contextual information. This difference can be seen in **??** chart.

Moreover, the COCO dataset uses images from non-canonical point view, allowing to the algorithm to be robust to everyday views. This feature can be observed in the plot 32, in this plot we can observe differences views of the same category. And clearly the coco's images is the most uni-conic representation.

(a) Pascal VOC. (b) ImageNet. (c) COCO.

Figure 32: Distribution of pascal.

Finally, the table 2 summarizes the main statistics of the dataset stated previously.

	VOC07	**VOC12**	**ImageNet [2014]**	**Coco [2015]**
trainval set	5011	11540	476688	165482
test set	4952	10991	40152	81434
Number of classes	20	20	200	80
Mean obj per image	2.5	2.4	1.1	7.2
Number person instances	4690	8566	-	300000

Table 2: Datasets tables

5.1.7 Evaluation of object detection algorithms

In order to compare the performance of the different datasets, each challenges establish a clear measure. In this thesis, we used the interpolated average precision (AP), used in the Pascal VOC challenge (based on [54]).

For each class, the precision/recall curve is computed from a method's ranked output.

- Recall, is defined as the proportion of all positives examples ranked above a given rank.
- Precision is the proportion of all examples above the rank which are from the positive class.

The AP summarises the shape of the precision/recall curve, and is defined as the mean precision at a set of eleven equally spaced recall levels [0,0.1,...,1]:

$$AP = \frac{1}{11} \sum_{r \epsilon (0,...,1)} p_{interp}(r)$$

The precision at each recall level r is *interpolated* by taking the maximum precision measured for a method for which the corresponding recall exceeds r:

$$p_{interp}(r) = max_{\hat{r}:\hat{r}>r} p(\hat{r})$$

The authors justified this measurement as a way to reduce the impact of the 'wiggles' in the precision/recall curve, caused by small variations in the ranking of examples. In the figure 33, we can observe this effect on the curve.

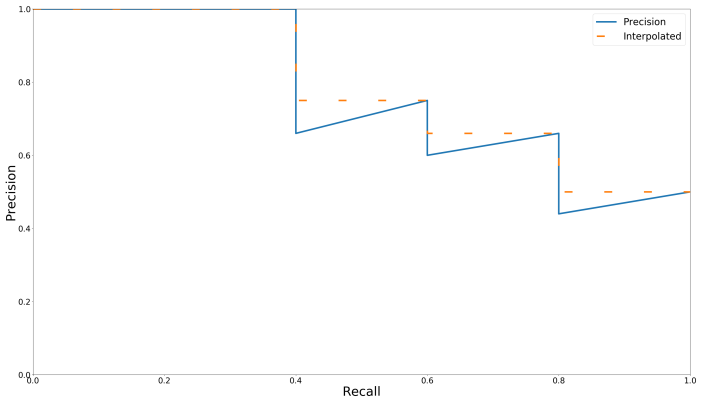


Figure 33: Comparision interpolated and normal curve.

In addition, detections were assigned to ground truth objects and judged to be true/false positives by measuring bounding box overlap. To be considered a correct detection, the area of overlap a_0 between the predicted bounding box B_p and ground truth bounding box B_{gt} must exceed 0.5 by the formula:

$$a_0 = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})}$$

where $B_p \cap B_{gt}$ denotes the intersection of the predicted and ground truth bounding boxes and $B_p \cup B_{gt}$ their union. The treshold of 50 % was set deliberately low to account for inaccuracies in bounding boxes in the ground truth data. Multiple detections of the same object in an image were considered false detections.

Finally, we want to point out, even we don't take into account in our implementation, setting the threshold IoU to a value of 0.5 could cause misdetections of small objects [8], they propose an adaptive setting of that threshold based on the size of the ground truth and so detect correctly small objects. In practice this change only affects 5.5% of objects in the detection validation set.

5.2 Datasets for multiple object tracking

Evaluating and comparing multi-target tracking methods is not trivial for numerous resasons.

- First, the perfect solution one aims to achieve is difficult to define clearly. Partially visible, occluded, or cropped targets, reflections, and objects that are very closely resemble targets; all impose intrinsic ambiguities, such that even humans may not agree on one particular ideal solution.

