Universidad de Alcalá
Escuela Politécnica Superior

Máster Universitario en Ingeniería Industrial

Trabajo Fin de Máster

Design, implementation and evaluation of automated surveillance systems

Author: David Valdivieso López

Advisor: Javier Macías Guarasa

2018

UNIVERSIDAD DE ALCALÁ

ESCUELA POLITÉCNICA SUPERIOR

Máster Universitario en Ingeniería Industrial

Trabajo Fin de Máster

Design, implementation and evaluation of automated surveillance systems

Author: David Valdivieso López

Advisor: Javier Macías Guarasa

Resumen

El reconocimiento de patrones ha conseguido un nivel de complejidad que nos permite reconocer diferente tipo de eventos, incluso peligros, y actuar en concordancia para minimizar el impacto de una situación complicada y abordarla de la mejor manera posible. Sin embargo, creemos que todavía se puede llegar a alcanzar aplicaciones más eficientes con algoritmos más precisos. Nuestra aplicación quiere probar a incluir el nuevo paradigma de la programación, las redes neuronales. Nuestra idea en principio fue explorar la alternativa que las nuevas redes neuronales convolucionales aportaban, en donde se podía ver en vídeos de ejemplos la alta tasa de detección e identificación que, por ejemplo, YOLOv2 podría mostrar. Después de comparar las características, vimos que YOLOv3 ofrecía un buen balance entre precisión y rapidez como comentaremos más adelante. Debido a la tasa de baja detecciones, haremos uso de los filtros de Kalman para ayudarnos a la hora de hacer reidentificación de personas y objetos. En este proyecto, haremos un estudio además de las alternativas de videovigilancia con las que cuentan empresas del sector y veremos que clase de productos ofrecen y, por otro lado, observaremos cuales son los trabajos de los grupos de investigadores de otras universidades que más similitudes tienen con nuestro objetivo. Dedicaremos, por lo tanto, el uso de esta red neuronal para detectar eventos como el abandono de mochilas y para mostrar la densidad de tránsito en localizaciones concretas, así como utilizaremos una metodología más tradicional, el flujo óptico, para detectar actuaciones anormales en una multitud.

Abstract

Automatic surveillance system is getting more and more sophisticated with the increasing calculation power that computers are reaching. The aim of this project is to take advantage of these tools and with the new classification and detection technology brought by neural networks, develop a surveillance application that can recognize certain behaviours (which are the detection of lost backpacks and suitcases, detection of abnormal crowd activity and heatmap of density occupation). To develop this program, python has been the selected programming language used, where YOLO and OpenCV form the spine of this project. After testing the code, it has been proved that due to the constrains of the detection for small objects, the project does not perform as it should for real development, but still it shows potential for the detection of lost backpacks in certain videos from the GBA dataset [1] and PETS2006 dataset [2]. The abnormal activity detection for crowds is made with a simple algorithm that seems to perform well, detecting the anomalies in all the testing dataset used, generated by the University of Minnesota [3]. Finally, the heatmap can display correctly the projection of people on the ground for five second, just as intended. The objective of this software is to be part of the core of what could be a future application with more modules that will be able to perform full automated surveillance tasks and gather useful information data, and these advances and future proposal will be explained in this memory.

Keywords: Automated Surveillance, Anomaly Detection, Deep Learning.

Contents

List of Figures

Chapter 1

Introduction

In this section, we will talk about our motivation and ambition that made us work on this project. Right after, we will share our vision in this matter, and we will specify the distribution of the chapters in this memory.

1.1 Motivation

Video Surveillance is an effective way to watch, control and check the number, flow and behavior of people, and it offers the possibility of extracting valuable information if we analyze it carefully. It is also a tool that can enhance the security systems and that can offer us a better perspective of what is happening in certain places: Inside a store, where owners can check that nobody is stealing, in open spaces to check that nothing wrong is happening or the crowded underground's corridors for flow control purposes for example.

Nowadays we are living in a time were the technology is getting more and more sophisticated and evolving year by year. In surveillance, with the increasing number of events that happens everywhere, is more common to see how much more crowded places can become and how sometimes, easy surveillance tasks can get more difficult to control when one person for example, needs to check simultaneously different incidents going through several screens. It could be said that rather than difficult, it can be a repetitive and boring task that could be automated or at least, it is easy to predict that these repetitive jobs could be aided with applications that automatize some of their parts.

Certainly, this is why traditional surveillance methods are becoming obsolete when we face this kind of events: Special sales day in shopping malls, big parking's with high transit density, industrial procedures where a several objects need their own tracking or barriers that need to be observed all the time, are some examples that demands a vision and control that could be enhanced by the advances of technologies and research that are being done worldwide.

Thanks to the improvements achieved in perception and systems with higher specifications and more powerful computers becoming more affordable, now is possible to think in the possibility of automating those processes and increase their effectiveness. For example, at the 2008 Beijing Olympic Games, MATE intelligent video was used, the leader technology in intelligent video surveillance system and video analytic systems. This kind of software can track suspicious behaviors, like people roaming in places they should not, high occupancy rates in some areas or tracking lost suspicious objects and, as result of those, throw an alarm to let the security surveillance group in charge act accordingly.

It must be pointed that, given the continued terrorist threats in the past, these applications can be in charge of tracking and detecting objects that can be suspicious and act consequently to prevent disasters in our society. There are many companies for which its core business is security and as a way of specialization, they are creating a range of software products that provide a better view of all the things that are happening inside big areas (as industries, malls or casinos) and, on the other hand, gather data that could help to shops to study if their corridors have good transit, or how much time people spend at certain places in the shop and reorganize their space distribution as a result.

1.2 A framework for automated surveillance systems

Patterns recognitions are getting a complexity level that allow us to recognize different kind of events or even threats and act accordingly to minimize the impact of a challenging situation and handle it in the best possible way.

Still, some improvement can be done, new algorithms can be developed. This project aims to use one of the new programming paradigms, the neural networks, as the main base for the program that this project presents, where this set of algorithms will be used to recognize people and other objects to establish the study of different behaviors and events.

This master's thesis approaches the use of YOLO [15], a high-efficiency network that outperform most of the other alternatives that are available right now. In the next section, we will explain briefly the different neural networks that have been used for determining the classification and location of people and objects for our tasks in this project.

There will be three parts in this project that will rely on YOLO and the use of traditional computer vision methodologies to contribute solutions with a quality that can be compared to current solutions in the market and even try to push some of them ahead. The main part of this project will be a detector for abandoned object, and more specifically, bags and backpacks. The program will check when a backpack is linked to a person or if it is the case, it is alone and abandoned.

Using this one-stage neural network, a second application will be creating a heatmap detector. By detecting correctly, the people in the images, it can be easily obtained the position of the people and therefore, build up a heatmap.

Lastly, using K-mean clustering, we will reproduce the pattern of normal activities in open places to detect when an abnormal activity is produced, being this an unusual behavior and alteration of the normal order. From the work obtained in this project, these parts can be improved and can evolve to add more options so the information can be studied in time and give a better feedback in the future, increasing the possibilities for different kind of abnormal activities, such as measuring and tracking the data so it can be easily diagnosed when abnormal cases are happening while the system is running.

1.3 Memory Structure

In the beginning, this memory contains a brief introduction about general thoughts on this matter: A little overview of why this field drags attention for artificial vision algorithms and how some events and the evolution of technology have allowed this science to create more powerful frameworks to detect abnormal behaviours among the crowd. It was also complemented with the objective that, in first instance, is what make this project try alternative solutions.

In the chapter State-of-the-art, a summary of what companies offer with be described, highlighting the products that are more repetitive on the lists. On the other hand, it will also include some of the solutions that different universities have brought in their research, trying to elaborate the evolution through different proposals.

The framework developed in this project is mainly based on deep neural networks, which will be followed in the next chapter explaining where they come from and what is their actual state, explaining after a long research why this project has chosen after the considerations of reading and checking its characteristics.

Finally, we will explain the modules that compose it, explaining in which files is contained the code that makes this possible and explaining in deep each step of the most important statement and iterations of the modules.

Chapter 2

State of the art

2.1 Introduction

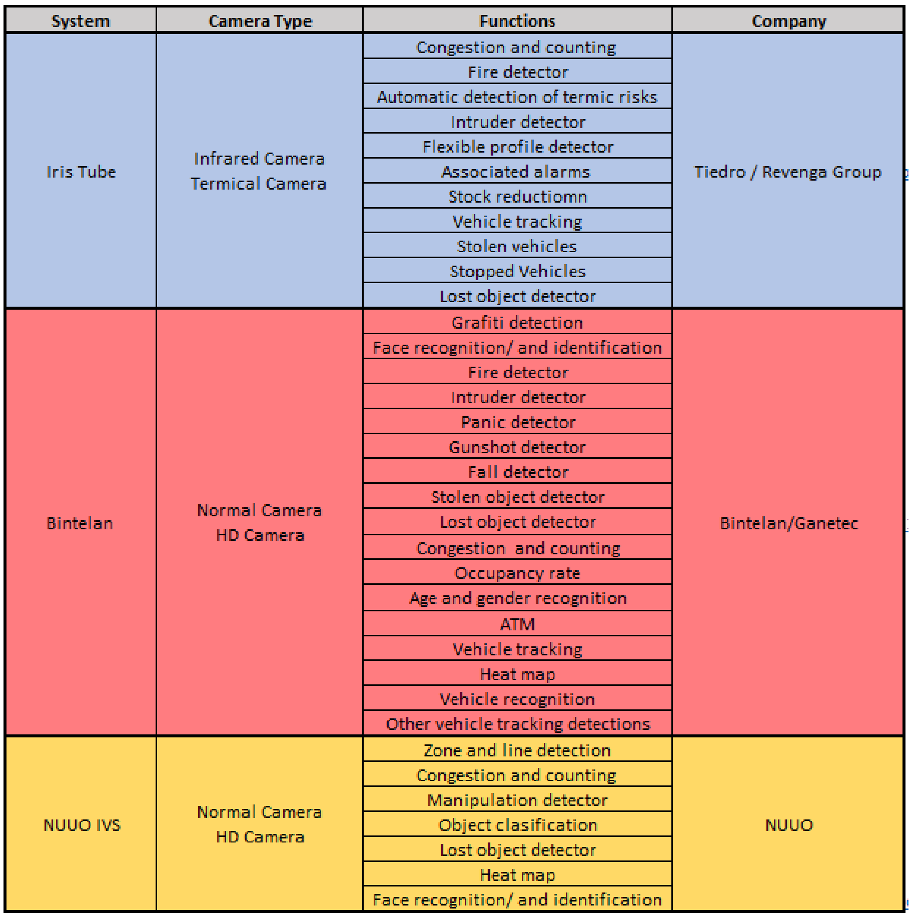
In the following section, we will talk about companies that already have produced applications for automated surveillance, and we will show which functions are the most common among them.

Right after, it will be shown a detailed description of the evolution of the different deep learning techniques used for detectors and classifiers, which is the topic that this project has invested most of the time in. To end with the state-of-the-art, it will be shown projects that already have been carried out by other universities in the abnormal crowd behaviour detection field. We will present conclusions for both parts, describing which approach we have had in mind to take.

2.2 Technologies and companies that have developed an automated surveillance software

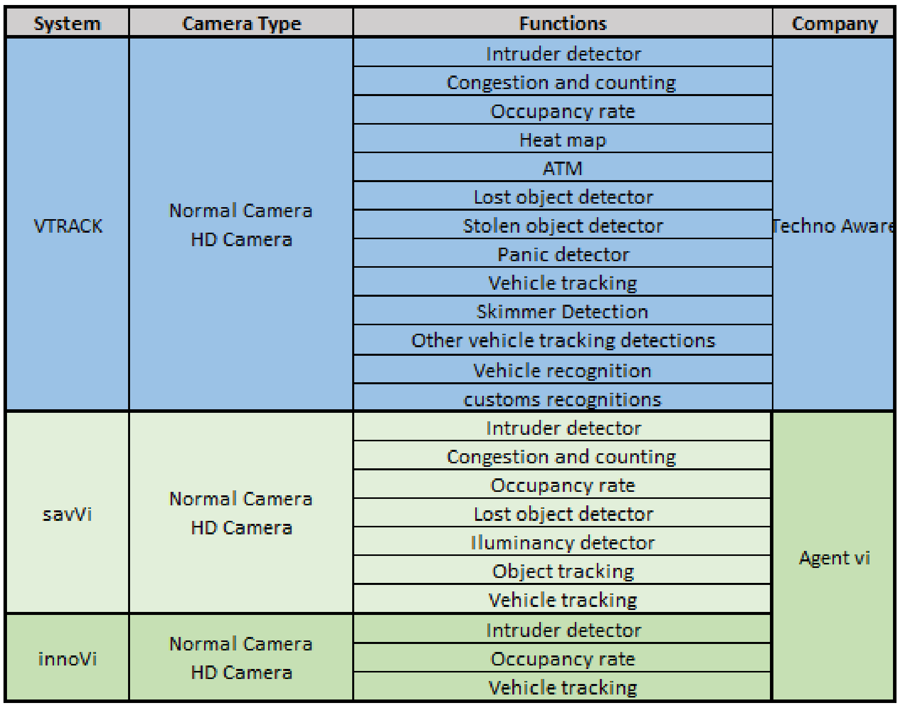
If we take a look of the companies that are offering high quality solutions for the problems we have spotted, we can find different products that may suit the situations we already have proposed. For example, can we know if suddenly, is there a suspicious bag that has being left in a train station? Or looking this kind of systems in a more profitable way, can we know where is the place of our shop where people are spending most of their time? Or maybe when do we get the maximum amount of people in a day? Or when do we get the minimum amount?

Some of the current companies and what they offer are gathered in the figures 2.1 and 2.2



System	Camera Type	Functions	Company
Iris Tube	Infrared Camera Termical Camera	Congestion and counting	Tiedro / Revenga Group
		Fire detector	
		Automatic detection of termic risks	
		Intruder detector	
		Flexible profile detector	
		Associated alarms	
		Stock reductiomn	
		Vehicle tracking	
		Stolen vehicles	
		Stopped Vehicles	
		Lost object detector	
Bintelan	Normal Camera HD Camera	Grafiti detection	Bintelan/Ganetec
		Face recognition/ and identification	
		Fire detector	
		Intruder detector	
		Panic detector	
		Gunshot detector	
		Fall detector	
		Stolen object detector	
		Lost object detector	
		Congestion and counting	
		Occupancy rate	
		Age and gender recognition	
		ATM	
		Vehicle tracking	
		Heat map	
		Vehicle recognition	
		Other vehicle tracking detections	
NUUO IVS	Normal Camera HD Camera	Zone and line detection	NUUO
		Congestion and counting	
		Manipulation detector	
		Object clasification	
		Lost object detector	
		Heat map	
		Face recognition/ and identification	

Figure 2.1: First list of companies and functions that their software include.



System	Camera Type	Functions	Company
VTRACK	Normal Camera HD Camera	Intruder detector	Techno Aware
		Congestion and counting	
		Occupancy rate	
		Heat map	
		ATM	
		Lost object detector	
		Stolen object detector	
		Panic detector	
		Vehicle tracking	
		Skimmer Detection	
		Other vehicle tracking detections	
		Vehicle recognition	
		customs recognitions	
savVi	Normal Camera HD Camera	Intruder detector	Agent vi
		Congestion and counting	
		Occupancy rate	
		Lost object detector	
		Iluminancy detector	
		Object tracking	
		Vehicle tracking	
innoVi	Normal Camera HD Camera	Intruder detector	
		Occupancy rate	
		Vehicle tracking	

Figure 2.2: Second list of companies and functions that their software include.

As we can see, most of the main functions that the companies offer are based on person tracking, vehicle tracking and object tracking. To understand correctly what the functions are they provide and the cameras that support them, we are having a further explanation.

Depending on which type of camera we have, as it is seen in the previous tables, the main type of camera used in surveillance activities are normal and HD cameras but, as we see with Iris Tube system, the algorithms are used with thermal and infra-red cameras.This type of cameras can be a satisfactory solution for seeking elevated temperature problems in certain parts of machines that ma y overheat with some processes or even if a fire starts.

It is also a good option if we want to detect people in an environment with many objects so the camera can capture the heat that is released by the bodies. On the other hand, high-tech cameras with great algorithms can also have these features if they can detect the smoke or the fire by discriminating colors in the images but, as we said, the algorithm must be good when this kind of situations are happening.

The difference between tracking people and tracking vehicles are mainly the size of a person compared to a vehicle and the speed they can reach. Also, we can do the same with lost objects and we can detect changes that are being recorded with a static background and knowing when an object is being dropped.

- Tracking people

  Tracking people is a basic function in automated surveillance systems as we see in the figure 2.3. The algorithm does RGB discrimination with the footage and seeks for brightness changes in a group of pixels, so we can figure out that a person is in our image.

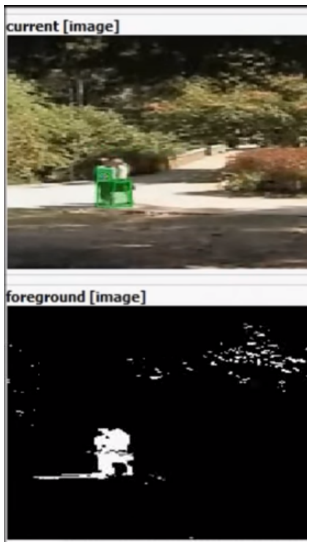


Figure 2.3: Human tracking footage from TechoAware [4]

- Facial + Radio Frequency Identification and License Plate Recognition

  A step further of tracking people is the facial recognition. More complex algorithms with higher quality cameras allow the software to get the information from faces captured in the images and, once they have captured the key points from the face, they are compared with the faces stored in a database.

  Same analysis can be done with vehicles, as the example we see in the figure 2.4. The recognition needs to be different because for vehicle plates we need to recognize numbers and letters in different positions, but the principle should be the same.

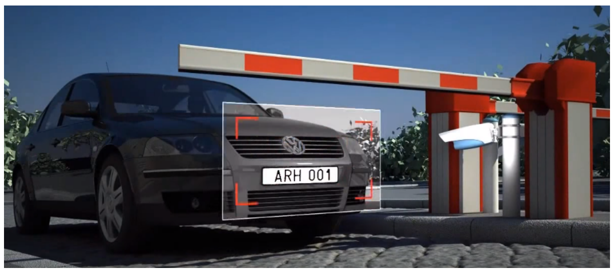


Figure 2.4: LPR tracking example by Omnitec Security [5]

- Counting, congestion or intruder detector

  By defining an area and tracking people, we can determine if this area is being overcrowded, we can count how many people are going through this area, etc. Useful information can be obtained if we analyze the data correctly and can determine how many people visit a shop or the rate of people who buy products in a store. In the figure 2.5, we can see an example of a camera recording and counting the people entering into the store and the proper application, counting.

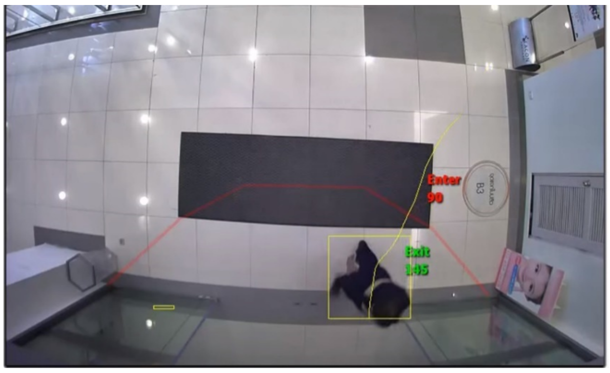


Figure 2.5: footage of counting people by NUUO [6]

- Fighting, panic and other patterns recognitions

  More complex algorithms developed. For example, if the camera captures a person running and usually people don't run through a corridor where camera is recording, we can tell that an event is happening in that moment. On the other hand, if a person track starts start having collisions with another one, we can tell that they may have started a fight.

  If we also can combine audio track with the images captured with the camera, we could also get a better idea of the thing that could be happening. Also, it is used a combined system of audio sensors and cameras nowadays by a system called Shot Spotter to know if there is a gunshot out in the city and determine by the audio track of different sensors around the city, where to determine where a pistol have been used. For indoor, it could also be developed following the same criteria.

2.2.1 Conclusions

Automatic surveillance systems are getting more and more sophisticated with the increasing calculation power that computers are reaching. Also, traditional ways where people watch a screen to get the information have become obsolete: algorithms that give us valuable data to know which problem we may have in an indoor place due to a bad disposal of our industry or our shop, can be developed with the tools available right now.

Patterns recognitions are also getting a complexity level that allow us to recognize different kind of threats and act accordingly to minimize the impact of a challenging situation, handling it in the best possible way.

But some improvement can still be done, new algorithms can be developed by combining technologies that are at our reach, like infra-red cameras, normal cameras and audio tracks from the video cameras.

Still, we think that with the new technologies that are emerging from research as it will be discussed later, more efficient detectors can be created. This project will have the objective of exploring what the new paradigms can offer and use them to set a start for a new application that can be competitive in this market.

This project will be focused in a determined pattern recognition (abandoned object detection) and a panic detector (abnormal crowd behavior). Nevertheless, it would be convenient to continue adding into this framework, basic modules that have been seen that are used by other companies and don't require too much time to be developed, like tracking or counting.