- Second, a number of different evaluation metrics with free parameters and ambiguous definitions often lead to inconsistent quantitative results across the literature.

- Finally, the lack of pre-defined test and training data makes it difficult to compare different methods fairly.

In contrast to other research areas in computer vision, still lacks large-scale benchmarks.

5.2.1 PETS

Targeted primarily at surveillance applications [10]. The 2009 version consisted of 3 subsets, S1 targeted at person count and density estimation, S2 targeted at people tracking, and S3 targeted at flow analysis and event recognition. In the 35 we can observe one image from this dataset.

Figure 34: Example of Pets.

Even for this widely used benchmark, we observe that tracking results are commonly obtained in an inconsistent fashion: involving using different subsets of available data, different detection inputs, inconsistent model training that is often prone to over-fitting, and varying evaluation scripts. Results are thus not easily comparable [32].

5.2.2 MOT challenge

The aim of the Multiple object tracking [MOT] is to standardize the use of multiple people trackings datasets, in order to do so, they solve the problems in this kind of dataset, explained above.

5.2.3 Evaluation of multiple people tracking algorithms

A critical point with any dataset is how to measure the performance of the algorithms. A large number of metrics for quantitative evaluation of multiple target tracking have been proposed. Choosing unique general evaluation is still ongoing.

On the one hand, it is desirable to summarize the performance into one single number to enable a direct comparison. On the other hand, one might not want to lose information about the individual errors made by the algorithms and provide several performance estimates, which precludes a clear ranking.

We will explain two sets of measures that have established themselves in the literature the CLEAR metrics [58], and a set of track quality measures [72].

As in the object detection metrics, we can classify each tracket, whether it is a true positive, that describes an actual (annotated) target, whether the output is a false alarm (or false positive, FP). This decisions is typically made by the well-known thresholding measure of Intersection of the union [IoU]. Also a target that is mised by any tracker is a false negative.

Due to we are working with multiple object, we assume that each ground truth trajectory has one unique start and one unique end point, that is not fragmented. So we need to penalty re-identification. This is called, identity switch [IDSW], and it is counted as if a

ground truth target i is matched to track j and the last known assignment was $k = j$. The next figure summarizes the stated measures (the grey area indicate the the matching threshold).

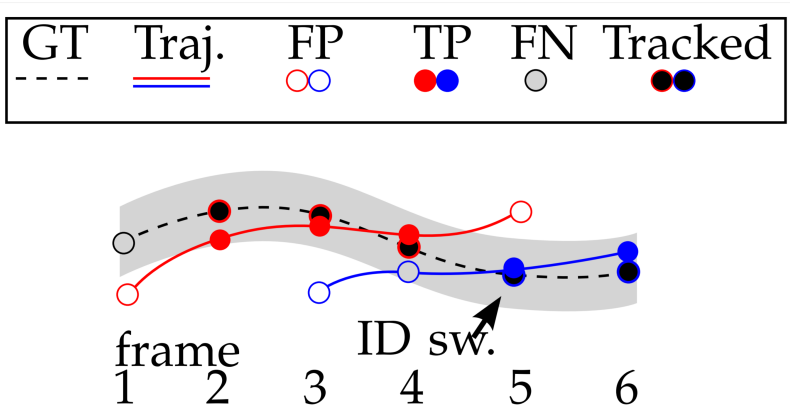


Figure 35: Example of measures.

Then after determining true matches and establishing the correspondandes it is now possible to compute the metrics over all the sequences.

The multiple object tracking accuracy [MOTA] [58] is perhaps the most widely used figure to evaluate a tracker's perfomance. The main reason for this is its expressiveness as it combines as it combines three sources of errors defined above:

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t}$$

where t is the frame index and GT is the number of ground truth objects. This measures gives an indication of the overall performance.

The multiple object tracking precision [MOTP] is the average dissimilarity between all true postives and their corresponding ground truth targets. For bounding box overlap, that is computed as

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t}$$

where c_t denotes the number of matches in frame t and $d_{t,i}$ is the bounding box overlap of target i with its assigned ground truth object. Thereby gives the average overlap between all correctly matched hypotheses. So, the MOTP is a measure of localization precision.