2.3 Abnormal activity research

2.3.1 Introduction

In this part of the research, we will start reviewing projects related to heatmap and crowd density. Right after, we will describe the research we have done about the abnormal crowd activity behavior reports, which affects directly to what has been designed and implemented for the abnormal crowd activity detector in the software developed in this project. To end this section, we will also talk about applications based on other technologies for lost object detectors so it can be minded what alternatives can be taken or added into this application in the future.

2.3.2 Heatmap and crowd density

Following the definition, a heatmap is a graphical representation of data that uses a system of color-coding to represent different values. When it comes to crowd density, it expresses the density flow perceived in an image in a scale of colors (or a scale of intensity).

Density estimation is one of the objectives which most of the research projects based on heat maps aim for. Older methods were based on people moves, like [7] tracked with local features and where they would first extract these features to study the contents of each frame to finally conclude with an analysis. Then, using a robust optical flow algorithm and a point rejection step using forward-background projection, the static features can be subtracted. To end, thanks to Gaussian symmetric kernel function it is possible to determine a density estimation. We can see the results in the figure 2.6.

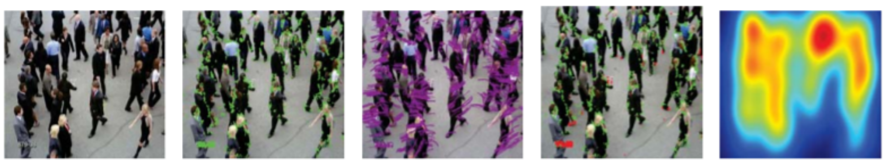


Figure 2.6: Heatmap obtained with the application developed by [7]

The objective is to know the density on a certain image. In this case, higher local features represent low crowd density. These local features are specifically obtained through corner detection helped by machine learning techniques in order to automatically find optimal segment test heuristics.

The final density map is obtained with an equation that is fed with a summation of these local features in their respective position extracted from a particular frame and the bandwidth of the 2D Gaussian Kernel. Objects with low or any motion would not be considered as person given that it is used a forward-backward verification scheme where the resulting position of a point is used as an input to the same point estimation step from the second frame into the first one, having a motion tracking in its own form.

As it is seen in [8], Lius brings another solution. Again, this is oriented towards a public safety view where the heatmap is used for crowd density control more than that for a commercial used, where it could be more interesting to know where the people spend most of their time in certain areas. Still, it is interesting to have this approach in mind for possible future ideas as we see in the figure 2.7

With the more commonly extended use of CNNs in different fields, an in this solution the use of an CNN has been adopted too. Different CNNs with receptive fields of different sizes are used to solve the problem with perspective. Using a multi-column CNN to estimate the density values around people heads have brought some successful results, but there are cases where irrelevant objects or backgrounds that badly affect the result of the density map.

To overcome this problem, they have used what it is called a MCNN-boost, a new network trained with the error of the first network and without a ReLU layer to learn negative valued errors. In other words, the second network is trained against the predicting of the error from the first network. For the later test, both networks are combined by an element-wise summation layer.

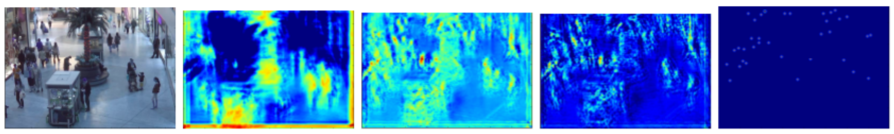


Figure 2.7: Heatmap obtained with the application developed by [8]. The position of the people is marked with a dot where their heads are.

As we can see, results of the boosting are good in the fourth image in this figure. And even though the result is good and represents the position of the head of the people giving an idea of how many people are in the scene, it does not represent a good visualization of what we are looking for in this project due to the lack of context in the density map result.

Another interest application is [9] as we can see in the figure 2.8, where the accumulation process to add new images for the heatmap is only activated when objects are moving in the scope of the camera. This project has also applied an algorithm for projective transformation of image to achieve better accuracy of the moving object density map.

To create the density map, the object occurrence density is calculated with the detections obtained from the frames. Moving objects are detected with background subtraction, and each detected moving object will be described by the coordinates of its center. In this case, a Kalman filter is used to detect the trajectories of the objects. After processing of a given time segment of video sequence, the density map is generated with the values obtained in the accumulation matrix.

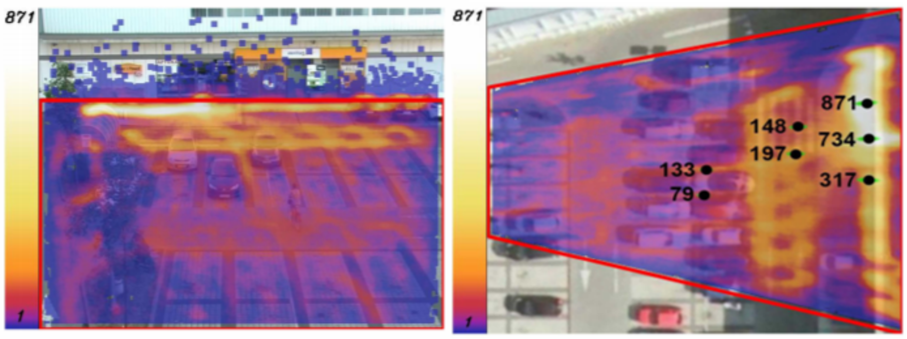


Figure 2.8: Heatmap obtained with the application developed by [9].Points with different levels of concurrence are shown in the image.

The number of moving objects that passed across a certain point can be determined thanks to the tracking that is done with the filter, obtaining then a clear idea of the numbers that can be handled with the density map. As they conclude, this system brings a definite improvement of people's safety in smart cities.

It would be a good addition to add certain information on the screen in our application as this project has done, displaying relevant information and having then an idea of the scale in numbers of the calculated density.

2.3.3 Abnormal crowd activity

As we have been seen earlier, there are many different companies that have tried to sell a package of products based on computer vision technologies to detect relevant cases that may happen in public and working scenarios. We can be found as well, researches that takes approaches that aim to solve the very same problems but relaying on different methodologies to impact in a better outcome.

Reliability in different environments and situations pushes multi-purpose systems that could handle the needs for each situation. Forbidden activities take a range of events, it could be detecting a person or an object going into a restricted area, or a higher amount of people bigger than allowed. Automated surveillance not only include the detection of these, but also aims to study statistically how is the occupation of a certain area or what is the rate of these events or behaviours that can be atypically happening among a crowd of people.

Different approaches can be taken to elaborate an algorithm that describes a normal behaviour and that can be give a positive value when an event occurs and doesn't follow the norm. Discerning deeper, we can tell when an event is happening individually to a person among the crowd or to the crowd itself.

The research done about this topic is immense but most of the times, based in the same technologies with different tweaks, changes or merging different ideas of a group of projects in order to make them more efficient. Sometimes, improvement is achieved in the quality of the detection, sometimes in the capability of adaptation of an algorithm to different types of environments.

For individual recognition and study of behaviours, we can see that methods like Long Short-Term Memory Networks (LSTM [16] and structures based on Hidden Markov Models [17, 18], Social Force Model [19] send to lead recurrently as the main approaches to detect specific behaviours and therefore, abnormal behaviors [20].

Abnormal activity recognition based on HMM defines and classifies a series of behaviors into states and possible states, training the model with normal and abnormal behaviors so the system can identify which one is more related to the current input. For example [21], The project developed by [22] aims to detect abnormal activity in indoor environments by detecting silhouettes through low level image detection techniques and using R transform to "measure" the shape, classifying different states in different histograms.

The time component used to check if any event is happening in the footage is a measure that is not easy to evaluate. Action recognition must be detected by the accumulation of actions in certain frames, which needs to take in consideration the value of the pixels and its evolution along the process.

There are networks based on recurrent neural networks (R-NN) and others go with the use of Long Short-Term Memory units (LSTM), given that both can handle the use if spatial and temporal information. A usual R-NN has short-term memory, and with the addition of LSTM it can obtain an improved long-term memory. This combination is done by [23] in order to learn to selectively focus on part of the video, classifying the action after checking not the whole video but some frames. Having a CNN underlying the LSTM helps to classify sequences directly and do temporal pooling of the features prior to the classification. Frame level classification can be achieved thanks to the analysis given by a normal CNN, where the network predicts the probabilities for the next location and possible classes of actions [24]. In contrast, other researchers prefer to use 3-D CNNs [25, 26], which can also perform well on different discriminative video tasks too when it is combined with a R-NN encoder-decoder framework or a linear classifier [10, 27]

As these projects show [28, 29], here it is an example use of the use of LSTM combined with a CNN. We can see the structure of this example in the figure 2.10.

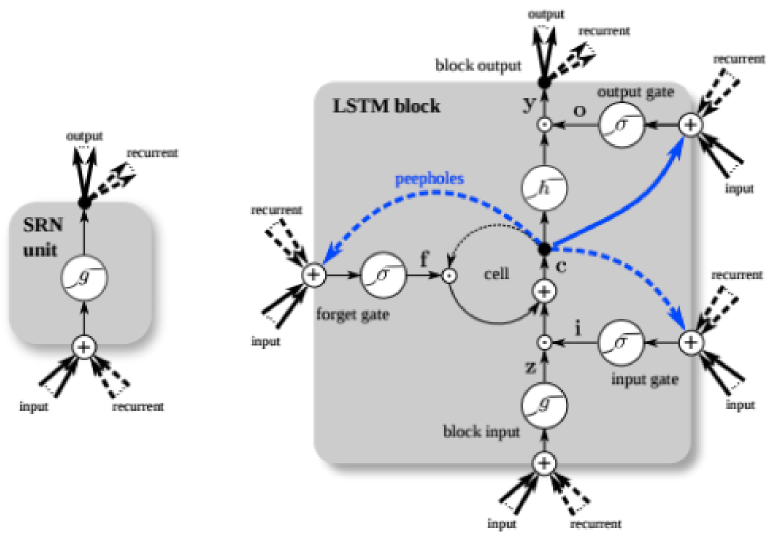


Figure 2.9: LSTM with peephole connection added by Gers and Scmidhuber

Usually, deep neural networks transport the information between layers through multiplications, therefore derivatives are susceptible to vanishing (get multiplied by a value much lower than 1) or exploding (get a high value). Effects of exploding gradient can be solved with truncated values, but vanishing gradient, after applying functions like sigmoid to trigger the value and maintain the information, can flatten the slope after reiterative operations. LSTMs are a solution for this problem, and instead of use multiplication of the previous data, which could provoke the loss of information if the values of one frame turns to be low enough, is changed for an adding operation, which allows to preserve the conditions of the data. LSTM also has a feature called the ”forget gate layer”, which consist in a sigmoid function that evaluates the current input, the previous output and decides if this information is going to be added into the cell state, which controls the influence that previous inputs have in the current output.

These options have a good performance in a determined group of behaviors, but they still need improvement, giving mismatches in the results [23]. Other projects have aimed to determine behaviors by using a more modest analysis behavior classifier based on predictive search, but results display overfitting and are not conclusive [30].

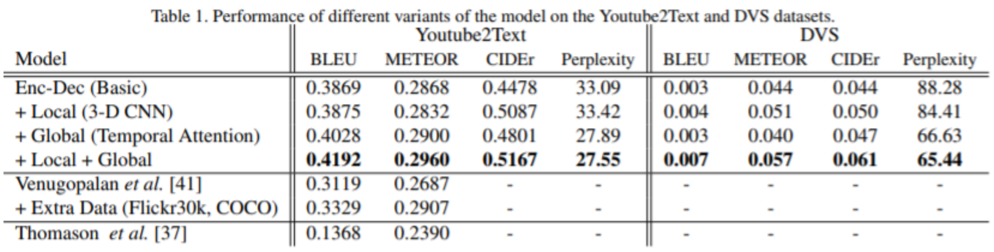


Table 1. Performance of different variants of the model on the Youtube2Text and DVS datasets.

	Youtube2Text				DVS			
Model	BLEU	METEOR	CIDEr	Perplexity	BLEU	METEOR	CIDEr	Perplexity
Enc-Dec (Basic)	0.3869	0.2868	0.4478	33.09	0.003	0.044	0.044	88.28
+ Local (3-D CNN)	0.3875	0.2832	0.5087	33.42	0.004	0.051	0.050	84.41
+ Global (Temporal Attention)	0.4028	0.2900	0.4801	27.89	0.003	0.040	0.047	66.63
+ Local + Global	**0.4192**	**0.2960**	**0.5167**	**27.55**	**0.007**	**0.057**	**0.061**	**65.44**
Venugopalan *et al.* [41]	0.3119	0.2687	-	-	-	-	-	-
+ Extra Data (Flickr30k, COCO)	0.3329	0.2907	-	-	-	-	-	-
Thomason *et al.* [37]	0.1368	0.2390	-	-	-	-	-	-

Figure 2.11: Performance of different variants of the model developed by [10] to describe action on different datasets.

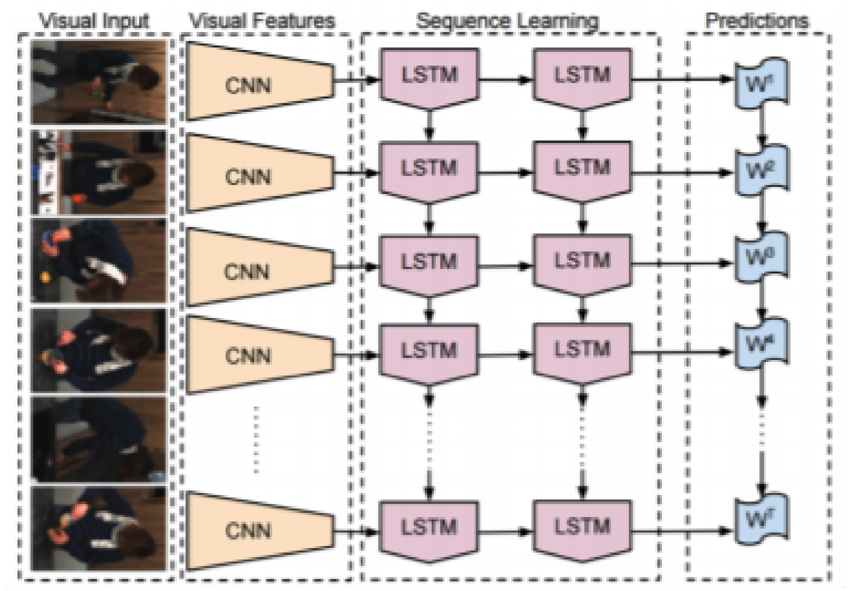


Figure 2.10: Image of the structure followed by projects that use a mixture of CNN and LSTM to finally retrieve an action classification of the whole process.

[10] needs some improvement but still can capture correctly actions in different videos. We can see the performance of its application in the figure 2.11.

Using what [24] calls a convolutional LSTM architecture [31], which consist in an encoder that contains a convolutional layer followed by a tanh non-linearity and a spatial max-pooling with sub-sampling layer, and a decoder, which follows the same process but changes the non-linearity factor of the tanh for nearest-neighbor spatial up-sampling.

Following this architecture [24], training the network with samples of normal activity can detect when abnormal activities events happen, such as detect when a bike goes through a path where only people walk in the training examples, achieving a good grade of competition among other state-of-the-art anomaly detection methods like [32], which also works with autoencoders but filters the input information using HOG+HOF histograms.

We can find the same approach in other projects too, for example in [11] as we see in the figure 2.12,where MHOF (Multiscale HOF) and EOH (Edge Orientation Histogram) are efficiently used to feed the algorithm based on compact projection [33] for motion context similarity search. More focused in indoor events [34] presented a robust study that evaluates crowed scenes and developed a highly adaptative algorithm capable of perform almost as intended in different environments, having certain limitations and focusing the interest on wrong directions and speed limitations.

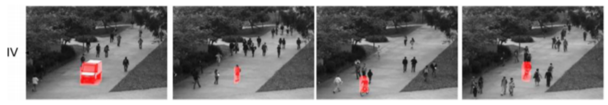


Figure 2.12: Individual abnormal activity detection images from [11]

As it was commented [8], Social Force Model was an approach followed by numerous projects. These models are based on the previous work brought by Harper and Brothers in the research Field Theory in Social Science, 1951. The idea behind this theory can be summarized in the reaction capacity of the pedestrian, which tends to be automatic and based on their experience in what reaction is the best. The model counts with attraction forces, like group of pedestrian going to the same end or repulsive forces, like borders of buildings, walls, street obstacles or even other pedestrian going in opposite directions, that could lead to a change in the path to reach the destination that they were aiming for.

Studying crowd behavior through this model could have successful results as [35] indicates, where the detection of riots or chaotic acts can be analyzed and located and being defined as an eccentric state of the crowd interactions. To achieve a good detection, their method is based on a holistic approach, and it consist in placing a grid of particles over the image and moving them with the underlying flow field.

Average optical flow is calculated in the grid and the motion is transmitted to the particle grid. The effective velocity is computed using bilinear interpolation of the neighboring flow field vectors. Therefore, particles move depending on the average speed of their neighborhood.

For training, the forces are mapped back to construct the force flow, and random portions of the image are extracted where the force flow is not zero. The values are included in a codebook using K-means clustering and to detect events, only a high force of flow triggers the abnormal event based on the values obtained by K-means. Test with optical flow are also done, even though the procedure at this part is not clear.

Average optical flow is calculated in the grid and the motion is transmitted to the particle grid. The effective velocity is computed using bilinear interpolation of the neighboring flow field vectors. Therefore, particles move depending on the average velocity of their neighborhood.

One of the projects based on this idea [36], a grid of particles is set and their velocities are calculated using fourth order Runge-Kutta algorithm along with the average velocity in optical flow field. Introducing Social Attribute-Aware Force (SAFM), it is formulated that the interaction force is influenced by terms of scale, disorder, congestion and interaction forces in the image. Scale term is calculated through distance transform, the disorder attribute is measured with the local change of direction of the particles, growing larger when the chaos is generalized in a group. The congestion attribute is similar to the disorder attribute, but it determines the density of the crowd and the group velocity. If the value of the formula gets over a threshold, an anomaly would be detected.We can an example of this application in the figure 2.13

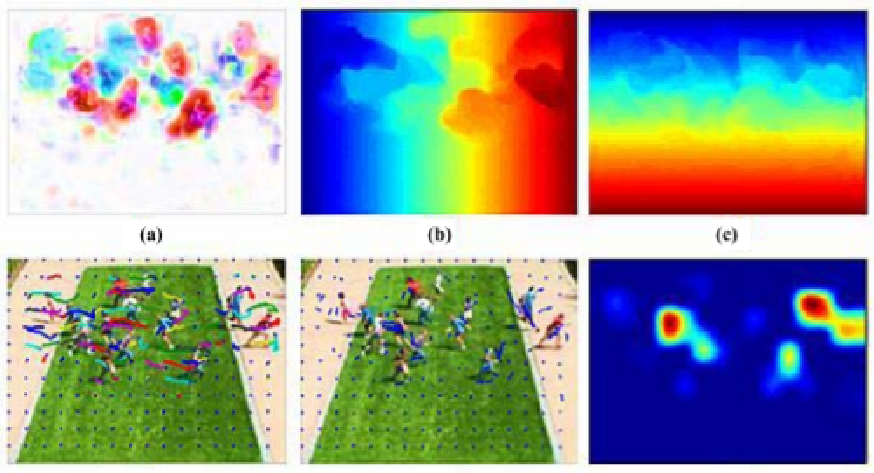


Figure 2.13: Process work of Social Attribute-Aware Force Model, where the optical flow is shown in HSV, they represent a particle advection scheme, the interaction force is calculated and the results of mapping the interaction forge to the image plane by bilinear interpolation are shown in the last image.

If we compare this project based on force model and the previous one made with spatial-temporal motion context, it is easy to see how in this case, force model does not perform as well as the previous method. Datasets where the results are exposed are different, but other methods as Adam or MDT are shown too and in comparison, we can see that spatial-temporal motion context do better in general. We can see an image of this application in the figure 2.14

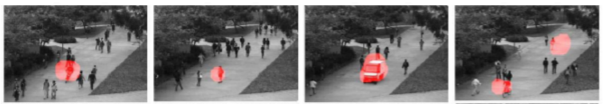


Figure 2.14: Image from video anomaly search in crowded scenes via spatio-temporal motion context.

Other approaches have been studied, where for example [12] expose that HMM is insufficient to detect abnormal events is not a great model for crowded scenes. This research has some similarities with the previous mentioned [35], where also follows a holistic method and believes that is better to study the behavior of the crowd as whole rather than following individual patterns.

The approach is similar: a grid of particles is placed and particle advection is performed to estimate the position in the following frames using optical flow interpolation. The trajectories are merged to have a representative motion of the movement detected. Again, using k-means clustering, an iterative clustering strategy is used to determine the number of clusters needed, starting first with a large number of clusters.