As we have stated above, another metrics are the track quality measures. Each ground truth trajectory can be classified as mostly tracked (MT), partially tracked (PT), and mostly lost (ML). This is done based on how much of the trajectory is recovered by the tracking algorithm. A target is mostly tracked if it is successfully tracked for at least 80% of its life span, without consider if there are an identity switch. If a track is only recovered for less than 20% of its total length, it is said to be mostly lost (ML). All other tracks

are partially tracked. Finally antoher quality measure is track fragmentations (FM), it counts how many times a ground truth trajectory is resumed at a later point.

5.3 Datasets for pedestrian identification

A number of datasets for image-based re-Identification have been released, and some commonly used datasets are summarized in table 3.

Name	**Date**	**Images**	**IDs**	**Cameras**	**Label**	**Evaluation**
VIPeR [62]	2007	1264	632	2	hand	CMC
iLIDS [68]	2009	476	119	2	hand	CMC
GRID [63]	2009	1275	250	8	hand	CMC
CAVIAR [15]	2011	610	72	2	hand	CMC
PRID2011 [22]	2011	1134	200	2	hand	CMC
WARD [7]	2012	4786	70	3	hand	CMC
CUHK01 [36]	2012	3884	971	2	hand	CMC
CUHK02 [34]	2013	7264	1816	10	hand	CMC
CUHK03 [35]	2014	13164	1467	2	hand/DPM	CMC
RAiD [42]	2014	1264	43	4	hand	CMC
PRiD 450S [51]	2014	900	450	2	hand	CMC
Market-1501 [75]	2015	32668	1501	6	hand/DPM	CMC/mAP

Table 3: Statistical comparision datasets.

Over recent years, progress can observed. The dataset scale is increasing. Many of these datasets are relatively small in size, especially those of early days, but recent datasets, such as CUHK03 and Market-1501, are larger. Both have over 1000 ID's and over 10000 bounding boxes, and both datasets provide good amount of data for training deep learning models. Second, the bounding boxes tend to be produced by pedestrian detectors, instead of being hand-drawn. Also, more cameras are used during collection, this helps to increase generalization. There are not a prominent dataset in the literature.

5.3.1 Evaluation for pedestrian identification

When evaluating identification algorithms, the cumulative matching characteristics (CMC) curve is usually used. CMC represents the probability that a query identity appears in differentsized candidate lists.

Formally [17], for each probe p from P_G we sort the similarity scores against gallery G, and obtain the rank of the match. Identification performance is then stated as the fraction of probes whose gallery match it at rank r or lower. If the set of probes with a close match is:

$$C(r) = \left\{p_j : rank(p_j) \leq r\right\} \quad \forall p_j \in P_G$$

where the rank is defined as before. We now define the Cumulative match characteristic (CMC) to be the identification rate as a function of r:

$$P_I(r) = \frac{|C(r)|}{|P_G|}$$

which we plot as the primary measure of identification performance. It gives an estimate of the rate at which probe images will be classified at rank r or better. One drawback of the characteristics is its dependence on gallery size, $|G|$.

6 Experiments

In this section, we will analyse the performance and the timings of our solution.

6.1 Performance

We used the measurents of the MOT challenge, the most import measurement is the parameter MOTA, it is a global measurment because it condenses false positive, false negative and identity switching. We can observe in the table **??**, comparing the previous version of the algoirithm and the lastes versiont that we increased that parameter.

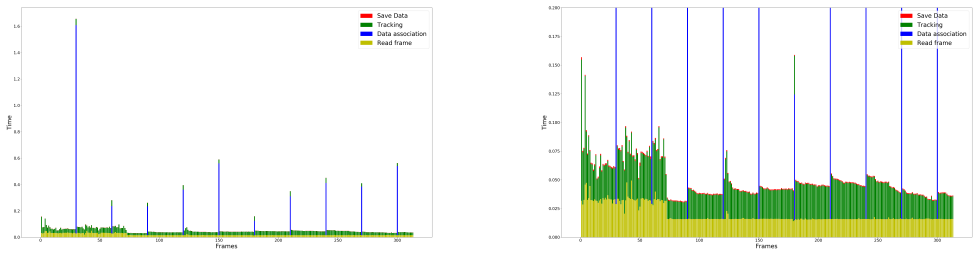
	Rcll	**Prcn**	**FAR**	**GT**	**MT**	**PT**	**ML**	**FP**	**FN**	**IDs**	**FM**	**MOTA**	**MOTP**	**MOTAL**
v1	24.4	49.5	5.17	517	12	180	325	27502	83462	827	1053	-1.3	69.3	-0,5
v2	19.6	61.8	2.52	517	3	127	387	13373	88754	618	936	10.8	70.3	7.5

Table 4: Results algorithm.

	Rcll	**Prcn**	**FAR**	**GT**	**MT**	**PT**	**ML**	**FP**	**FN**	**IDs**	**FM**	**MOTA**	**MOTP**	**MOTAL**	**FPS**
02	12.9	51.4	3.63	54	0	13	41	2181	15526	113	146	0.1	67.1	0.7	9.02
04	28.5	71.2	5.23	83	0	41	42	5495	33980	185	290	16.6	71.1	17.0	12.3
05	30.9	42.4	3.41	125	3	43	79	28571	4713	83	109	-12.2	67.8	-11.1	17.94
09	38.7	68.6	1.78	25	1	19	5	932	3225	62	71	19.7	71.2	20.9	10.52
10	5.4	62.4	0.62	54	0	4	50	404	11647	81	133	1.5	68.4	2.1	14.23
11	19.7	65.6	1.05	69	0	16	53	948	7366	72	121	8.6	71.4	9.4	17.49
13	6.2	34.9	1.75	107	0	9	98	1315	10743	32	59	-5.6	67.1	-5.4	20.5
Global	19.6	61.8	2.52	517	3	127	387	13373	88754	618	936	10.8	70.3	7.5	15.85

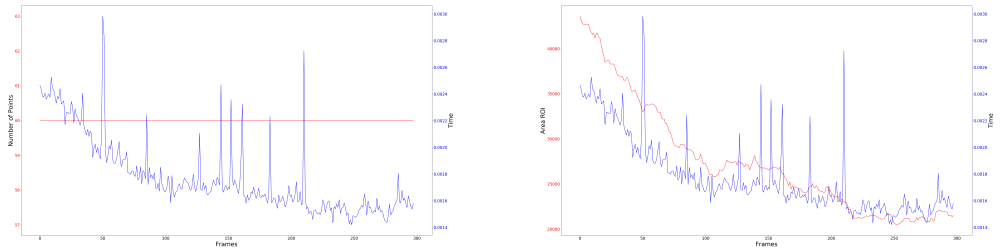
Table 5: Results algorithm.

6.2 Timing



(a) High texture person. (b) Low texture person.

Figure 36: Differences texture examples.



(a) High texture person. (b) Low texture person.

Figure 37: Differences texture examples.

(a) High texture person.

(b) Low texture person.

Figure 38: Differences texture examples.

7 Conclusions

7.1 Future work

- Port to C++. Port the developed software to high speed software.
- GPU implementation. Use Gpu advances, to get a parallel implementation of the code.
- Add propbabilistic framework. Include bayesian filter techniques to increase perfomance.
- Study new siamese architectures, like inception stem.

References

[1] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147 – 169, 1985.

[2] Sean Bell and Kavita Bala. Learning visual similarity for product design with convolutional neural networks. *ACM Trans. on Graphics (SIGGRAPH)*, 34(4), 2015.

[3] Jane Bromley, Isabelle Guyon, Yann Lecun, Eduard Säckinger, and Roopak Shah. Signature verification using a "siamese" time delay neural network. In *In NIPS Proc*, 1994.

[4] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, June 2005.

[5] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-FCN: object detection via region-based fully convolutional networks. *CoRR*, abs/1605.06409, 2016.