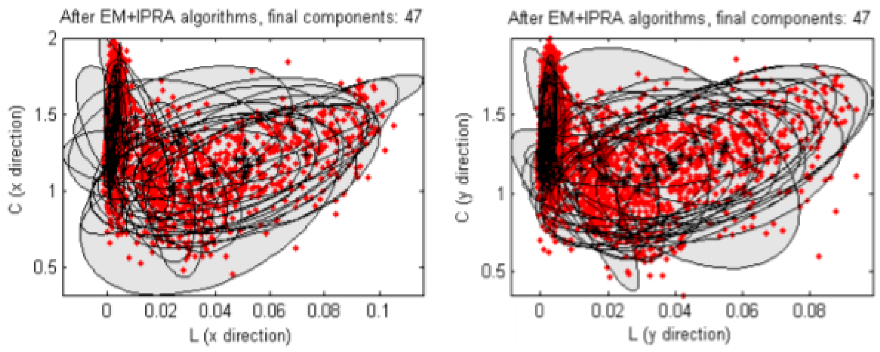


Figure 2.15: Figure extracted from [12]. Diagram of two chaotic features of x (left) and y (right) of learned 4-D mixture of Gaussian model.

By studying Lyapunov exponent, the correlation dimension and the mean of the flow of the coordinates x and y, this solution checks the growing distances between trajectories. After checking clips where only normal actions happen, gaussian mixture models (GMMs) are used to describe the probability density function of normality. We can see an image of the diagram in the figure 2.15.

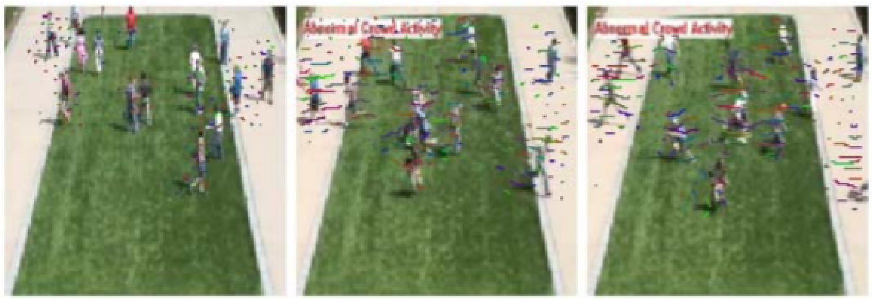


Figure 2.16: figure extracted from chaotic invariants of Lagrange particle trajectories

Finally, to detect anomalies in the videos, particle advection is performed in every clip with a certain number of particles. Chaotic features are calculated for six sequences and learn three normality models. Even though the report does not specify the process, they determine in which new clips abnormal events happen or not by likelihood, and finally, results describe an area under ROC of 0.99, compared with social force models that could reach 0.96 or pure optical flow with 0.84.We can see results in the figure 2.16.

2.3.4 Lost object detectors

Lost object detectors are something that have caught the attention of the people who develop new algorithms to track abnormal behaviours in computer vision. This section will not be as extended as the previous one, given that the number of papers in this matter is lower than the ones related to general abnormal crowd behaviour detection. Why? Maybe because the algorithm that is needed to detect an abandoned object can be easier to obtain, the variations in this field are lower (most of the time, it is based on background subtraction). In addition, there is already a solution that performs well with the dataset that is widely used for this purpose. Just as it is pointed in our motivation, the objective for the following projects is the same: Detect suspicious abandoned objects that can potentially be the origin of a terrorist attack in a public place. Firsts attempts already knew that these abandoned objects would have a stationary position and would have a recognizable backpack shape. Even though the first filters could detect somehow abandoned object in crowded spaces [37], we have noticed the success rate is not correctly valued and the process are not well described [38].

A powerful framework [37] was created couple of years later based on background subtraction based on three main steps: First, a background model is created with the first hundred frames and accumulated the pixel values of each frames into an image. The second step is the background subtraction process where new frames are compared to this model to detect new objects in the area. The third step is the removal of noise in the image and the contour length extraction, where the report describes that if the object is static, between two frames there will not be change of size. To detect if the static contour relates to a person or to an object, a complex formula based on the slope gradients of the contour detected classifies the object with a human and non-human label. As a result, the application will raise an alarm if the object is static in consecutive frames. It is a simple but an effective solution.

Various projects were done after this framework was created, but a new approach [13] with a new feature, a mixture of long-term and short-term state information, could return a back-tracing verification of the moment where the object was abandoned and therefore, check the frames to visualize the person who abandoned the object, showing a robust viability in the results. We can see an image of its results in the figure 2.17

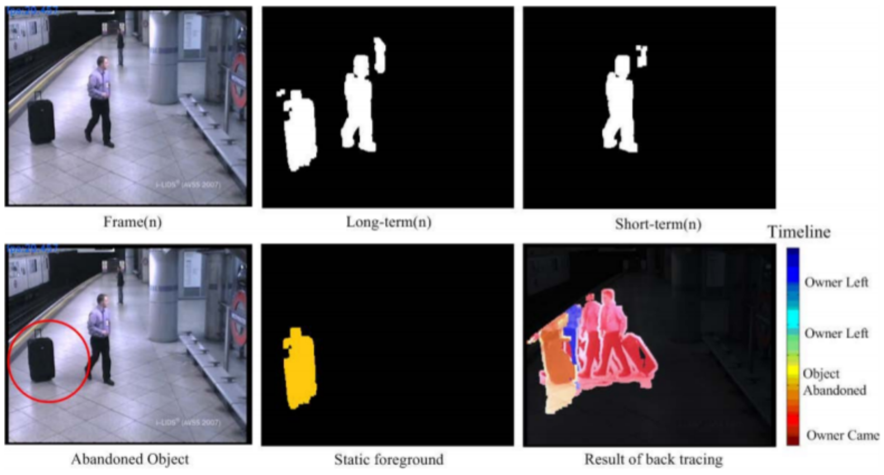


Figure 2.17: Image from the framework developed by [13]

This is performed with a simple pixel-based finite state machine (PFSM) model to detect static objects. To detect if the contour belongs to a person or if it represents an object, the object detection with discriminatively trained part-based models (the object detection with discriminatively trained part-based models) is used, fasten up with a convolutional procedure. To perform the back-tracing, the closest person to the luggage or object is denoted as owner for further tracking. The blob with the most similar color distribution will be used for reidentification purposes. Until the owner surpasses a certain threshold distance, the reidentification will be repeated. Only certain blobs are considered thanks to the spatial-temporal windows.

When the luggage is unattended for 30 seconds and is not picked up before that time or, if the owner walks further than a 3 m of distance between the bag and him, the application declares an abnormal event. As results are shown in their report, this is one of the state-of-the-art methodologies to detect abandoned luggage, reaching a total precision and recall in their experiments with PETS2006 [1] and AVSS2007 Datasets. They also created a dataset exclusively made of abandoned luggage videos called ABODA, which will be explore for this project.

To end this section, the last project that will be referred is [14] with a method based on background modeling and change detection. The background is continuously updated if there is certainty that the pixel belongs to the background. HOG is used to detect changes between two frames and specifically, between cells on a created grid. Filtering the changes, they obtain a binary map as a result, as we see in the figure 2.18

To separate the new objects, they use a technique called blob separation, where scores are given depending on the matching scores with their previous position.

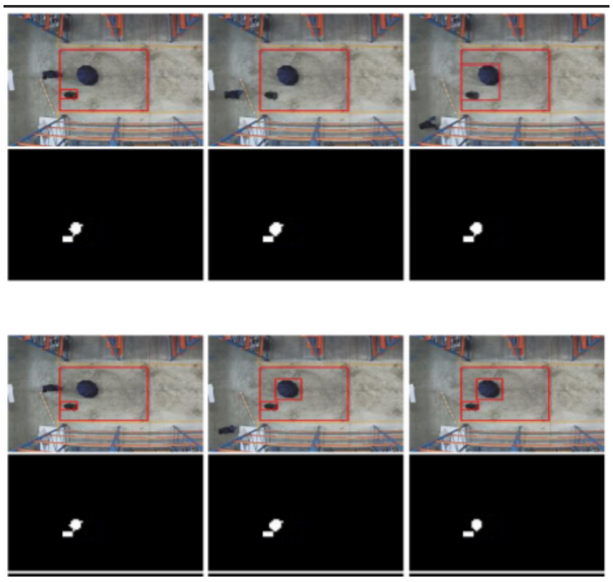


Figure 2.18: Object detection results obtained by [14]

As a result, the authors describe that this method, without blob separation, will show a problem where two objects with short distance in between would be recognized as only one object. Due to the continuous verifying process, the objects would uncheck their lost status once they are removed from the image. They also prove that this method can detect small objects, and even though background subtraction can do the same, identify two different objects independently is a great contribution.

2.3.5 Conclusions

As we can conclude after this research based first on the products and services offered recently by companies dedicated to automated surveillance, and secondly based on the reports provided by several research groups exposing different techniques to study abnormal behaviors, on one hand individual abnormal activities and on the other hand, abnormal crowd activities, we can notice a gap between what is offered and what has been researched in the last years.

Research projects brought by universities are different and show a variety of methodologies and approaches for similar purposes. It has been reviewed through the previous section that the methodologies they have used could have been based on good ideas but in the end, they did not lead to good results. Nevertheless, we have also seen methodologies that show a high percentage of positive results, which implies that they could have viability to be used in commercial products if they are properly adapted.

Regarding to the methodologies, we have reviewed the progress across the crowded models. Force models have had a great impact and evolution, showing that they are capable to detect abnormal events, mainly crowd dispersion events. Nevertheless, the most impressive results are based on Lagrangian mechanics as we reviewed before.

On the other hand, it seems that LSTMs combined with convolutional networks has the best performance to detect individual abnormal activities, proving that other approaches based on HMM and more conventional artificial vision techniques are not as effective as them.

When it comes to abandoned objects, it is easy to notice that there are not as much research reports as it can be found for abnormal crowd detection, and this can be due to the complex behavior that the last can present. Still, we believe that improvement can be brought on both spaces and with the evolution of deep neural network, important details can be subtracted from the frames.

Chapter 3

Theoretical soundness: Deep neural network, Kalman Filter and Optical Flow

3.1 Introduction

We decided to start the development of a software that could gather different solutions to provide an alternative option to handle a low effort and repetitive task like surveillance can be, where a great part of the time, monitoring dozens of screen can be as challenging as boring when nothing happens for hours and even days, which can lead to a lack of attention in certain moments.

Based on a training data, a model can be created to detect when events that does not follow the behaviours patterns established in the images, highlighting a zone when an anomaly is detected there. Hence, a semi-supervised surveillance system is believed to be the most effective way to perform this activity, where the person who is monitoring can still check the global situation while if an abnormal event is detected, the proposed application would inform the security group that an event out of the normal patterns is taking place and act in consequence.

In order to create an application with these features, the training data needs to have enough viability, meaning that if there are not enough normal events handled in the dataset, the detection of abnormal evens can be shattered with false alarms.

Still, automated surveillance is a part of computed vision that brings so much attention and that nowadays has several datasets that allow new investigations to test their new algorithms and it has been already proved that it is possible to detect in almost real cases the detection of these behaviours.

There is a long list of activities that can be detected thanks to computer vision and, specifically when it comes to surveillance, there are different task that we could classify from a lower to a higher emergency. For this application, we have proposed two main approaches to detect anomalies that is included in the group of major priority emergency events, which are the detection of suspicious lost object and detection of crowd anomalies (that could be classified as panic attacks or stampedes). In addition, we used part of the process done for the object lost detection to create a visual presentation of heat maps, a part of the framework that would need an update in the future to display the information in a more comprehensive way (such as displaying the number of concurrency in the hot spots).

In the following section, we will show the main technical resources that already exist and that haven been used to carry out this project. To learn what they are, we have reviewed the utility of these tools and a deep research has been done.

First, we will be present the methodology used for lost object detection: we have done a research about the classifiers based on neural networks to know all the alternatives that have been developed in this field. Right after, we will explain how Kalman filters work, given that it has been used to predict the position of the objects when the detector struggles to find them in certain frames.To end, we present the theoretical bases of optical flow estimator and k-means, which are the main technical resources used to detect abnormal crowd activity in this project.

3.2 The backbone of the project: Neural Networks

3.2.1 Introduction

In this part, we will introduce a brief history of the evolution of the artificial neural networks (NNs), starting from the first filters used in the Support Vector Machines (SVM) for classification, and evolving to the first classifiers based on neural networks and ending with the recent state-of-the-art networks, where we will establish the differences between one-stage and two-stage networks.

Because YOLO was the network chosen for the development of this application, at the end of this part we will show a more extended description of the evolution that this NN has gone through, as well as other one-stage network that has appeared because of its popularity.

3.2.2 Evolution of classifiers

Neural networks are breaking through artificial vision techniques by giving notorious and efficient results to problems that computer vision have been trying to address and solve since the beginning of its creation.

Haar-feature classifiers, like the ones we see in the figure 3.1, were the first models used to extract the data analyzed and come up with a classification of certain patterns that confine the figure of a person's face or figure. These classifiers are applied to the image to detect changes in the contrast of an image that can be given by edges in different positions [39].

Models based on these classifier features find the best threshold to classify the desired object positively or negatively. After classifying the images, the method will have an error rate and after a training process where the selection of classifiers is changed and optimized, the application is meant to have a lower error.

The final classifier is a weighted sum of weak classifiers, which in their own they can't classify images, but together they form a strong classifier. The final setup has around 6000 classifiers features, and it is said that with 200 of them, a detection with 95 percentage of success rate can be obtained.

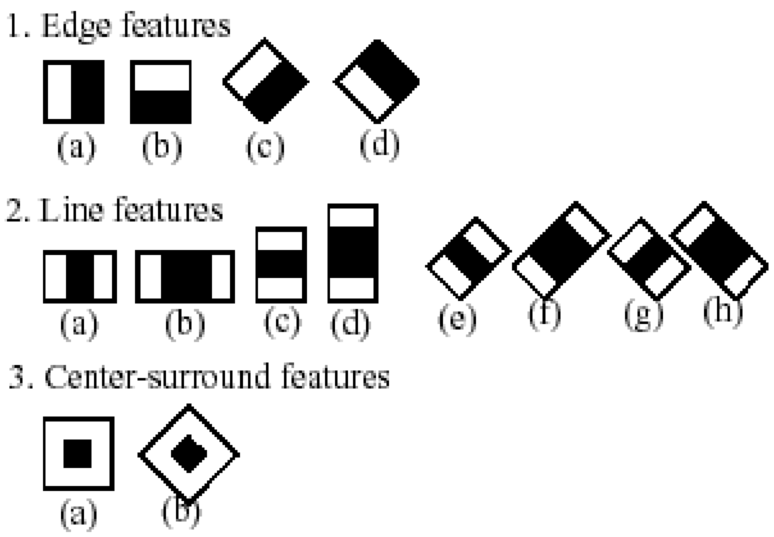


Figure 3.1: Example of simple Haar-feature classifiers

The cascade classifiers were the next step in this evolving process. We have a classifier that is trained with views from different perspectives of an object to have a positive confirmation of that object and a negative confirmation of the rest of the objects. When a classifier is applied across the image, obtaining a positive output would show that the region is likely to contain the object that it was trained to find. The cascade refers to the way this big classifier is made: Simpler classifiers that are applied subsequently and getting more complex in deeper levels in the region of interest give us the result of the object that we may have in the region of interest.An example of this process can be seen in the figure 3.2

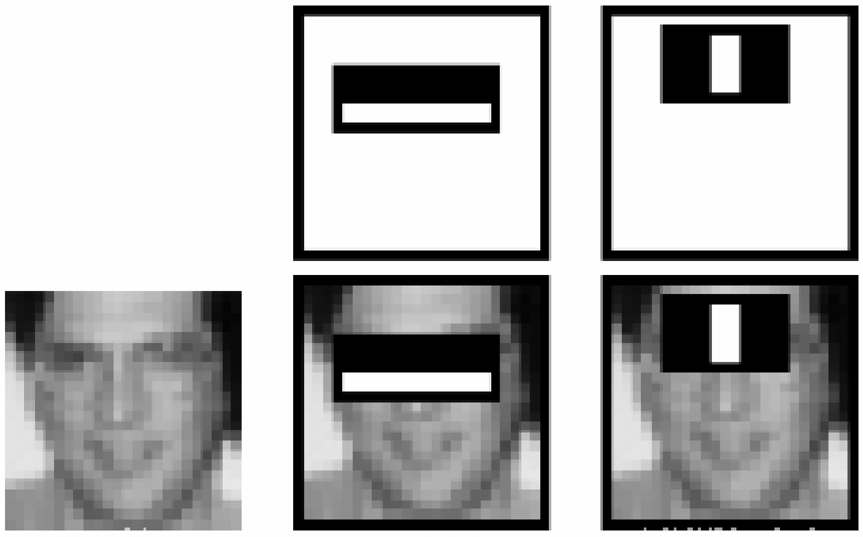


Figure 3.2: Example of how a simple linear feature is applied in a region of interest

Haar-feature classifiers have been chosen manually after trial and error testing, obtaining as results a number of classifiers that can detect certain objects in the region of interest. This process, eventually, can take a long time and can become tedious for creating a group of filters that can work to have positive results for a new object and, at the same time, a negative one for the rest of the objects.

In convolutional neural networks, we get over this problem by automatizing this process: The neural network determines by training which kernel have good results with the training subset of images that we provide. Because this is a process where the network learns to extract the information from the images, these algorithms are called deep learning algorithms. We can see an example of the typical structure in the figure 3.3

As we can see in the following Figure, these networks contain different types of convolutional layers and a final fully connected layer. Alexnet, which got the highest performance in the ImageNet Large Scale Visual Recognition Challenge on 2012, is formed by five convolutional layers and three fully connected layers.

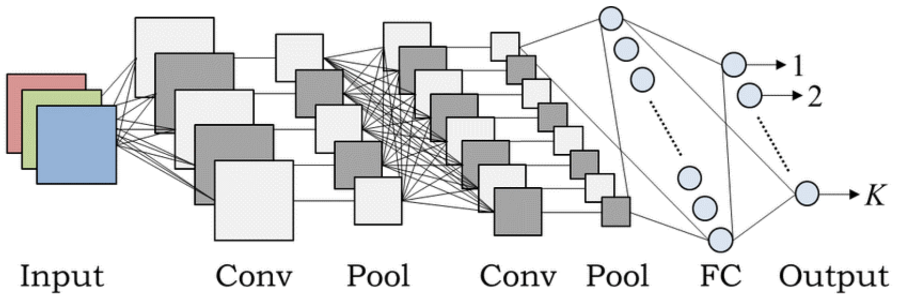


Figure 3.3: Example of the layer distribution in a convolutional neural network

During this process, the objective is not losing the information of the features while reducing the images into lighter and easier forms to process. In the convolution layer part, we can distinct two types of processes:

- Feature extraction process: The feature extraction from the images is done in the convolutional layers. The image is convolved with a kernel which has a small size compared to the original image. Different kernels are applied across the image and the output obtained is a resized image with the depth of the number of filters used. If a filter fits the values in a certain position on the image, the output image will show a positive value in that region. In every convolutional layer, we are probably using more than one filter, hence the depth of the output of this layer is given by the number of filters used.

  The size of the filters and its parameters (hyperparameters) are chosen, but the values change and evolve during the training to get better results with every iteration the network does. The method used in neural networks that can make us achieve better classifications is called back propagation.

- Rectification process: An attractive characteristic of convolutional neural networks is the incredible limited amount of time they need to classify an image. Studies show that mixing layers that add non-linearity to our network provokes reaching the training error rate faster and, depending on the method we use in this kind of layers, the training error rate can be even six times faster [40].

  The most effective and therefore the most used activation layer is, as Nair and Hinton names [41], the rectified linear unit (ReLu), which is defined with the following equation:

  $$f(x) = \max(0, x)$$

  Using these layers that add nonlinearities to convolutional layers outputs produces an improvement in the network's learning process.

- Dimension reduction process: Given the high number of pixels in the images, we need to somehow reduce the quantity of information while we also need to save the spatial information that the image contains.

This subsampling process is done with pooling and, depending on how the information is wanted to be handled, there are different types of pooling (max pooling or average pooling for example) that can be used.

- Classification process: After a certain number of layers where several features in the image have been extracted and when the size of the image has been reduced after passing the image throughout pooling layers, the last pieces of information are passed to what is called a fully connected layer. This layer takes the output from the last layer and determine which of these outputs shows an activation for the desired result at the end of the network. Depending on how the neurons are activated throughout all the network, different values are provided to the fully connected layer, and so this is translated into probabilities of the different classes that this layer provides as the final output.We can see an example of this process in the figure 3.4.

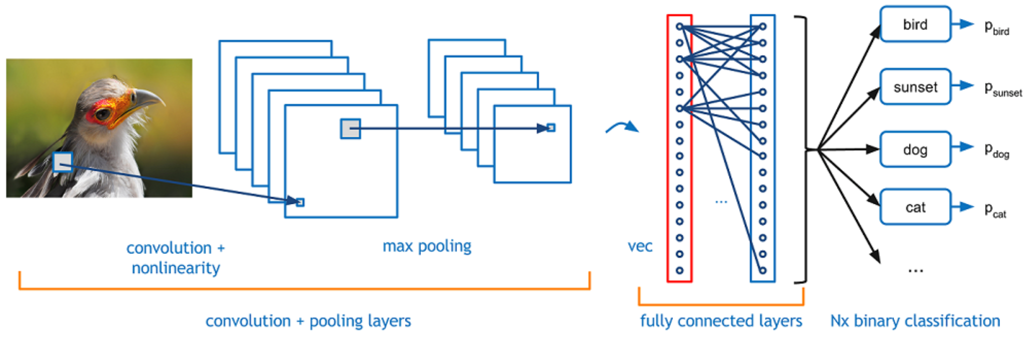


Figure 3.4: Neural networks have been widely tested to classify different animal species.

In the beginning, the results that the neural networks give in the output are not accurate enough and so, the network needs training. The computer adjusts the weights of its filters through a method called backpropagation.

Backpropagation aims to reduce the error function that the neuron has. This is given by the difference between the probabilities of the results compared to the actual class that the training image is tagged with. The network then knows which probabilities should be higher and which should be lower. Determining then the weights that contribute more for classifying the correct class and dropping the recognition rate of the other classes, the weight of the neurons is updated.

Finally, the error is decreasing through these iterations, and its decreasing speed is called the learning rate of the network. The objective therefore for having an efficient network is reducing the learning rate while achieving the best possible result in the loss function in the least time.

Nevertheless, these networks can only bear with images where the object to detect is clear and where any other object are obstructing its boundaries in the image. Convolutional Neural Networks (CNNs) have the capacity for object classification but falls for object detection. Unlike image classification, object detection probably may find more than one object in the image.

With a new approach, a group of researchers from the university of Berkeley added a module to perform object detection to the top neural network for image classification from those days. Running selective search [42], an algorithm that sets initial smart regions using the method of Felzenswalb and Hutterloncher [43] and gathers the regions together depending on its similarities, they achieve process by the use of the hierarchical algorithm with the support of the measures of colours and textures similarities as they explain. We can see a visual example of selective search in the figure 3.5.

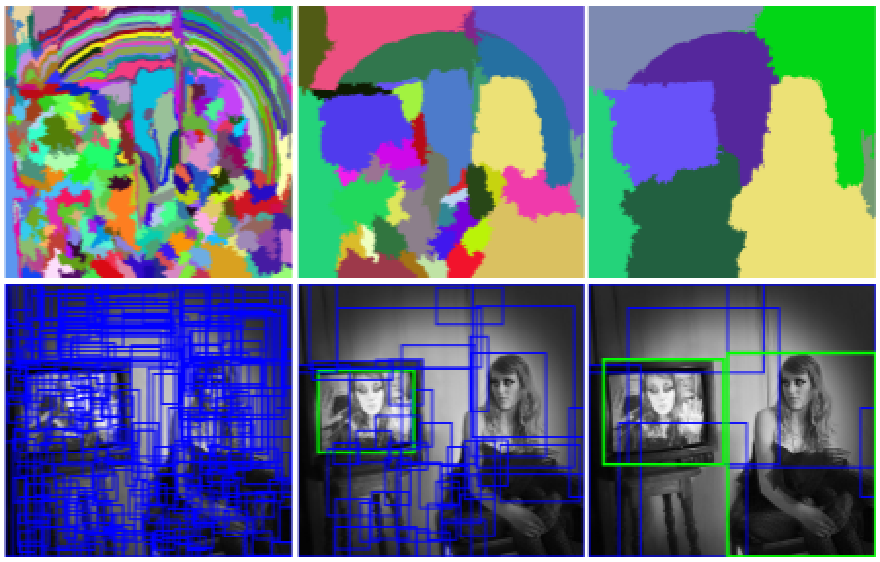


Figure 3.5: Detections after using selective search

Selective search can generate up to 2000 region proposals. These regions are going to be fed into the CNN in wrapped images of 227x277 pixels regardless of the size of the initial region, and then the R-CNN will extract its features with the convolutional network. We can see its model in the figure 3.6.

For each class that is available, the network extracts a vector using the SVM and scores it. With all the scores, a greedy non-maximum suppression strategy rejects a region if it has an intersection-over-union (IoU) overlapping with a larger region that has a higher score than the learned threshold.

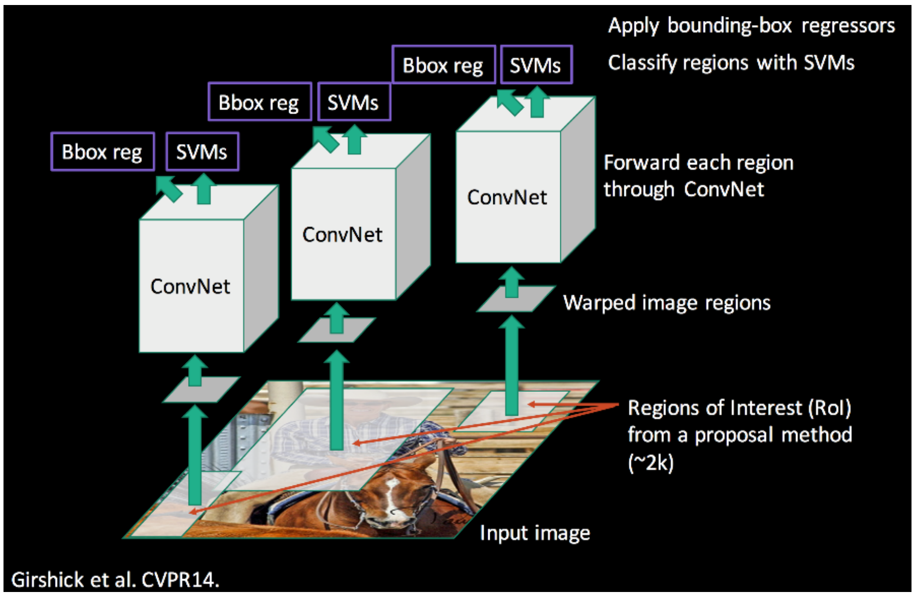
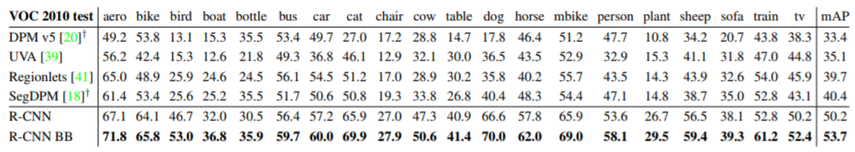

Apply bounding-box regressors
Classify regions with SVMs
Bbox reg
SVMs
Bbox reg
SVMs
Bbox reg
SVMs
ConvNet
ConvNet
ConvNet
Forward each region through ConvNet
Warped image regions
Regions of Interest (RoI) from a proposal method (~2k)
Input image
Girshick et al. CVPR14.

Figure 3.6: R-CNN model

R-CNN [44] achieved a mean average precision (MAP) of 53.3 %, being one of the most successful networks on 2014 and getting a place in the state of the art. We can see more in depth the results in the figure 3.7.



VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 [20]†	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [39]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [41]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [18]†	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	**71.8**	**65.8**	**53.0**	**36.8**	**35.9**	**59.7**	**60.0**	**69.9**	**27.9**	**50.6**	**41.4**	**70.0**	**62.0**	**69.0**	**58.1**	**29.5**	**59.4**	**39.3**	**61.2**	**52.4**	**53.7**

Figure 3.7: Mean average precision results achieved in the VOC 2010 test by different approaches.

Nevertheless, searching for 2000 regions made this network work fast by then, but incredibly slow at detection compared to the next networks that came soon. Thus, the training needed different fine-tunes (for the region proposal loss and the feature detection for the SVMs) and the time and space needed for training were aspects to be improved.

The year after the R-CNN paper was released, Ross Girshick updated the R-CNN and proposed a fast region-based convolutional network method for object detection instead of using selective search as the standard R-CNN was using [45]. This new approach would increase the performance at all levels while also improving the final scores for the detections. In addition, Fast R-CNN does not have different fine-tuning stages but a single-stage that uses multi-task loss and does not require disk storage, achieving the solution needed for the normal R-CNNs.

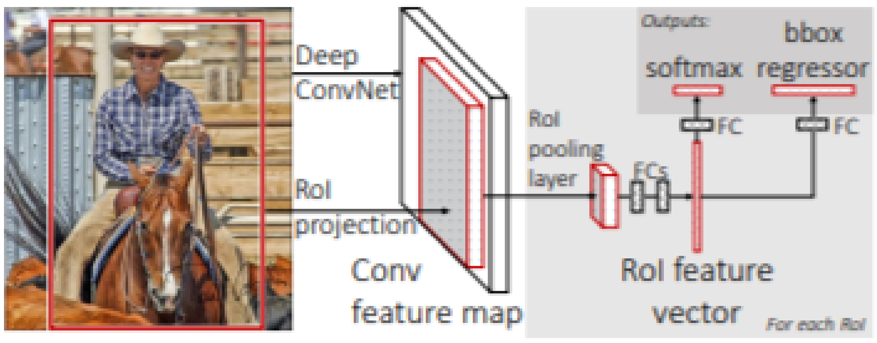
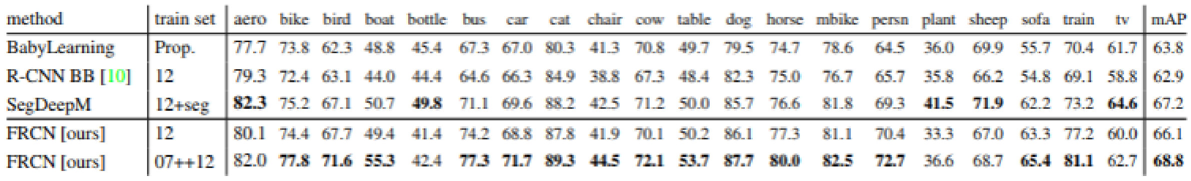


Figure 3.8: Fast R-CNN model

The architecture for object detection in Fast R-CNN processes the entire image with convolutional and max pooling layers to produce a convolutional feature map. For every possible object found, a region of interest (RoI) pooling layer extracts a fixed-length feature vector from the feature map, which will be passed through a fully connected layer that has two outputs: one that predicts the probability of the object classes and one that returns four real-valued numbers for each object found of that class. We can se its model in the figure 3.8, and some results of R-CNN in the figure 3.9.

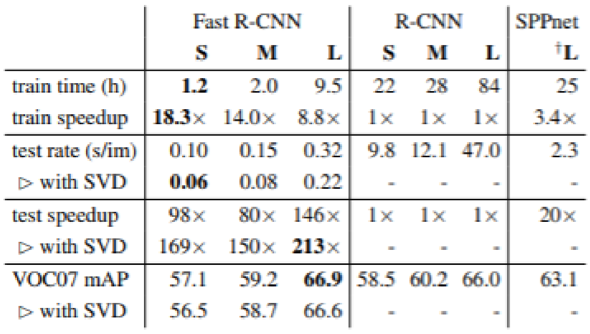


method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	**82.3**	75.2	67.1	50.7	**49.8**	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	**41.5**	**71.9**	62.2	73.2	**64.6**	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	**77.8**	**71.6**	**55.3**	42.4	**77.3**	**71.7**	**89.3**	**44.5**	**72.1**	**53.7**	**87.7**	**80.0**	**82.5**	**72.7**	36.6	68.7	**65.4**	**81.1**	62.7	**68.8**

Figure 3.9: Image train test VOC 2010, same as the previous one. It shows a post version of R-CNN too that achieved better results

Using this new method for object detection, Fast R-CNN processes images 146 times faster than R-CNN and training times are reduced up to 9 times, and with truncated singular value decomposition

(SVD) it can slightly improve a bit more. When it comes to classification, softmax classifier learns during fine-tuning instead of training between positives and negatives detections like R-CNN used to do with SVMs. Results are slightly better when FRCN tries when the first method, so it is proved that it leads to a better performance. We can see the improvement achieved in the figure 3.10.



	Fast R-CNN			R-CNN			SPPnet
	S	**M**	**L**	**S**	**M**	**L**	†**L**
train time (h)	**1.2**	2.0	9.5	22	28	84	25
train speedup	**18.3**×	14.0×	8.8×	1×	1×	1×	3.4×
test rate (s/im)	0.10	0.15	0.32	9.8	12.1	47.0	2.3
▷ with SVD	**0.06**	0.08	0.22	-	-	-	-
test speedup	98×	80×	146×	1×	1×	1×	20×
▷ with SVD	169×	150×	**213**×	-	-	-	-
VOC07 mAP	57.1	59.2	**66.9**	58.5	60.2	66.0	63.1
▷ with SVD	56.5	58.7	66.6	-	-	-	-

Figure 3.10: Results comparison between Fast R-CNN and its predecessor

As its report says [45], towards real-time object detection with region proposal networks, Ros Girshick along his team knew that, even though FRCN improved with the new detection network, it had a high computation cost that was still a drawback. Faster R-CNN [46] proposes a fully convolutional network that predicts object bounds and classification scores at each position, reducing computation cost as it was desired for FRCN.We can see its model in the figure 3.11.

Region Proposal Networks (RPNs) form the backbone of the Faster R-CNN. Convolutional feature maps used by region-based detectors are used for generating region proposals and by adding regress region bounds and object classification at each location on a regular grid, it is possible to create a fully convolutional network that can be trained on fine-tuning region proposals on one hand, and on the other hand, trained on object detection while keeping regions fixed.

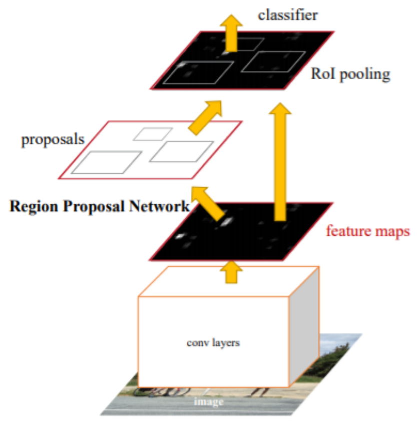


Figure 3.11: Faster R-CNN model

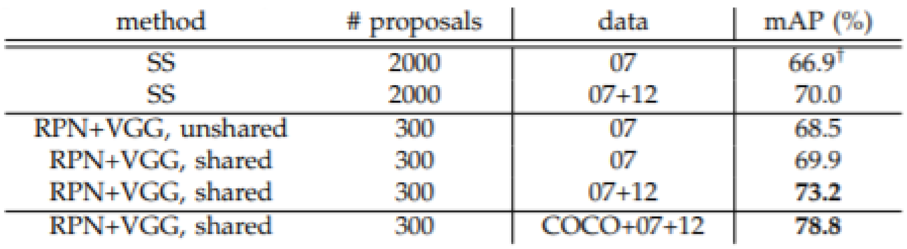
To predict regions, the RPN takes an image of any size as an input and return a set of rectangular object proposals that will tell if we have an object or not by giving an objectness score. The algorithm is based on a pyramid of k anchor boxes that slides across the window to find the best fitting box for a detected object, obtaining a more cost-efficient methodology. With each anchor, the network outputs the probability of having an object in its boundaries. Then the fast R-CNN detector use de convolutional features computed on a single-scaled image. With the pyramid of anchors, features can be shared between anchors without extra costs. To train the RPN, methods like stochastic gradient descent (SGD) or back propagation are commonly used.

The second module of the Faster R-CNN is the FRCN, used for classification purposes. To establish a bridge between both modules, given that both networks are trained separately, and their convolutional layer will be modified in different ways, creators have come up with different training patterns:

– An alternating training, that could be seen like a unique network training where the RPN is trained first and then, the proposals are used to train the FRCN and back again. The network tuned by the FRCN is used to initialize RPN in an iterative way.

– A second method would be approximate joint training, where they actually work like one network given that both errors are tuned at the same time.

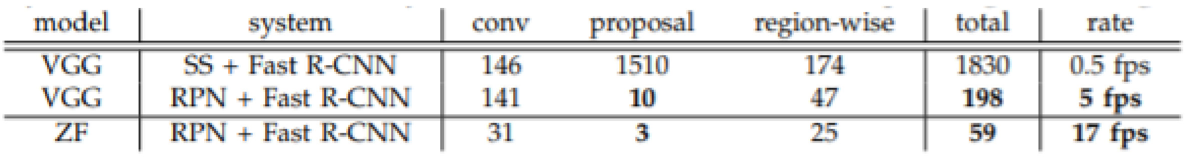
Finally, non-maximum-suppression (NMS) is adopted in the RPN to reduce the redundancy of the proposals estimated. We can compare the number of proposal depending on the approach in the figure 3.12.



method	# proposals	data	mAP (%)
SS	2000	07	$66.9^{\dagger}$
SS	2000	07+12	70.0
RPN+VGG, unshared	300	07	68.5
RPN+VGG, shared	300	07	69.9
RPN+VGG, shared	300	07+12	**73.2**
RPN+VGG, shared	300	COCO+07+12	**78.8**

Figure 3.12: Number of proposals using different detection approaches and mean average precision achieved.

This method does not harm the accuracy and reduces the number of proposals. Timing and mean average precision are greatly improved with the RPN, decreasing the rate of detection and the total accuracy.We can compare the frame rate using different CNN in the figure 3.13.



model	system	conv	proposal	region-wise	total	rate
VGG	SS + Fast R-CNN	146	1510	174	1830	0.5 fps
VGG	RPN + Fast R-CNN	141	**10**	47	**198**	**5 fps**
ZF	RPN + Fast R-CNN	31	**3**	25	**59**	**17 fps**

Figure 3.13: Frame rate results using different CNN architectures.

When the neural networks contain a region proposal network and therefore, the detector is formed by two networks, we say it is a region proposal-based network. On the other hand, one year later, different networks that implemented the whole system in only one-stage network.

First, in May 2016 YOLO (You Only Look Once [47] was presented by Joseph Redmon, proposing a solution planned as a regression problem, dividing the image into a grid from which will it return a certain amount of bounding boxes and associated class probabilities in only one evaluation, allowing fine-tuning and optimizing directly on detection performance. We can see its model in the figure 3.14.

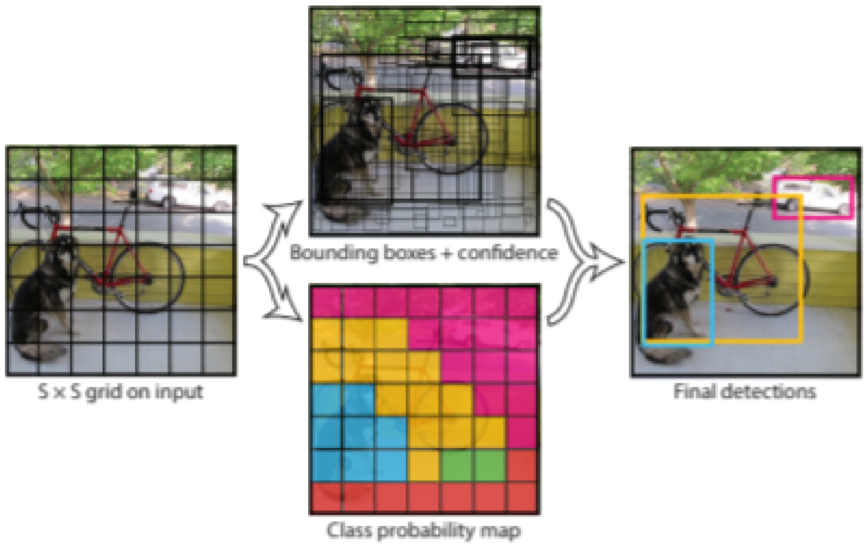


Figure 3.14: YOLO process scheme

Each grid of YOLO proposes potential bounding boxes and can share some similarities with the boxing method used by the R-CNN depending on the regions obtained with the boundaries of their regions, but the difference between the R-CNN methods is that only 98 bounding boxes are proposed per image instead of the 2000 that could be created from selective search.

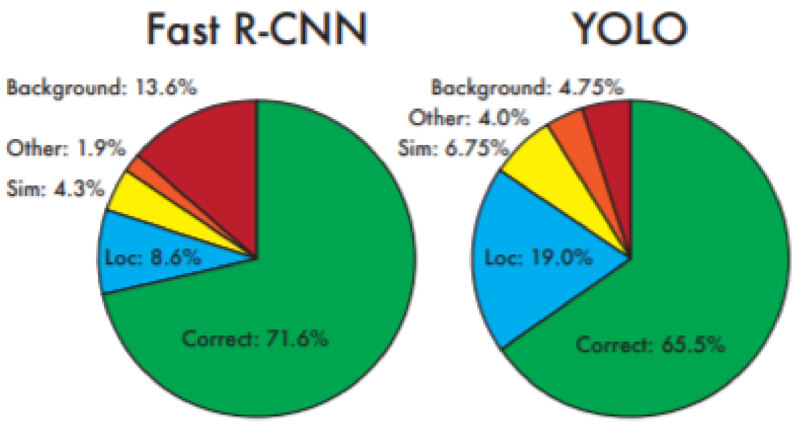


Figure 3.15: YOLO and Fast R-CNN mean average precision comparison

YOLO achieved a good performance than enables its system to run even on real time, reaching detections up to 45 frames per second. Still, having this rate has its drawback, where the mean average precision as we see in the results falls compared to previous methodologies, sacrificing accuracy for computational performance. We can compare its mAP in the figure 3.15.

Finally, in December 2016 one last competitor in the object detection and classification field came out with the name of Single Shot MultiBox Detector (SSD), using again a one-stage network architecture to carry out the whole detection [48].