[6] Martin Danelljan, Gustav Hager, Fahad Shahbaz Khan, and Michael Felsberg. Learning spatially regularized correlation filters for visual tracking. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[7] Abir Das, Anirban Chakraborty, and Amit K. Roy-Chowdhury. *Consistent Re-identification in a Camera Network*, pages 330–345. Springer International Publishing, Cham, 2014.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[9] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[10] J. Ferryman and A. Ellis. Pets2010: Dataset and challenge. In *Proceedings of the 2010 7th IEEE International Conference on Advanced Video and Signal Based Surveil-*

lance, AVSS '10, pages 143–150, Washington, DC, USA, 2010. IEEE Computer Society.

[11] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. Flownet: Learning optical flow with convolutional networks. *CoRR*, abs/1504.06852, 2015.

[12] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119 – 130, 1988.

[13] Ross Girshick. Fast r-cnn. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[15] Doug Gray, Shane Brennan, and Hai Tao. Evaluating appearance models for recognition, reacquisition, and tracking. In *In IEEE International Workshop on Performance Evaluation for Tracking and Surveillance, Rio de Janeiro*, 2007.

[16] Douglas Gray and Hai Tao. *Viewpoint Invariant Pedestrian Recognition with an Ensemble of Localized Features*, pages 262–275. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[17] Patrick Grother, Ross J. Micheals, and P. Jonathon Phillips. *Face Recognition Vendor Test 2002 Performance Metrics*, pages 937–945. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[18] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, pages 10–5244. Citeseer, 1988.

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[20] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 FPS with deep regression networks. *CoRR*, abs/1604.01802, 2016.

[21] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Comput.*, 18(7):1527–1554, July 2006.

[22] Martin Hirzer, Csaba Beleznai, Peter M. Roth, and Horst Bischof. Person re-identification by descriptive and discriminative classification. In *Proc. Scandinavian Conference on Image Analysis (SCIA)*, 2011.

[23] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, and Kevin Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. *CoRR*, abs/1611.10012, 2016.

[24] J. Redmon. You only look once. `https://github.com/pjreddie/darknet` [2017-06-11], 2017.

[25] Jay Rambhia. Median flow tracker in python. `https://github.com/jayrambhia/MFTracker` [2015-08-15], 2015.

[26] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *Pattern recognition (ICPR), 2010 20th international conference on*, pages 2756–2759. IEEE, 2010.

[27] Kye-Hyeon Kim, Yeongjae Cheon, Sanghoon Hong, Byung-Seok Roh, and Minje Park. PVANET: deep but lightweight neural networks for real-time object detection. *CoRR*, abs/1608.08021, 2016.

[28] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. 2015.

[29] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012.

[30] M. Köstinger, M. Hirzer, P. Wohlhart, P. M. Roth, and H. Bischof. Large scale metric learning from equivalence constraints. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2288–2295, June 2012.

[31] Laura Leal-Taixé, Cristian Canton-Ferrer, and Konrad Schindler. Learning by tracking: Siamese CNN for robust target association. *CoRR*, abs/1604.07866, 2016.

[32] Laura Leal-Taixé, Anton Milan, Ian D. Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *CoRR*, abs/1504.01942, 2015.

[33] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1(4):541–551, Dec 1989.

[34] W. Li and X. Wang. Locally aligned feature transforms across views. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3594–3601, June 2013.

[35] W. Li, R. Zhao, T. Xiao, and X. Wang. Deepreid: Deep filter pairing neural network for person re-identification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 152–159, June 2014.

[36] Wei Li, Rui Zhao, and Xiaogang Wang. *Human Reidentification with Transferred Metric Learning*, pages 31–44. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[37] Shengcai Liao, Yang Hu, and Stan Z. Li. Joint dimension reduction and metric learning for person re-identification. *CoRR*, abs/1406.4216, 2014.

[38] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[39] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015.

[40] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference*

on Artificial Intelligence - Volume 2, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[41] M. Chatterjee, Y. Luo. Similarity learning with cnn. `http://slazebni.cs.illinois.edu/spring17/lec09_similarity.pdf` [2016-12-15], 2016.

[42] N. Martinel and C. Micheloni. Re-identify people in wide area camera network. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 31–36, June 2012.