SSD informs that using a network which does not resample pixels or features for bounding box hypothesis, enables them to maintain a good frame rate while having good results in high-accuracy detections). The improvements brought by the Single Shot Detectors are using small convolutional filters to predict object categories and offsets in bounding box locations, using the separate predictors for different sizes in later stages of the network where the feature maps have already a solid detection.

Compared to the Faster R-CNN, SSD gets rid of the region proposal network by applying a multi-scale feature and default boxes. This is possible to do by using lower resolution images that makes the process lighter and therefore, computationally faster.

The SSD is composed by 2 blocks: The first block consist in a feature map extractor that uses VGG-16 (Visual Geometry Group for object recognition, formed by a mix of 16 convolutional and pooling layers) which creates four region proposals (having these regions the same centroid but different bounding boxes). The SSD uses a multi-scale feature map, as we see in the figure 3.16, that reduces the spatial dimension gradually, changing the resolution of the feature map and the size of the objects detected in it.

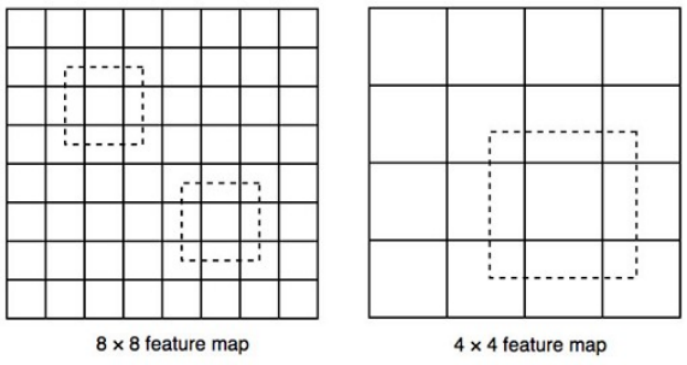


Figure 3.16: Multi-scale map feature introduced in SSD

With each spatial prediction, the second part will be used for object detection, and it is formed by a group of feature layers that use a cascade of small convolutional filters that calculate the score probability for all 21 classes, returning only the class with the highest score above a certain threshold. We can see a comparison of its architectures in the figure 3.17.

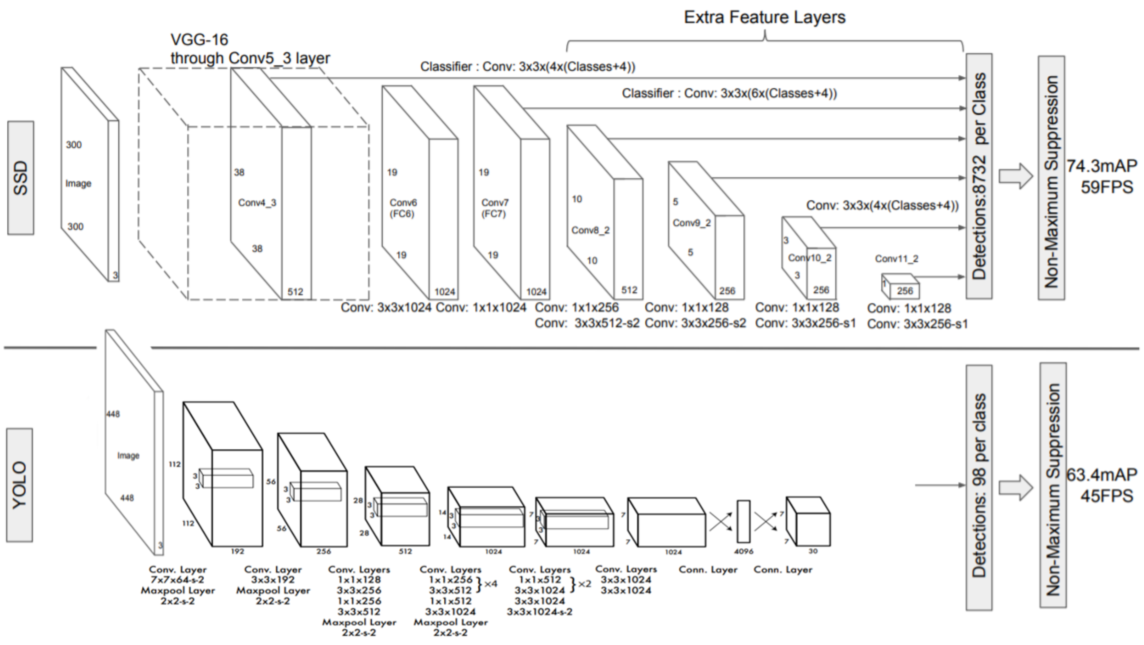

SSD
VGG-16
through Conv5_3 layer
Extra Feature Layers
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
Conv: 3x3x(4x(Classes+4))
Image
Conv4_3
Conv6 (FC6)
Conv7 (FC7)
Conv8_2
Conv9_2
Conv10_2
Conv11_2
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256
Conv: 3x3x512-s2
Conv: 1x1x128
Conv: 3x3x256-s2
Conv: 1x1x128
Conv: 3x3x256-s1
Conv: 1x1x128
Conv: 3x3x256-s1
Detections:8732 per Class
Non-Maximum Suppression
74.3mAP
59FPS
YOLO
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer
Detections: 98 per class
Non-Maximum Suppression
63.4mAP
45FPS

Figure 3.17: YOLO's and SSD's architectures comparison

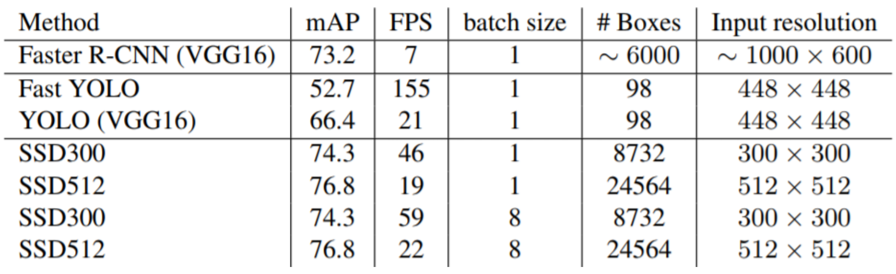
On the other hand, compared to YOLO, the only other network that works with bounding boxes and predictions as a one-stage network by then, we can appreciate these differences:

– Different model for predicting detection in each feature layer. The differences in size through different layers are easy to see, having an improvement in the final mAP (mean average percentage) results.

– SSD uses convolutional predictors for the detections, having them in the top of the architecture. While YOLO uses an intermediate fully connected layer, SSD prefers to relay on the convolutional layers to calculate the score probabilities.

For training, the SSD framework needs an input image with the ground truth boxes for each box for each object.

As seen above, the image is evaluated with different aspect ratios at each location and so it is matched with the ground truth boxes, and so, the model loss is a weighted sum between location loss [44] and confidence loss. With these changes, it can be highlighted the results obtained with VOC2007 test and comparing their 74.3% of detections with 59 FPS against the 73.2% Map of Faster R-CNN with 45 FPS or 63.4% Map of YOLO with 45 FPS). We can see a comparison between different CNNs in the figure 3.18.



Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Figure 3.18: Result comparison between the different state-of-the-art detection frameworks.

To design our application, instead of using a two-stage faster R-CNN where the detector first uses the Region Proposal Network (RPN) and then apply the classifier to the candidates it has found, we have decided to go for single-stage classifiers as YOLO: they have a good balance between frame rate and mean average precision (mAP).

3.2.3 Choosing YOLO

3.2.3.1 Introduction

In the beginning of this project it was clear that to reach the final objectives, we needed deep learning techniques to perform detection and classification tasks. The idea of using a neural network was interesting after checking that using traditional methods in Matlab for these purposes could take a longer time to build from scratch and some solutions were already presented.

In this chapter, there will be a short introduction about the features that the new versions of YOLO have, and we will discuss which has been chosen for the final application after studying their characteristic and testing them with real footage.

During the development of this project, the version that have been tested are the robust YOLO v3 version and lighter versions of YOLO such as Dark flow and Tiny-YOLO. Even though it has not been tested in this project, we will introduce the changes that YOLOv2 brought so it will be easier to understand what YOLOv3 includes and the changes compared to its predecessors.

The objective is clear: detect the objects that are needed, such as suitcase and backpacks and then, select the better version that will have the best trade-off between speed and accuracy.

3.2.3.2 YOLOv2

Since the first version of YOLO, two major version have been released headed by Ali Farhadi and Joseph Redmon. In December 2016, close to the date where SSD broke into the ecosystem of object detectors and classifiers, the second version of YOLO known as YOLO9000 [49] brought some improvements compared to its predecessor. We can see an image we got running our test in the figure 3.19.

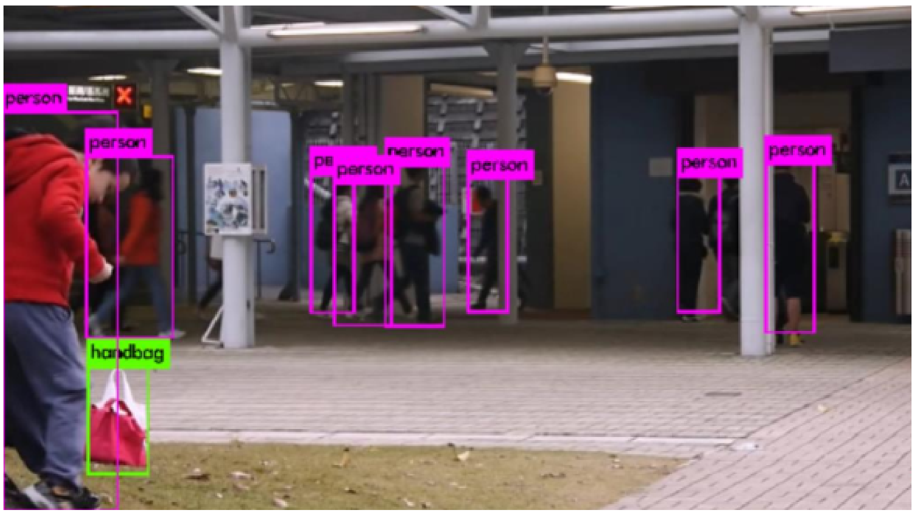


Figure 3.19: Test image obtained with YOLOv2.

As it was already addressed on the first paper, YOLOv1 had a 63.4 mAP that compared to other state-of-the-art classifiers, like the Fast R-CNN with 70% or the faster R-CNN with 73.2%, it fell short in accuracy. SSD on the other hand, solved the problem by using different methods in the final layers to improve the classification problem.

The first step for YOLOv2 was to improve this issue. Using different techniques, explained below, the creators manage to improve the overall accuracy:

- Using Batch normalization [50], which improves overall stability and reduces overfitting at the same time.
- Upgrading to higher resolution classifiers, where now YOLOv2 accepts image with 448x448 compared to the previous 224x224 resolution, allowing the network to adjust better in the filtering process. The group of filters at the beginning of the network will take care of resize the image so it can fit the network needs. This is shown in the figure 3.20.

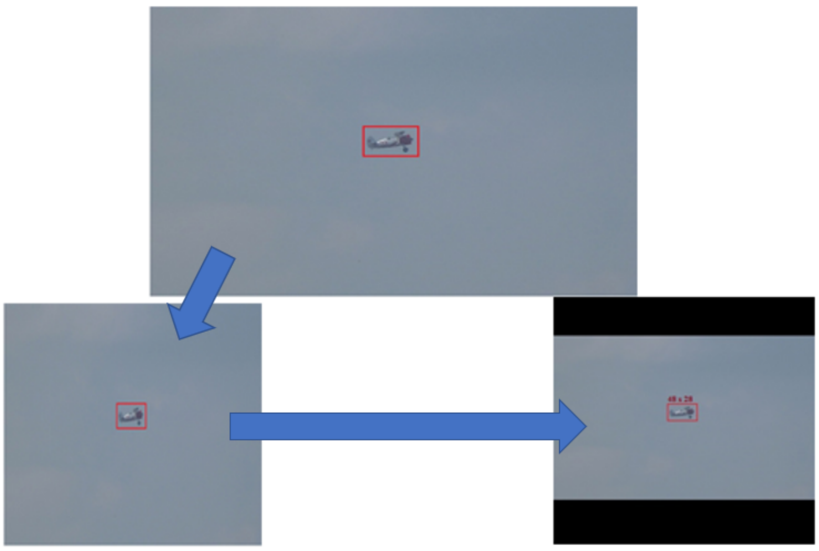


Figure 3.20: YOLOv2 introducing a function to resize images to a specific size in order to handle better different input sizes.

- Switching to anchor boxes, which is the system that was already used by Faster R-CNN or SSD and ditching the fully connected layers that YOLO used to have. The network shrinks to operate on 416 input images so a feature map of 13 x 13 can be obtained at the end with the filter the network uses. Where YOLO predicted objectness and class probability based on its grid, now YOLOv2 does it based on the anchor boxes.
- Adding Dimensions Clusters, which is the result of running k-means clustering on the training set bounding boxes to automatically find good priors instead of choosing priors by hand, as other networks do. This allows to use a weak supervision when it comes to training the network.
- Location referenced to the grid cells, just like it was already done in YOLOv1 after checking first that using a similar model based on RPN, results would tend to a system instability.
- Adding multi-scale training and enabling YOLOv2 to work with different input sizes (from 320 to 608). Thanks to a new classification network called Darknet-19(based on the number of convolutional and Maxpooling layers that stack inside the network), YOLOv2 obtained

a detection improvement, reaching a 78.6% compared to the 63.4% mAP of its predecessor. Despite using VGG-16 with other networks, it is considered too complex and too heavy to be used along YOLO.

– Using the WordTree, a hierarchical model for classification created for the training stage, the performance of the network improves, reaching a higher level of precision which implies that when an object reaches a high confidence value, the network can discern between races of the same species for example. Thanks to this methodology of combining datasets, YOLOv2 can improve the labelling of the objects without losing confidence on the real score probability of the actual object detected. We can see the model of this architecture the figure 3.21.

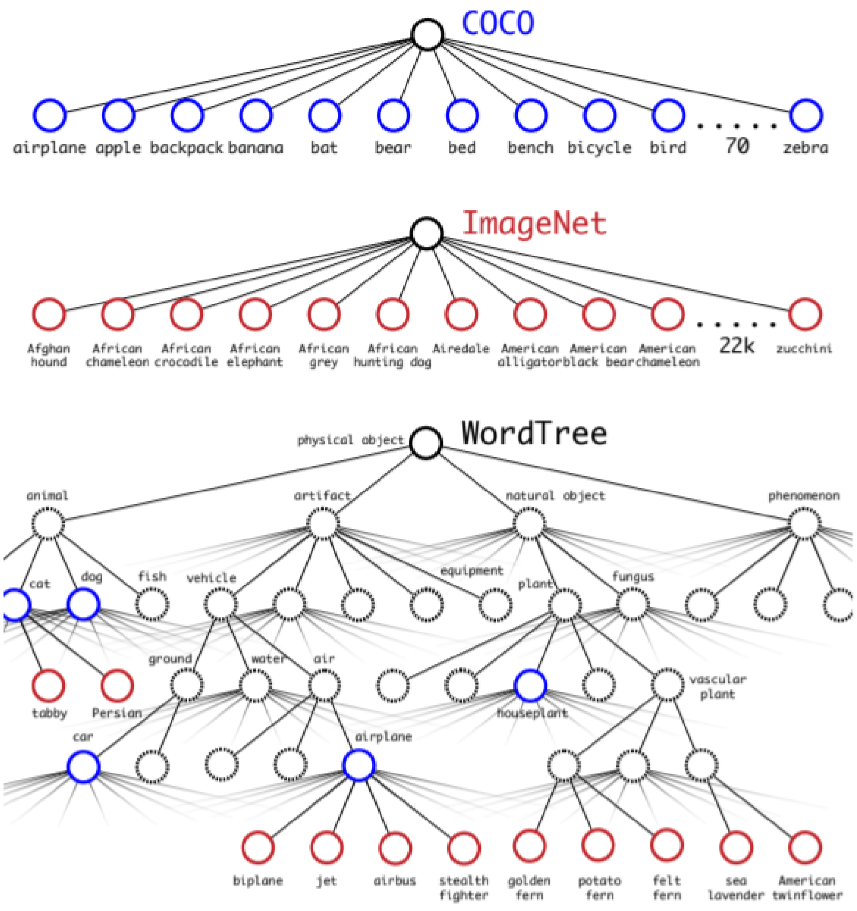
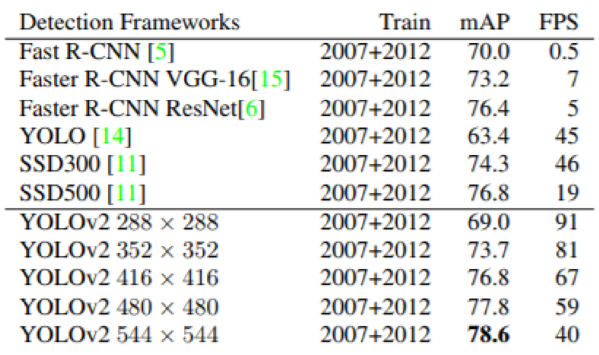


Figure 3.21: WordTree dataset model.

After implementing all these improvements, most of them brought by the other existing object classifiers such as the Faster R-CNN or the SSD, YOLOv2 becomes once more the state-of-the-art after achieving a 78.6 % Map and a frame rate that enables live classification. We can see a comparison of different frameworks in the figure 3.22.



Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288×288	2007+2012	69.0	91
YOLOv2 352×352	2007+2012	73.7	81
YOLOv2 416×416	2007+2012	76.8	67
YOLOv2 480×480	2007+2012	77.8	59
YOLOv2 544×544	2007+2012	**78.6**	40

Figure 3.22: Mean average precision achieved by different frameworks.

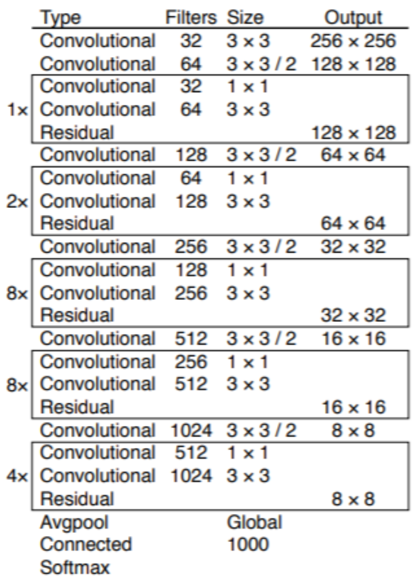
Nevertheless, despite having an overall improvement in classification and detection, YOLOv2 still drags some shortcomings that YOLOv1 had, and it is the learning rate of clothing and equipment and therefore, their detection. This version definitively, would be hard to use in the experiments that this project presents and for that, it will be taken in consideration the changes brought by the next version and will check if at least theoretically, can bear the challenges this project has.

3.2.3.3 YOLOv3

Compared to the big changes that were related between YOLO and YOLOv2, YOLOv3 [15] does not rework the whole formula but introduces some subtle changes to improve the detection of the network by giving a step back in the frame rate.

Bounding boxes for example remain intact and they follow the same principle of YOLOv2 by giving the prediction of the four corners of the objects. Multilabel classification is still used to help the network to detect objects of different sizes, specially object of small sizes. K-means clustering keeps creating 9 clusters which are divided evenly across scales in the training process. We can see YOLOv3's architecture in the figure 3.23.

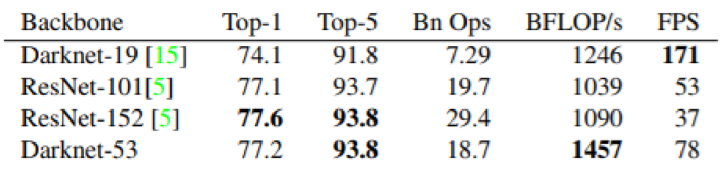
But this time, instead of using Darknet-19, which is the part of the network dedicated to classification, YOLOv3 uses Darknet-53, composed as the name says by 53 convolutional layers which uses a repetitive pattern of filters with 1x1 and 3x3 size, followed finally by a residual layer.



	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.23: YOLOv3 architecture.

The composition of this backbone for the classification task, compared to ResNet (which is the classification network adopted by RetinaNet, another state-of-the-art network used for detection and classification [51], shows that YOLOv3 achieves a similar accuracy in the TOP-5 results while proving higher efficiency and computational performance compared to their competitors, reaching double frame rate than ResNet. We can see a comparison of backbones in the figure 3.24.



Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

Figure 3.24: Comparison of backbones.

YOLOv3 also explain different ideas that were tried but that in the end, were not good for the network. These attempts were trying different ideas to improve the values created for the bounding boxes by using offset predictions for width or height, or linear prediction instead of logistic, which also leads to a decreasing value in the map. Another experiment was using focal loss, a different loss function that had good results for RetinaNet. Nevertheless, it did not work as intended for YOLOv3 because again, this approach drops the final mAP of the network. We can see an image we got running our test in the figure 3.25.

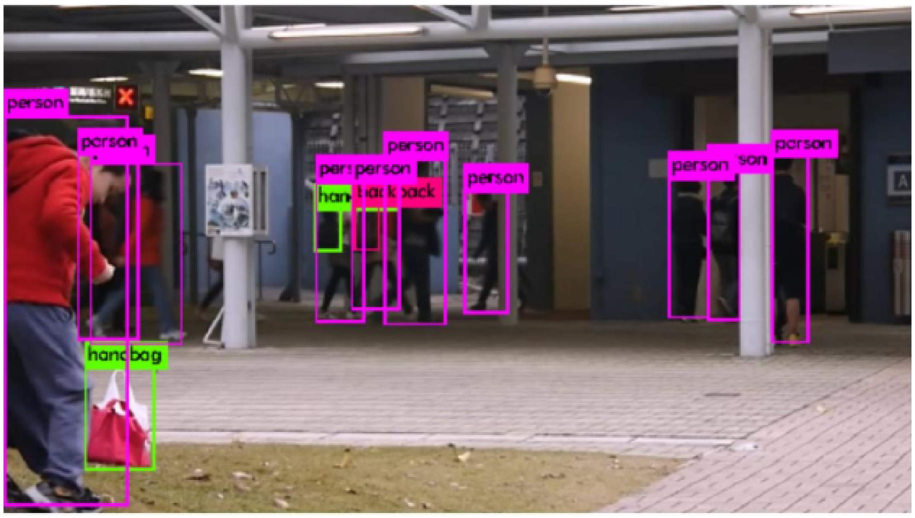


Figure 3.25: Test image obtained with YOLOv3.

As we can see, even though the overall accuracy has been improved, we can have already some doubts from this paper of YOLO's capability to detect correctly backpacks, so we will have to check with our database if this neural network will give us good results.

3.2.3.4 YOLO light versions

While YOLOv3 aims to improve its accuracy by dropping off the good frame rate it could achieve with a newer version, other side-versions of YOLO which are already satisfied with the actual accuracy, aim for getting an even better speed when it comes to detection and classification. The name given to these versions are based on the classifier backbone that the state-of-the-art network studied uses.

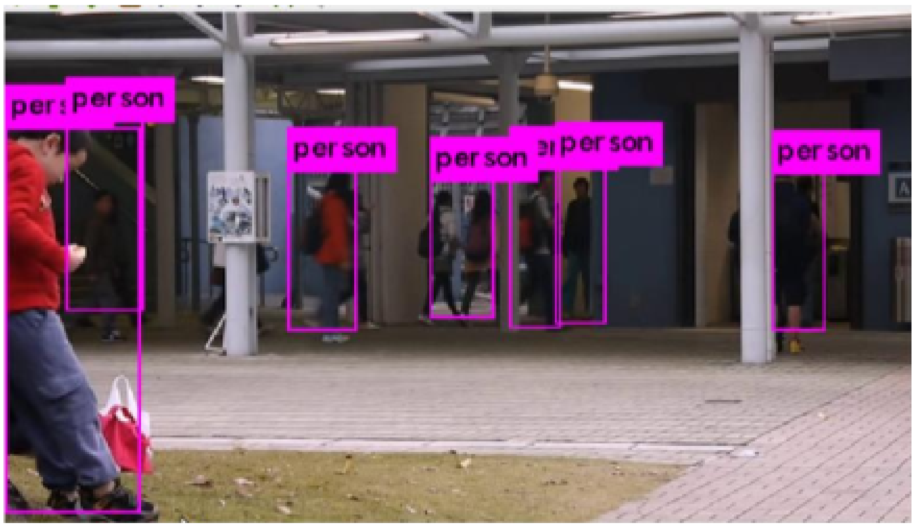


Figure 3.26: Test image obtained with Darkflow.

3.2.3.4.1 Tiny Darknet

Tiny YOLO born as an answer to tiny neural networks as SqueezeNet [52], which its size is only of 4 MB, what allows this network to be used even in embedded systems. Based on Darknet-19, the one used by YOLOv2, Darknet only drops 1.3 points in the mAP in the TOP-5 list of accuracy while considerably reducing the weight of the network. A later configuration was created too for the newest YOLO version, the Tiny version of YOLOv3 has a bigger difference compared to YOLOv3, having a bit more than the half of the accuracy than the original network.

This configuration has weight files much lighter as it is pointed, and the results obtained with this method will be shown in the experimentation part later.

3.2.3.4.2 Darkflow

Tensorflow is an end-to-end open source platform for machine learning [53]. When Tensorflow is used with APIs like Keras, it can easily prepare neural networks to test architectures as a quick prototype. Darkflow is the attempt of re-building the YOLO structure using this model. YOLO was originally written in C, while Darkflow is an exportation of YOLO using Tensorflow as its main platform, which was developed in Python. Even though YOLOv3 already counts with a wrap in Python, it is a different approach chosen by the creator of Darkflow [54].

In the repository, there is no information about the performance. We can see an image we got running our test in the figure 3.26.

Recently, an experiment has shown that Darkflow has a significant worse processing time, being 408 second compared to the 148 seconds of the original Darknet and having as a result, 10.5 fps compared to the 29.1 fps in the original [55].

3.2.3.5 Conclusions

The needs for automated surveillance are to have systems that can perform their tasks in real time while they maintain a good performance. YOLOv3 sacrifices frame rate for performance, and as we will see in the first testing steps with the dataset from Avenue, this version can detect better than the other the backpacks among the crowd. YOLOv2, even if it faster than the last version, falls in accuracy and it does not detect the same amount of backpack as YOLOv3 does.

Lighter versions of YOLO have shown that they are not prepared for these tasks, but their fast speed can maybe make them suitable for less demanding purposes. Still, it can be seen how better versions of YOLO can distinguish between two people that are close, when the lighter versions at some point group them as one.

3.2.4 Kalman Filter

3.2.4.1 Introduction

During these experiments, it is observable that sometimes, YOLO struggles to detect small objects. This generates in objects that disappear through a certain frame and reappear after that. Therefore, we need a mechanism to estimate the possible position of the object to facilitate the re-identification and correct tracking of the objects.

Kalman filter has been chosen for being a simple but an effective algorithm that can predict the position of an object with a good performance, and its results will be displayed later in this memory.

3.2.4.2 The recursive algorithm

Control theory is based in the existence of mathematical model that represents the expected behavior of the systems to control. Nevertheless, any mathematical model is exact due to these reasons:

- There are some effects that are hard to model and are only approximations.
- Systems are affected by noise and not all of them can be considered.
- Sensors do not always transmit a valid information hence; outputs do not always correlate completely with the real model.

Kalman [56] is a digital filter that recursively minimizes the estimated error of its predictions thanks to probabilistic techniques done on the sensing system and the noise related to it.

Estimated probabilistic methodologies are based on the knowledge that is acquired beforehand about the system, in other words, how the model works. The Kalman filter can estimate the evolution of the system and compare the prediction to the real value sensed as it goes repetitively predicting the next state, so that is why it is a recursive method and why it has less error that other probabilistic methodologies.

To improve the estimation performance, this algorithm considers following information of the system:

– Mathematical discrete model of the system to study, already affected by the noise included in the detection (vk) and model of states (wk). We can see state and measurement functions in the figure 3.27.

$$x_k = f(x_{k-1}, u_k, w_k)$$
$$z_k = h(x_k, v_k)$$

Figure 3.27: State and measurement function.

– The random variables wk and vk represent the process and measurement noise error respectively. They are assumed to be independent (of each other) with normal probability distributions. We can see the figure 3.28 refered to the process and measurement noise functions below.

$$p(w) \approx N(\mu_w, Q)$$
$$p(v) \approx N(\mu_v, R)$$

Figure 3.28: Process and measurement noise functions.

In practice, the process noise covariance Q and measurement noise covariance R matrices might change with each step or measurement but for simplicity, we will keep them constant.

With this information, the estimation is obtained through a process of prediction and posterior correction that aims to minimize the estimation error, obtaining the optimum estimation included in the Kalman filter.

The noise that affects all the measurements ($\text{Mw} = 0$ and $\text{Mv} = 0$) is eliminated by simple calibration techniques to ease the steps in the performance of the estimator. As it can be seen, the movement of the people in the videos present linear trajectories, therefore Kalman is a good way to estimate the position in future frames.

Building up the equations, we can describe the state matrix, which represents the state of the variables we are interested in and the measurement matrix as we see in the figure 3.29.

$$\dot{x}(t) = A \cdot x(t) + B \cdot u(t) + w(t)$$
$$z(t) = C \cdot x(t) + v(t) = x(t) + v(t)$$

Figure 3.29: Next state update and measurement matrix.

The n x n matrix A relates to the state at the previous time step to the state X at the current step. The n x l matrix B relates to the optional control input to the current state X. The m x n matrix H in the measurement equation relates the current state X to the measurement Z at the current step.

Kalman filter is based on error probabilities to determine an estimated state. This is where we define our a priori state estimate at the first step and then it is how we obtain the a posteriori estimate error given a measure Z.

The equation that computes the a posteriori state estimate X as a linear combination of the a priori estimate X and a weighted difference between an actual measurement and a measurement prediction X, as we see in the figure 3.30.

$$\hat{x}_k = \hat{x}_k^- + K(z_k - H\hat{x}_k^-)$$

Figure 3.30: Next state prediction matrix.

In case that the value of the prediction agrees with the measurement, the Kalman filter would work as intended and any correction would be needed for the next step. If not, a little adaption will be done in the state matrix. The n x m matrix K is chosen to be the gain that minimizes the a posteriori error covariance.

$$\begin{aligned} K_k &= P_k^- H^T (H P_k^- H^T + R)^{-1} \\ &= \frac{P_k^- H^T}{H P_k^- H^T + R} \end{aligned}$$

Figure 3.31: Kalman gain matrix.

As the measurement error covariance R approaches to 0, the gain K will make the different between the measurement and the previous state have more impact in the current state. Otherwise, if the estimate error covariance P approaches to 0, the previous state will have a bigger impact on the current state matrix. We can see the Kalman gain function in the figure 3.31.

The estimated process is done by using a feedback control: The filter estimates the process state and receives the measurements as a feedback. This can be divided in a group of time update equations and a group of measurement equations. The time update equations will move forward time the state of our process and the error covariance estimate the prediction for the next time step, while the measurement equations will make our system work in a range of values close to the real state of our model and therefore, they will prepare the information to reveal new a priori estimate to generate a new posteriori estimate reliable for the model.

The final equations for the time update can be seen as prediction processes, and they can be seen in the figure 3.32.

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$

$$P_k^- = AP_{k-1}A^T + Q$$

Figure 3.32: Predict phase state

The final equations for the correction step will be seen in the figure 3.33.

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1}$$

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

$$P_k = (I - K_k H)P_k^-$$

Figure 3.33: Correct phase state

In the actual implementation, the measurement covariance R is usually measured prior to operation of the filter. It is possible to obtain the typical error values in first experiments in order to determine the variance of the noise measurement.

To obtain the process noise covariance Q is a harder task to achieve due to the inability to completely observe the process that it is been estimated. A relatively simple process model can propose acceptable results with some uncertainty in Q.

By fine-tuning these parameters, a better performance can be achieved in the predictions from the filter. Representing these two stages finally, the processes are represented in the figure 3.34.

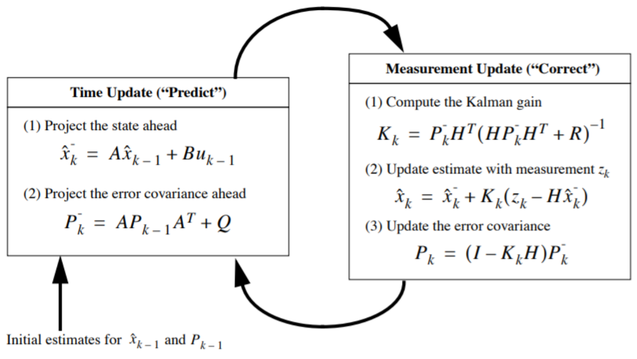

Time Update ("Predict")
(1) Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$
(2) Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update ("Correct")
(1) Compute the Kalman gain
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
(2) Update estimate with measurement z_k
$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
(3) Update the error covariance
$P_k = (I - K_k H)P_k^-$
Initial estimates for $\hat{x}_{k-1}$ and P_{k-1}

Figure 3.34: Correct and predict phases loop.

3.2.4.3 The model defined in this framework

Due to the problems that YOLO has to detect backpacks under some circumstances, we needed a solution to reidentify the bags that the people were carrying with them and would show up in the detection after some frames. For people it does not happen as much as it happens for backpacks but still, we have added an option to use kalman filters with people too.

The first solution was to pin the backpack to the centre of their respective owners when they were not found by YOLO. Nevertheless, this is not a good solution because the error between the possible position of the backpack and the centroid of the person, sometimes, is quite different. Not all the bags are carried on the back and therefore, for those cases a big distance could be the difference between doing a reidentification or failing at it.

Using Kalman filters, it has been proved that the estimation of the backpack when YOLO does not find it or when there is occlusion is a much better solution. These are the basic ideas that have been considered to develop the filter for each backpack:

- The filter is always following the loop predict-correct in each frame. The a priori information of the state will be the first measure done to have an initial point to start.
- The state considers the position in the axes of x and y as well as the velocity in both directions.
- When the backpack is detected, its position and velocity will be measured the information, proceeding with the predicting and updating iterations.
- If the backpack is not detected, it will use the position of the previous prediction as a measure, and it will use the velocity of its owner (if it is linked to one) to predict the next state. If YOLO does not see the backpack but sees the person (highly probable), then this will give a good estimation of its position giving that they are going to have a similar velocity vector.

Finally, measurements have determined that distances between pixels from position predictions are not too distant from actual detections, and therefore these are the values that have been fine-tuned to obtain the results for these experiments.

Chapter 4

Implementation and Results

4.1 Introduction

In the following sections, we will describe how we distribute the code that implements the different applications under this framework to perform their objectives. The code essentially is divided in three parts, which are three modules that conforms the application. Two of them are based on YOLO and the other one is based on optical flow density.

4.2 Lost object detector

The proposed solution for the Lost object detector is based on YOLO. This module can be divided in three steps as it is shown in the next figure. The first step is the detection and classification of people and backpacks in the image. Then, the second step is doing the identification of those people and backpacks and the update of the respective objects in the program that represent them. Finally, the third step is the update of their states after the linking process, where it can be told whether the backpack has an owner or whether it is abandoned.

It was the aim of this project to try a new approach in this field. Even though background subtraction seems to work fine as it has been reviewed, it may does not work in certain occasions where the person that is abandoning the bag is surrounded by more people, and that is the case in most of the sequences recorded for this project. Also, background subtraction techniques cannot tell specifically what kind of object is being abandoned, and it can be another important feature to be marked in favor of our approach.

Also, thanks to the advances in the deep neural network technology, this is an approach that could be performed in real time with some fixes in the future.

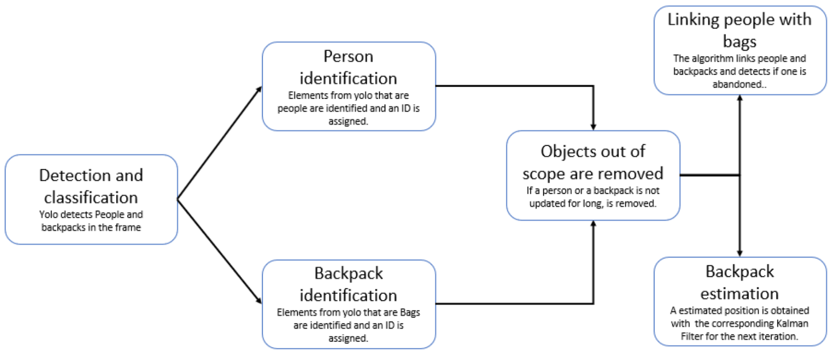

Detection and classification
Yolo detects People and backpacks in the frame
Person identification
Elements from yolo that are people are identified and an ID is assigned.
Backpack identification
Elements from yolo that are Bags are identified and an ID is assigned.
Objects out of scope are removed
If a person or a backpack is not updated for long, is removed.
Linking people with bags
The algorithm links people and backpacks and detects if one is abandoned..
Backpack estimation
A estimated position is obtained with the corresponding Kalman Filter for the next iteration.

Figure 4.1: Detection and classification flow chart

4.2.0.1 Detection and classification

The detection and classification are performed by YOLO. The function used in this case is called Detection. It needs as inputs the configuration files and weight files from YOLO, which are loaded thanks to the python wrap that was already prepared for this version. The classes used are from the COCO dataset [57], and the file that contains the names of these classes is loaded too.

An array with the objects found in the image as well as their position will be given as an output from this function.

4.2.0.2 Identification

In this process, the creation and update of the objects from the People and Backpack classes are done. The identification function gets empty arrays of objects from the classes Kalman, People and Bags. We also tried to improve the identification process by adding histogram comparisons, but after obtaining bad results in the test, we decided to not use it until we achieve an improving in its accuracy for the reidentification.

We can divide this module in two different parts: the bag and people identification and the predictions of bag's and people's positions done with Kalman.

4.2.0.2.1 People and bags

Through a function called Unchain, elements in the string are separated and the position, probability and label of every object detected in the image are checked. The application will only accept the labels "person" to detect People and "suitcase", "backpack" and "handbag" to detect Bags. We can see the flow chart of this part in the figure 4.2.

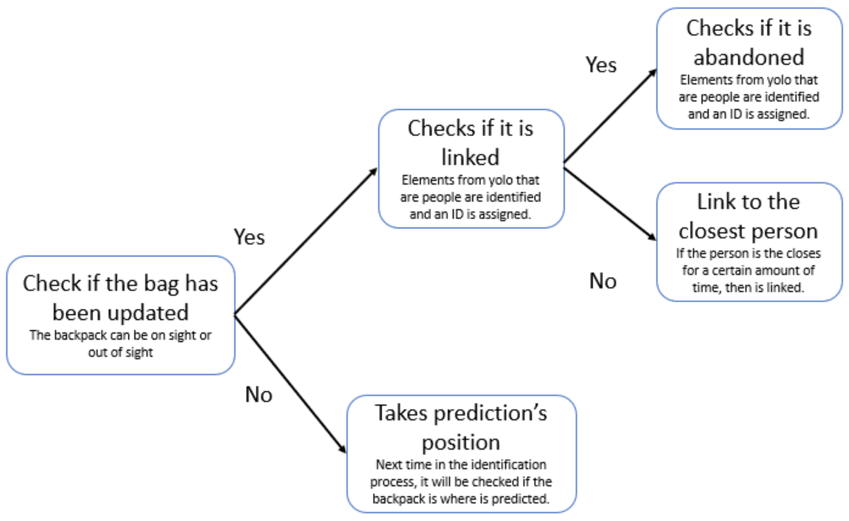

Check if the bag has been updated
The backpack can be on sight or out of sight
Yes
No
Checks if it is linked
Elements from yolo that are people are identified and an ID is assigned.
Takes prediction's position
Next time in the identification process, it will be checked if the backpack is where is predicted.
Yes
No
Checks if it is abandoned
Elements from yolo that are people are identified and an ID is assigned.
Link to the closest person
If the person is the closes for a certain amount of time, then is linked.

Figure 4.2: Identification and linking process flow chart

- For loop to classify object as people or bags

  YOLO returns the label of the object, the position and its certainty about the prediction and we distribute those in different variables, so it is easier to analyze.

  It is only valid for us if the label is a person, a backpack, a bag or a suitcase. In the beginning, we tried to make YOLO only return these objects by altering the number of filters in the cfg file by following certain steps that also would demand to edit labels. After some attempts, YOLO would not return any information, so it was decided to filter the labels later in the application.

  To better handle the backpacks, all the labels that refer to this object are changed inside the loop to bag. This will ease any function that depends on the label that an object has by only having to distinguish between person and backpacks.

  In case we have a bag or a person, it is found inside an if statement a function that determines or create a new object of these classes depending on certain factors that will be explained down below. These functions will then return the ID of the object that has been found in the image, which will be drawn in the image with the corresponding ID and label.

- Inside the if statement of the labels

  Inside the if statements, there are more functions that take place after determining or updating a new object. As we told before, this function will have an array that contains the information of the people, the current frame and the position of the new person and will return an ID with the person that YOLO has detected. The next process verifies the update state of the person, so the function called LastUpdate is used for this purpose. In case a person is not updated after a certain number of frames, this could mean that the person is no longer on screen and in order to avoid bad reidentifications in the future, we need to erase them from the array.

  Thanks to the ID, it will be possible to obtain the position of the person in the array to read the velocity in the current frame, which will be necessary in case that this person has a bag so the Kalman filter can be updated correctly.

  Finally, the status of the person will be extracted to be represented on the screen. A person which his bounding box is rose, is a person that does not have any bag linked. A green box means that a person has a bag linked to him. A red box is a person who has abandoned his bag behind.

- Determine new or update function for a bag object

  Remember that now the function has as an input the position of a new object and the current frame. This part of the code is similar to the detection of a new person but, for bags that have not been updated, we decided to use the Kalman filters. We can see the flowchart of this part in the figure 4.3.

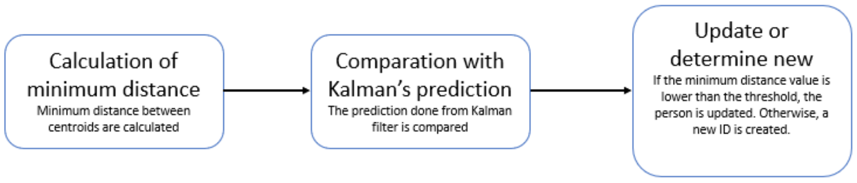

Calculation of minimum distance
Minimum distance between centroids are calculated
Comparation with Kalman's prediction
The prediction done from Kalman filter is compared
Update or determine new
If the minimum distance value is lower than the threshold, the person is updated. Otherwise, a new ID is created.