[43] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[44] Alexis Mignon. Pcca: A new approach for distance learning from sparse pairwise constraints. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 2666–2672, Washington, DC, USA, 2012. IEEE Computer Society.

[45] Marvin Minsky and Seymour Papert. *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA, 1969.

[46] Hyeonseob Nam and Bohyung Han. Learning multi-domain convolutional neural networks for visual tracking. *CoRR*, abs/1510.07945, 2015.

[47] Paul B. Single shot detector tensorflow. `https://github.com/balancap/SSD-Tensorflow` [2017-02-21], 2017.

[48] Dean A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In D. S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 305–313. Morgan-Kaufmann, 1989.

[49] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[50] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[51] Peter M. Roth, Martin Hirzer, Martin Köstinger, Csaba Beleznai, and Horst Bischof. *Mahalanobis Distance Learning for Person Re-identification*, pages 247–267. Springer London, London, 2014.

[52] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Learning Internal Representations by Error Propagation, pages 318–362. MIT Press, Cambridge, MA, USA, 1986.

[53] Amir Sadeghian, Alexandre Alahi, and Silvio Savarese. Tracking the untrackable: Learning to track multiple cues with long-term dependencies. *arXiv preprint arXiv:1701.01909*, 2017.

[54] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., New York, NY, USA, 1986.

[55] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[56] Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[57] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, July 2014.

[58] Rainer Stiefelhagen, Keni Bernardin, Rachel Bowers, John Garofolo, Djamel Mostefa, and Padmanabhan Soundararajan. *The CLEAR 2006 Evaluation*, pages 1–44. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007.

[59] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[60] Ran Tao, Efstratios Gavves, and Arnold W. M. Smeulders. Siamese instance search for tracking. *CoRR*, abs/1605.05863, 2016.

[61] Gerald Tesauro. Temporal difference learning and td-gammon. *Commun. ACM*, 38(3):58–68, March 1995.

[62] UC Santa Cruz. Viper: Viewpoint invariant pedestrian recognition. `https://vision.soe.ucsc.edu/node/178` [2016-12-15], 2007.

[63] University of London. Qmul underground re-identification. `http://personal.ie.cuhk.edu.hk/~ccloy/downloads_qmul_underground_reid.html` [2016-12-15], 2009.

[64] University of Oxford. Town centre dataset. `http://www.robots.ox.ac.uk/ActiveVision/Research/Projects/2009bbenfold_headpose/project.html#datasets` [2017-02-21], 2011.

[65] Joost van de Weijer, Cordelia Schmid, Jakob Verbeek, and Diane Larlus. Learning color names for real-world applications. *Trans. Img. Proc.*, 18(7):1512–1523, July 2009.

[66] Nam N. Vo and James Hays. Localizing and orienting street views using overhead imagery. *CoRR*, abs/1608.00161, 2016.

[67] Alexander Waibel, Toshiyuki Hanazawa, Geofrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Readings in speech recognition. chapter Phoneme Recognition Using Time-delay Neural Networks, pages 393–404. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[68] T. Wang, S. Gong, X. Zhu, and S. Wang. Person re-identification by discriminative selection in video ranking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(12):2501–2514, Dec 2016.

[69] Wei Lee. Single shot detector caffe. `https://github.com/weiliu89` [2017-02-21], 2017.

[70] Kilian Q. Weinberger and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.*, 10:207–244, June 2009.

[71] B Widrow. An adaptive 'adaline'neuron using chemical 'memistors', 1553–1552, 1960.

[72] Bo Wu and R. Nevatia. Tracking of multiple, partially occluded humans based on static body part detection. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 951–958, June 2006.

[73] Sergey Zagoruyko and Nikos Komodakis. Learning to compare image patches via convolutional neural networks. *CoRR*, abs/1504.03641, 2015.

[74] R. Zhao, W. Ouyang, and X. Wang. Learning mid-level filters for person re-identification. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 144–151, June 2014.

[75] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark. In *Computer Vision, IEEE International Conference on*, 2015.

[76] Liang Zheng, Yi Yang, and Alexander G. Hauptmann. Person re-identification: Past, present and future. *CoRR*, abs/1610.02984, 2016.