Figure 4.3: Determine new or update function flow chart for bags

- Determine new or update function for a person object

  This function is similar to the one described before but still; it has some differences that will be explained below.

  Given that YOLO does not fail so much detecting people, for now to do a tracking process we do it with the minimum distance between centroids and maximum area likeness. We can use kalman filters, which is optional, and the comparison would be done right after the calculation of the minimum distance.

  Aside from calculation the Euclidean distance to obtain the minimum distance between the centroid of the person that YOLO has found and the people that are already in the array, the area of the people are compared too. After some experiments, this technique has resulted useful to have a better reidentification when people are close to each other.

  As it was told with the backpack, we use a third part of the mean of all people in the image as a dynamic threshold. The first option to update a person is that the person with the minimum difference in area is under this threshold distance. The second possibility for an update is the person with the minimum distance.

  In case that none of these possibilities are fulfilled, a new person will be appended to the array and the ID of the new person will be returned.

4.2.0.2.2 Kalman

- Kalman associated to objects

  Similar to what is done to determine new or update, a class has been created to relate the ID of the backpack or the person and the Kalman filter associated to that object. If the ID is not found in the array, it means that a new backpack needs a new filter and therefore, it will be created and added to this Array.

- Updating the filter with measurements and making a prediction

  As it has been seen in the theoretical bases, the position of the bag, as well as the speed of the bag or, in case it is linked to a person, the speed of the person, will be fed into this function. Thanks to Kalman, we will obtain a prediction of the bounding box's centroid in the next frame.

- A prediction with objects that have not been updated

  In case that a bag has not been updated, out from the loop where bags and people are updated, the array will check bags that have not been updated and we will predict the position and draw the bounding box of the backpack's possible position on the screen. This will ease the reidentification if YOLO detects the backpack later on, given that the actual position will be close to the estimation we have done.

  In case it is a person, we have decided to repeat the prediction stage until it is found again.

4.2.0.3 Linking Process

Right after the People,Bags and Kalman Arrays have been updated, they are sent into the function LinkedBackpack, which takes care of doing this third part of the process. We can see the flow chart of this part in the figure 4.4.

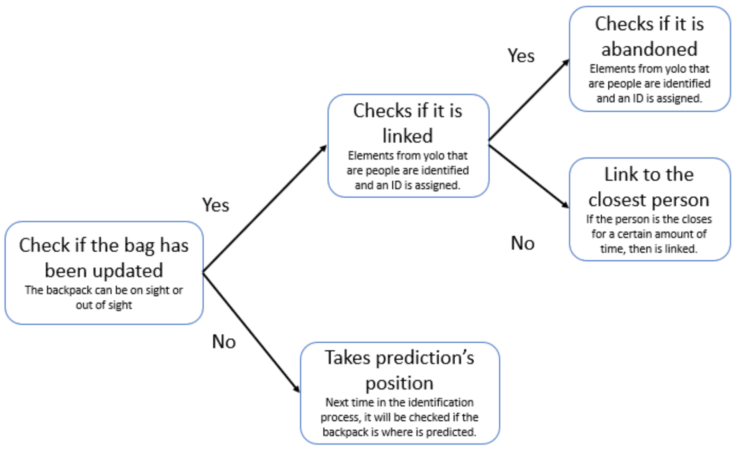

Check if the bag has been updated
The backpack can be on sight or out of sight
Yes
Checks if it is linked
Elements from yolo that are people are identified and an ID is assigned.
Yes
Checks if it is abandoned
Elements from yolo that are people are identified and an ID is assigned.
No
Link to the closest person
If the person is the closes for a certain amount of time, then is linked.
No
Takes prediction's position
Next time in the identification process, it will be checked if the backpack is where is predicted.

Figure 4.4: Linking process flow chart

- Linked backpack function

  In this function, we will compare over the time if a bag is close to a person long enough to classify it as an union and, in case a person is linked to a bag and the distance between both of them is above a threshold, an alarm will be triggered. Initially, a minimum distance will be compared between the backpack and the people on scene. Only distances with people that have been updated will be measured, saving only the minimum distance into an array.

  After a short period of time, if the array is full showing that a person has been close enough to the backpack, we will determine this person the owner of the bag. With every new frame, the oldest position will be erased and a new one will be added and reorganized into the frame. This is used to have just the necessary information in the array.

  As it has been done before to reidentification, the backpack needs to be at a close distance to the most repeated owner. If the minimum distance is not maintained, it could mean that the backpack is already in the scenario and anybody is close enough to be its owner. If the bag has not been linked yet, this person will be marked linked to the bag and, in case somebody is already linked, nothing will happen. This step is needed in case a person stops to talk to another one and the centroid of the bag ends closer to the other person, the function will not make this new person the owner of the bag.

In case that the bag is updated and the distance of the bag with the person is above the threshold after being linked, the function called check abandoned will see if the bag is close enough or if it is needed to raise an alarm.

- Check abandoned function
  Check abandoned will measure the distance between the bag and the person. If the bag is further than two times the width of the person, we can objectively say that the backpack is too far away from its owner. This distance will be saved into an array. The function will measure if during a close gap of time the distance is above the threshold. The array will be checked and if these distances are incremental or higher that the height of the person, which is a higher distance that the width of the person, we will confirm the alarm and the status of the backpack and the person associated will be turned into red.

4.2.0.4 Results

As we were advancing the project, there are some problems with YOLO detecting small object at long distances. In addition, backpacks when are abandoned sometimes lose their natural look and become shapeless objects in the ground, which complicates things even more to this classifier.

The attempt of this project was having classified the object that was lost while having a great reliability on the actual class of the object. Perfect conditions are needed to have positive results as it is shown in some examples. It can be said that the algorithm works, but as it has been proved, certain conditions are needed for a proper operation. We can see one of our first results in the figure 4.5.

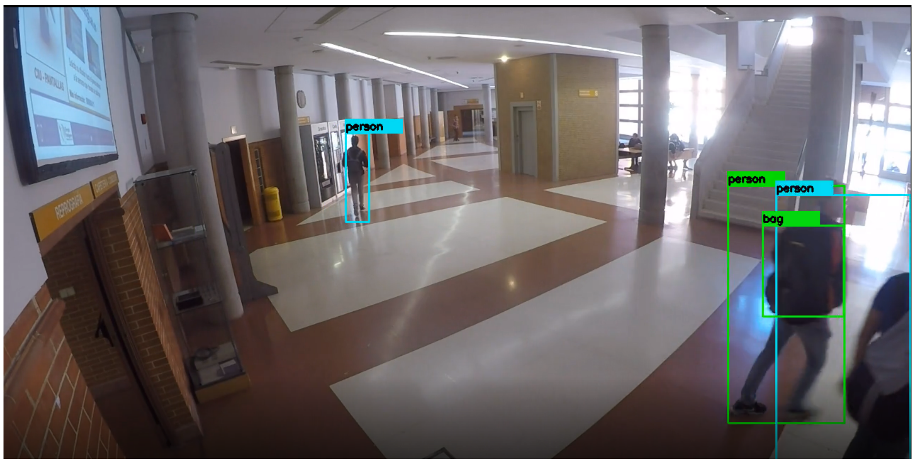


Figure 4.5: One of the first positive results from this module. At first, it was promising.

The database we have used to start the experiment and therefore, the one we have used for doing the trial and error testing is our own dataset called GBA [2] for abnormal activity as seen in the figure 4.6. The main footage of the database is oriented for classifying different actions. In the beginning, we used this database to check if YOLO was good enough to detect the backpacks and the people that were walking through the corridor. First videos showed that we could work with it and we could detect and link people with backpack as long as they were not going too far. This seemed to be a good start and so, new videos were recorded to have a more specific dataset for to abandoned objects.

As we recorded the datasets, the testing we did with YOLO showed that our recordings were not as good as we expected. If we check the dataset, we can see that there are too many people in most of the footage, which despite of being real for some situations, it can be difficult for testing.

Another flaw of our dataset is the use of not conventional backpacks, which could make YOLO struggle when one of these bags appears in the images. We have tried several videos and only one has had enough rate of detection to show how this module works.

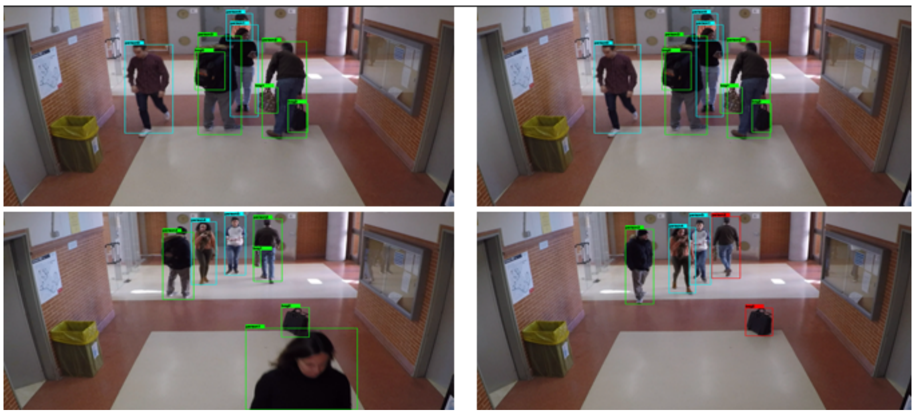


Figure 4.6: From GBA 07-12 near abandoned objects clip, we have obtained a video with enough rate of detection to show what we would like to expect from this module.

We have tried another dataset, PETS2006 [1] as we can see in the figures 4.7, 4.8 or 4.9, which even though it is a good dataset for detecting lost objects, the quality of the images and the distance between the camera and the people is too far away to properly allow YOLO detect the backpacks.

PETS2006 also, due to the quality of the image, gave us some problems when we tried to detect people and keep up with the reidentification. After using the Kalman filters for people, we could see an improvement to reidentify people.

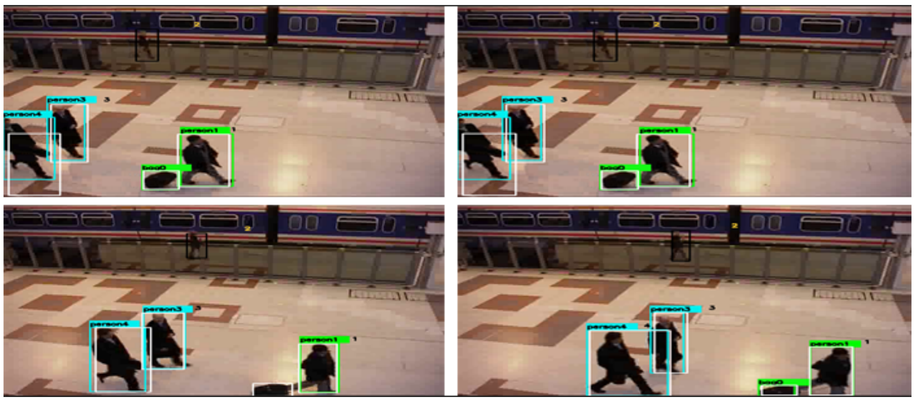


Figure 4.7: From PETS2006 S1-T1-C3 clip, we have seen that YOLO struggles to detect backbacks and suitcases much more than with the GBA dataset.

Keeping an acceptable reidentifying rate helped us to assign and detect, as it was our objective, who was the person responsible of abandoning a certain bag (in case it was previously linked to a person).

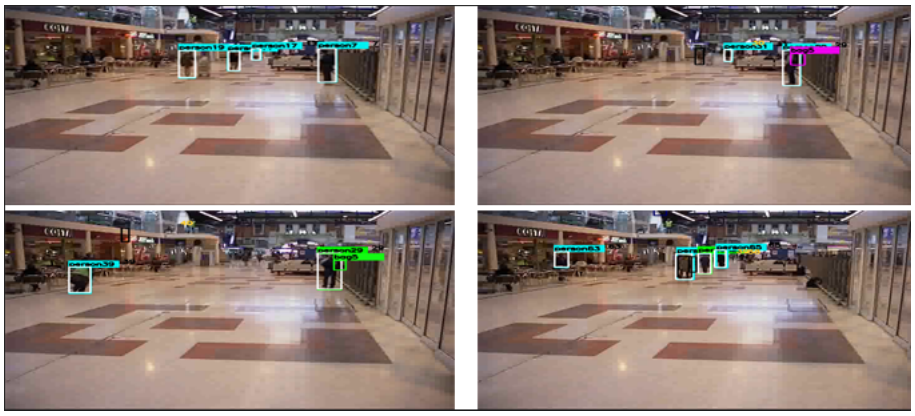


Figure 4.8: PETS2006 S1-T1-C1 clip.

The number of videos tested were three from PETS2006 [1] dataset and seven from GBA [2]. Only in one of those videos, YOLO could detect abandoned backpacks. We believe part of the problem could be the training dataset used, so for future test, we would recommend retraining YOLO with a longer dataset of images, including pictures of backpacks on the floor and in different positions : We have observed that most of the times, the detection rate is higher when the people are carrying the backpacks than when the backpacks are standing on the ground.

The detection phase runs images at 5.3 fps approximately, while the identification and linking process runs at 13 fps approximately. The system we are using a PC with Intel Core i7-8700 - 3.7 GHz CPU processor with a GeForce GTX970 GPU.

We believe that with some improvements, we consider that its performance can increase the percentage of success rate to acceptable levels. These solutions may be:

- As other projects have already accomplished, add background subtraction to recognize objects when YOLO cannot complete this task can be an alternative to have in mind in the future.
- Train YOLO with a better dataset of backpacks. We believe that, even though empty bags can be shapeless, this could improve the detection rate.
- Test different classifiers such the Faster R-CNN or the SDD. New versions of YOLO may come out and with it, improvement in the quality of detections.

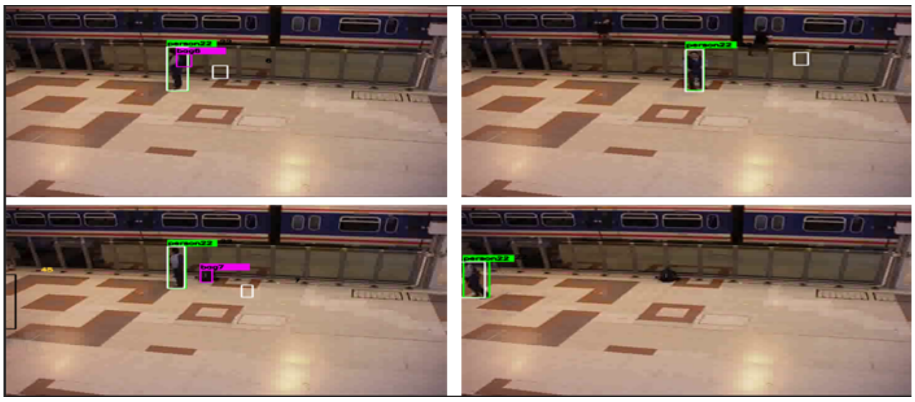


Figure 4.9: PETS2006 S1-T1-C3 clip.

To sum up, the software can prove that our lost object detector could be a good application for automated surveillance, but the lack in certain aspects of the deep neural network such as problem with the detection of little objects or the look that backpacks have when they are dropped, drags some problems that needed to be addressed. Still, we believe that with new additions in the future like background subtraction or the improvement that networks may have over the time, it is possible to have a better application in the future.

4.3 Heatmap generation

As the previous module, the backbone of this module is YOLO, and thanks to this network that detects and classifies object and, what is interesting for us, it enables us to filter and show only the people's trace.

Other alternatives that do not require a deep network to detect if a person is detected on an image could have been useful too. Background extraction, blob fusion and aspect ratios may could have had similar results. But this project already has put time investigating deep neural networks and it seemed a good option to take advantage of it and use it now that there was another chance to do it. We can see the flow chart in the figure 4.10.

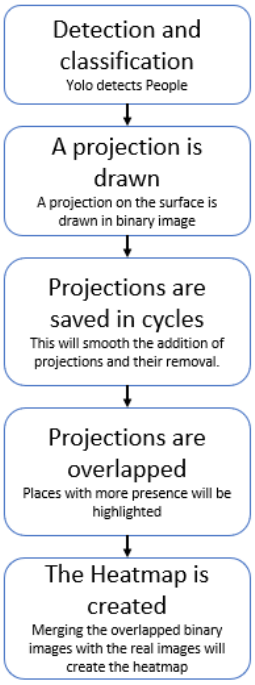

Detection and classification
Yolo detects People
A projection is drawn
A projection on the surface is drawn in binary image
Projections are saved in cycles
This will smooth the addition of projections and their removal.
Projections are overlapped
Places with more presence will be highlighted
The Heatmap is created
Merging the overlapped binary images with the real images will create the heatmap

Figure 4.10: Heatmap flow chart.

For every person detected on the image, an ellipse is drawn on their feet. Given that most of the cameras are located at a given height from the floor, this shape fits perfectly where people have been while they were walking or just staying on the floor.

To have this information, binary images are created, and these projections are drawn there. After all ellipses have been drawn in the binary image, they are saved in a folder where the next binary images are going to be saved too. Cycles of images are saved in folders and these cycles at the same time are saved in folders to decide the gap time that is desired to have in consideration to draw the imprints on the image.

With simple merging functions from OpenCV, the projections will have a weight and will be overlapped all together. Then, these will be merged with the actual image and the trace of the people will be visible.

The number of images saved will impact on the temporal trace it is desired to have on the images. Larger times reflected on the image with long-time heatmap imprints will be possible, but we have noticed that this also leads to have a slower process due to the time that it takes to overlap a large number of binary images and merge them with the actual image.

4.3.1 Results

This module shows in which places of an image there is more transit and where not. Group of images can show if, for example, people tend to stay more in a zone or on the other hand, if they walk straight and do not even stay to check what is there : In shops, it could be useful to know which parts are more visited and which parts do not bring so much attention. In the university, it can be used to see if people tend more to use stairs or go for the elevator. We can see actual results in the figure 4.11 and 4.12 .

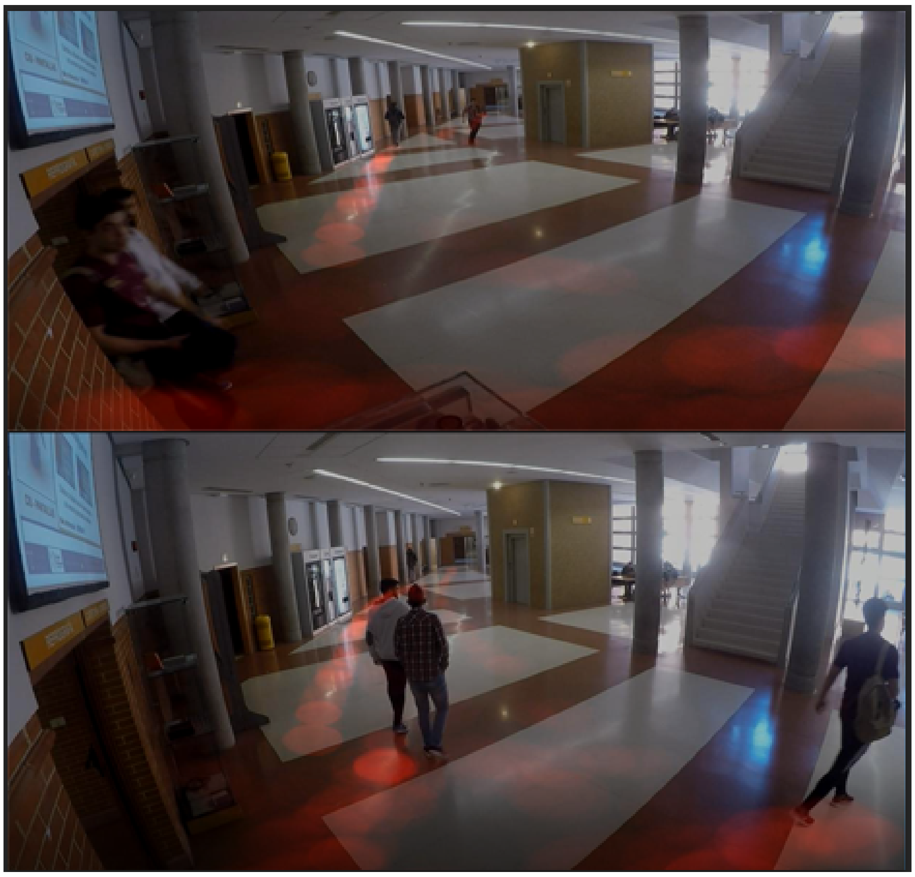


Figure 4.11: Heatmap results on GBA.

Future improvements for this module would be finding a solution to have long-time trails without sacrificing time to add these binary images to the actual image. It is obvious that the more information we have, the better and right now, we are not quantifying with numbers the density of people that the traces in the images can have.

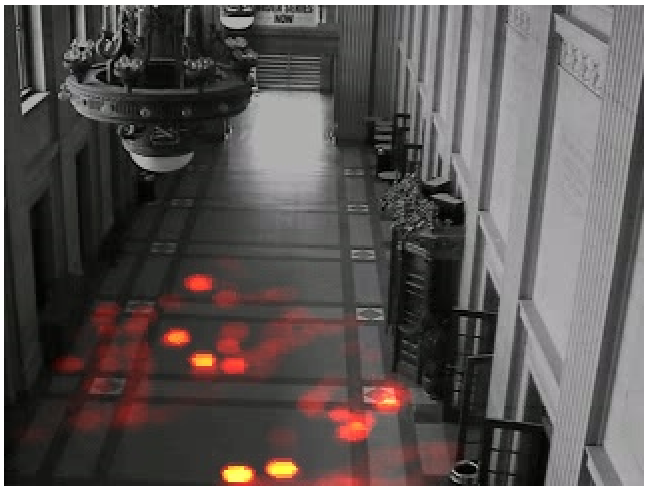


Figure 4.12: Heatmap results on UMN

In addition for the future, tools to translate this visual information in more statistical information would be useful too and would add more value to this framework if it is well executed, pointing as we said the quantity of density that the traces represent or for example, using different colors to display the evolution of density over the time. For example, as it was seen in the research, another project would track and count the number moving objects of the most transited places in the image, an option that could be easily implemented in the future given that thanks to YOLO, we already know the number of people in the image.

4.4 Stampede

In this module, we have developed an algorithm that mainly works with optical flow and k-means clustering, handling the information of the training data in codebooks and using it to create a threshold that detects anomalies in crowded environments. It is a simple solution that requires low use of resources from the computer, which is important in order to detect anomalies in real time.

We have overviewed different solutions that have similar approaches and that keywords as k-means, clustering and optical flow can be found in their projects. But after deeper reviews on their articles, it couldn't be found a full coincidence on the methodologies. Still, it is needed to say that at a first glimpse, some of them seem to perform outstandingly.

It is obvious after doing this research that optical flow is a good base to create a method to detect anomalies on image sequences. We have seen that its calculation and the consequent use of different algorithm can bring to light events that show disparity corresponded to what can be found compared to normal events.

Our project also took a glimpse at [58] work, which was not leaded towards the use of k-means clustering but more in the relevance of the optical flow and the influence on what was the optical flow and direction in the neighbor cells, studying also the direction of the movement.

Still, for the approach it was wanted to be taken, the thoughts on what was represented on the datasets, the direction of the movement itself did not have a direct impact, but the trajectories could have indeed. Still, to detect panic situations, an abnormal change of flow in the crowd is what to our eyes, would be what our mind would trigger the feeling of a suspicious event taking place on the images.

Following these thoughts, we experimented with different data sources to see what influences the quantity of flow through the image sequences:

After placing a grid that could reveal values like the average flow and maximum flow in every cell, it was revealed that for cells that had movement, it was not so important the number of people in one cell moving rather than the distance and angle of the camera, computing greater values given the displacement of the pixels in the cell.

We will explain in depth the steps in the flow-chart, giving deeper insights on what is happening on each step. Training flow chart is shown in the figure 4.13.

4.4.1 Image resizing

We present an overview of this procedure in the figure. Initially, it was decided to adapt the image size to a specific size. We do this conversion to work with lighter images and reduce the computational cost. Also, it is useful to optimize the variance of optical flow, given that it depends on the object velocity variations. Having the same size and knowing that cameras use to be located at a same height, this will minimize variations in the optical flow from different footages.

As it was pointed before, this is obviously not the only variable that influence the amount of flow detected on each grid. The distance between the people and the camera has its influence, where bigger variations will be measured if the camera is closer to the people, which is translated to an increase of optical flow. Still, this step is valuable because we reduce the computational cost while preserving the information in the image, which is more efficient.

4.4.2 Optical Flow Calculation and Grid calculation

Optical flow is the impression produced when an object is displaced in two consecutive frames, where a vector of movement can be extracted and describe the translation of a point between them. In other words, it measures the translation of a point with a movement gradient.

$$I(x, y, t) = I(x + dx, y + dy, t + dt)$$

After applying Taylor series approximation, the formula equation will be:

$$fx * u + fy * v + ft = 0$$

where:

$$fx = df/dx; fy = df/dy; u = dx/dt; v = dy/dt;$$

Lukas-kanade method takes a 3x3 patch around the point. Applying Lukas-kanade on an image, the function used will set a number of pixels which will be tracked, and the optical flow and scale will be returned.

There are different implementations for obtaining the optical flow. The one we use in this project is based on Gunner Farneback's algorithm [59], which is a two-frame motion estimation algorithms that works in two steps : first, approximate each neighborhood of both frames by quadratic polynomials and then under certain transforms, the result is translated to an estimated displacement.

To sum up, using this function it is possible to obtain the values of translation in cartesian values of pixel displacement. The function also includes some parameters that can be modified, but their values are standardized for general use and it has been proved that those values work well.

A conversion is done to obtain the absolute magnitude of displacement as well as its angle. We remind that, for panic events, independently of the direction that the crowd is moving, what is going to be measured to detect an abnormal event is the increase of the general flow in the image.

The image is then divided in a NxN grid (6x6 in this case). In each cell, each value of magnitude is introduced into an array to obtain the mean value of that array and hence, obtain the average flow motion of the cell. A condition is also implemented where the mean cannot be too high, representing a bad read on the data or zero. All these means are introduced again in a new array, which is going to give us the information of the general amount of movement that we have in one scene.

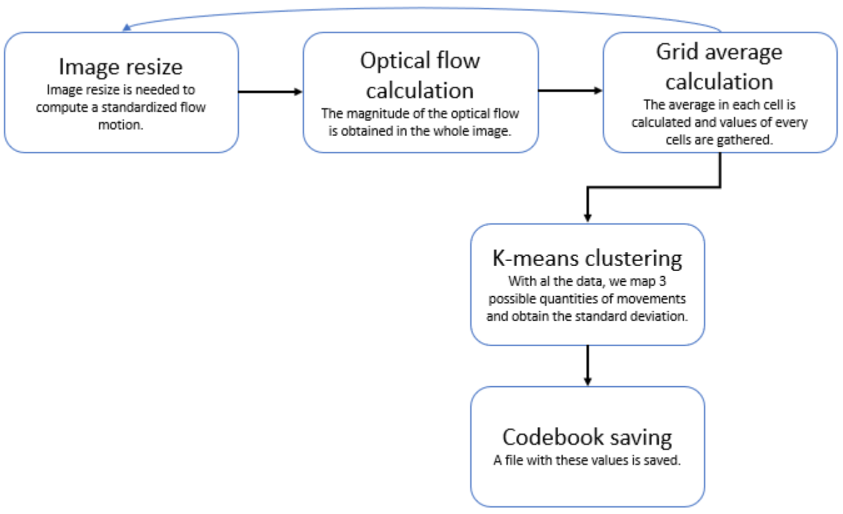


Image resize
Image resize is needed to compute a standardized flow motion.
Optical flow calculation
The magnitude of the optical flow is obtained in the whole image.
Grid average calculation
The average in each cell is calculated and values of every cells are gathered.
K-means clustering
With al the data, we map 3 possible quantities of movements and obtain the standard deviation.
Codebook saving
A file with these values is saved.

Figure 4.13: StampedeT flow chart.

4.4.3 K-means clustering

To correctly treat this information, we have decided to organize it using k-means clustering [60], a simple but powerful machine learning algorithm. This algorithm performs a numerous of repetitive calculations to find which is the centroid of the k-clusters that are desired to be found in a series of data, which is calculated until it is found the minimum sum of squares between the centroid of the cluster and its data. K-means cluster are represented in the figure 4.14.

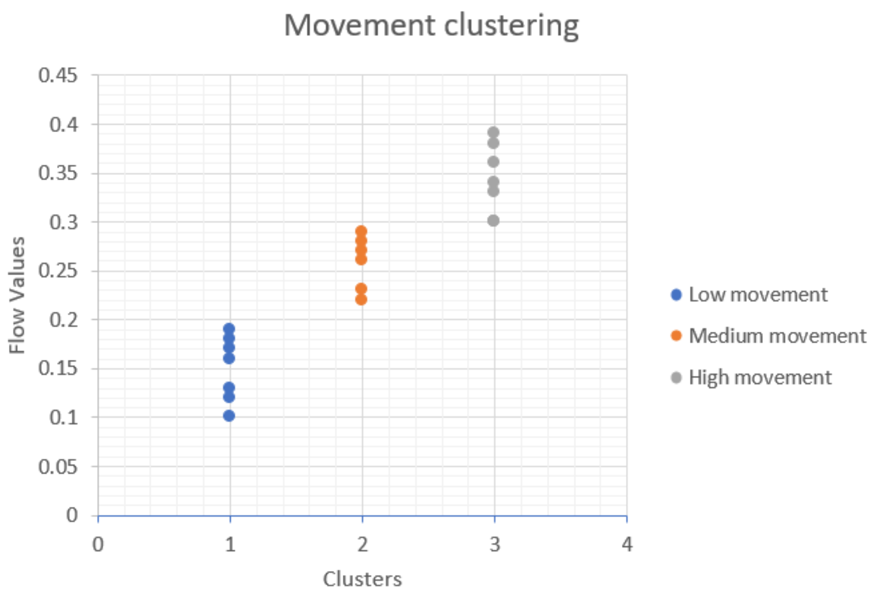


Figure 4.14: K-means graphic.

It is decided to use three clusters because, after observing the videos, it was considered it has enough variation in the range of the clusters to correctly adapt to the new observations in these groups. An example of the grid representation is on the figure 4.15.

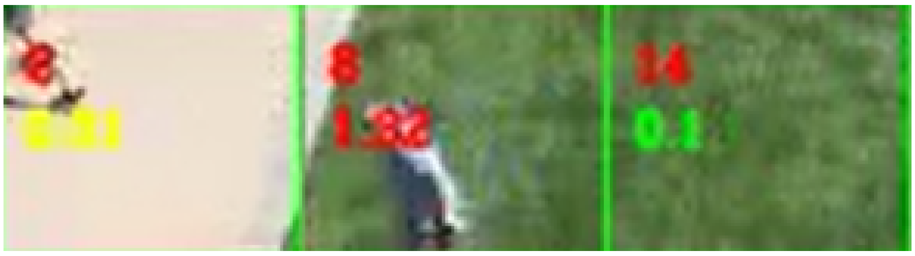


Figure 4.15: Stampede grid result.

4.4.4 Anomaly detection

An increase of movement that is above the maximum values of high amount of movement would certainly mean that something is happening and therefore, an abnormal event may be occurring. To detect it a threshold gate is created, and it demands that an average flow in one cell must be higher than the value of the centroid obtained for the high-speed movement cluster plus its maximum deviation. The expression that has been used is:

Where F is the multiplier factor that is equal to one plus the square of five of one divided by the compactness K. The one that is out of the square of five is useful in case that the compactness is too high and, in the end, the items in the root are low. In case the compactness is too low, the root will return a more acceptable variable. These have come up after trying with different videos were the normal flow motion would work in a range of 0.5 to 0.7, to videos where the normal flow motion goes between 2 and 3.

Finally, to obtain the threshold, the centers obtained are multiplied by this factor, which will make them more suitable to find events that go from normal to abnormal. We can check the flow chart for the test part in the figure 4.16.

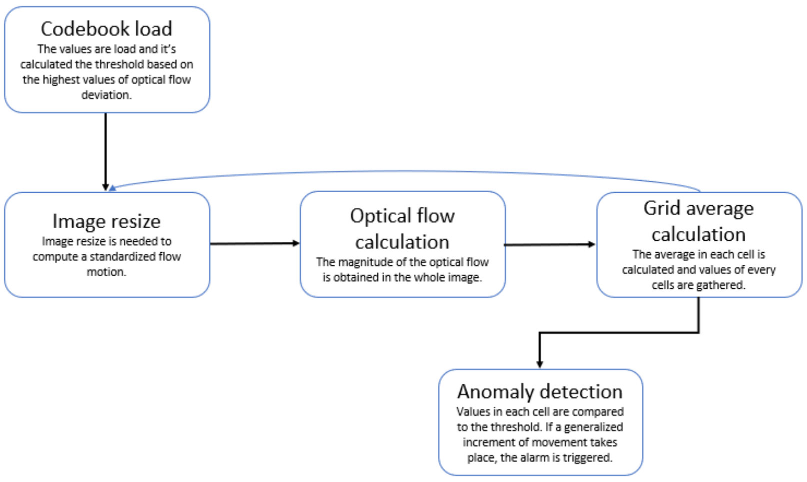

Codebook load
The values are load and it's calculated the threshold based on the highest values of optical flow deviation.
Image resize
Image resize is needed to compute a standardized flow motion.
Optical flow calculation
The magnitude of the optical flow is obtained in the whole image.
Grid average calculation
The average in each cell is calculated and values of every cells are gathered.
Anomaly detection
Values in each cell are compared to the threshold. If a generalized increment of movement takes place, the alarm is triggered.

Figure 4.16: Stampede flow chart.

Having good results imply to avoid false positives and what is more important, not having false negatives, which means that the framework would be a total failure. To prove the efficiency of the algorithm, it is needed a fair dataset to check the performance in different situations.

In the beginning, different approaches were tested, where instead of using k-means clustering, we used maximum values of optical flow to determine if an anomaly would be taking place through a number of frames. After obtaining bad results, average was checked but still, results in different dataset were despair when we tried to establish objective numeric relations between the number of people detected and the optical flow obtained in certain points. We tried using K-means then, and we achieved good results to detect the increase of movement in different situations, which was a good starting point.

As we mentioned, a local point where an abnormal amount of flow is detected does not have to mean anything. In the corridors of the university, it can mean that a person is going late to classes and is just running to arrive in time. Therefore, it must be a generalized condition.

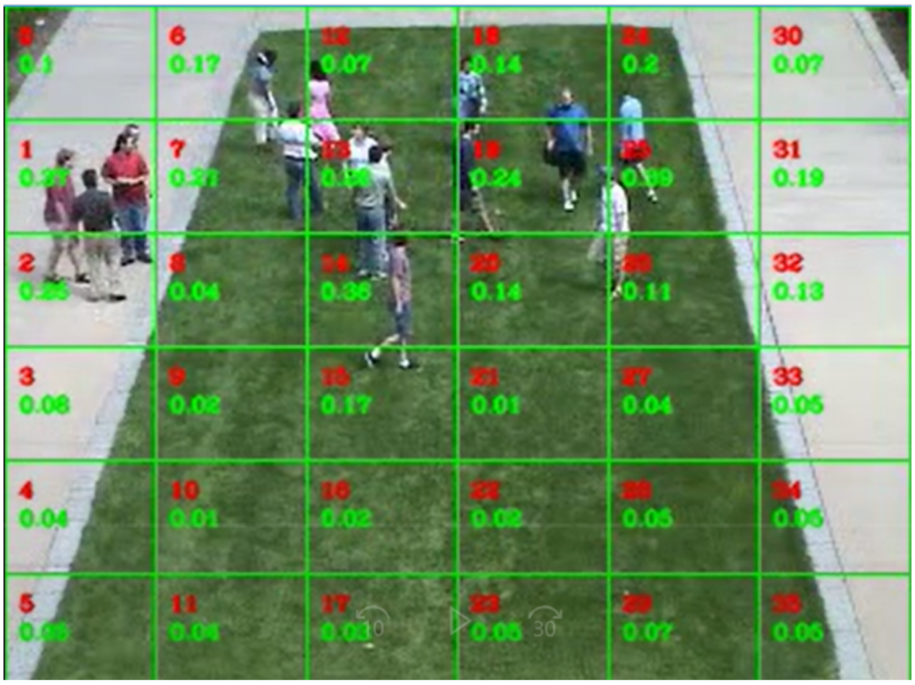


Figure 4.17: Stampede normal activity example.

Our proposal is that in five different cells an abnormal amount of flow is detected, an abnormal event is occurring in the clip and an alarm would be displayed in the terminal. Thanks to the k-means clustering, training with videos makes that the centroid and the deviation can be adapted to different environments, which means that it can be higher if the camera is closer to the ground and detects higher quantity of movement, or in the opposite conditions, can be more sensitive if it is far from the crowd and lower variations trigger the threshold. We can see an example of the grid placed in the figure 4.17.

4.4.5 Codeboook information

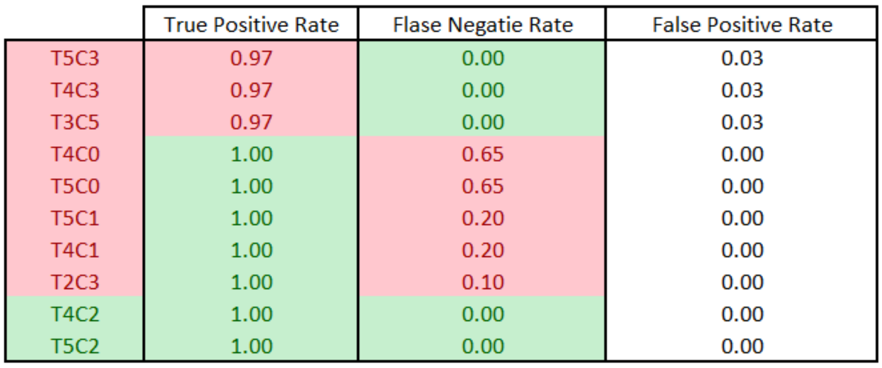
In this file, not much information is saved. To speed up also the information retained from the video, only 1 every 10 frames are going to be measured, which has been proved after test that still can have enough information from the training process. After applying k-means clustering, information related to the position centroids and the standard deviation of the values that conforms each cluster is saved. This information obtained in the training process and loaded in the testing process.

4.4.6 Results

We have used the UMN dataset [3] to train and test our algorithm. The dataset consist in three groups of videos, where the first group contains two videos, the second six videos and the third three videos.

With every group of videos, we have divided each of them between normal and abnormal activity clips.

With the first group, we have trained with one clip of normal activity and tested it with the other three clips. With the second group, we have trained with one clip of normal activity and tested it with the other eleven. Finally, with the third group we have trained again with one normal activity clip and tested it with the other three. We can see how changing different thresholds related to number of cells used to trigger an alarm and frames, prompts different performance in the UMN dataset as we see in the figure 4.18.



	True Positive Rate	Flase Negatie Rate	False Positive Rate
T5C3	0.97	0.00	0.03
T4C3	0.97	0.00	0.03
T3C5	0.97	0.00	0.03
T4C0	1.00	0.65	0.00
T5C0	1.00	0.65	0.00
T5C1	1.00	0.20	0.00
T4C1	1.00	0.20	0.00
T2C3	1.00	0.10	0.00
T4C2	1.00	0.00	0.00
T5C2	1.00	0.00	0.00

Figure 4.18: Results achieved with our algorithm after trying with different frame duration and quantity of cells to raise an alarm.

As it was marked for the lost object detector, a bigger dataset could be evaluated in order to put this module under test with videos that could more easily trigger false positives to prove if the ideas that have built this part, are effective or if it need certain changes. We know that the best results we have achieved here are a bit unreal, given that in some normal situations, some people can run and the threshold would be activated for only two cells : We have played around with part of our dataset and, even though it is not prepared for abnormal crowd activity, we have seen that some people running around could trigger the threshold when there is not actually happening any abnormal situation. The display of an actual alarm is shown in the figure 4.19.

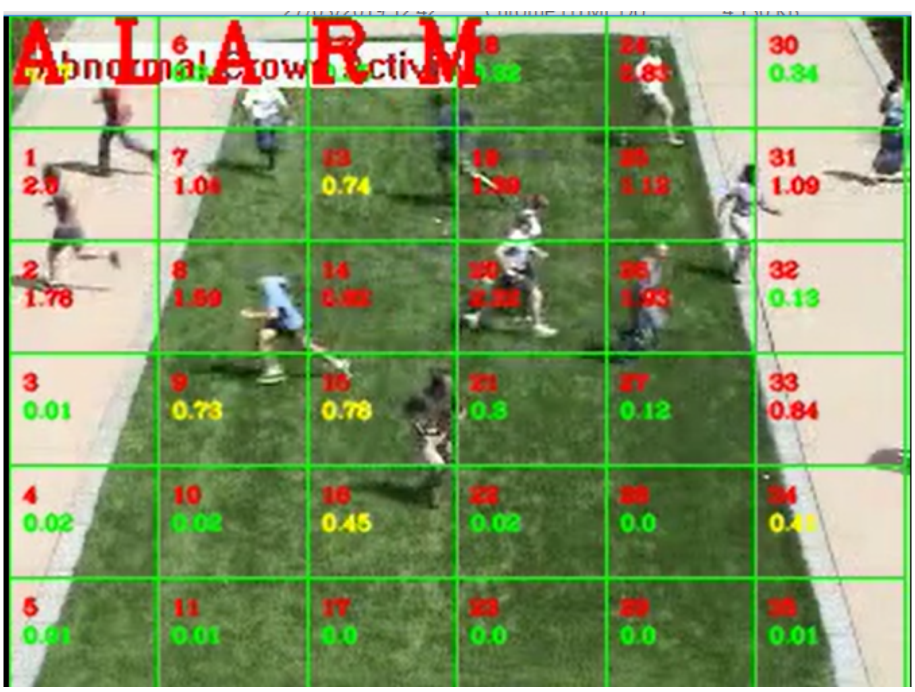


Figure 4.19: Alarm triggered in one of the test clips from the UMN dataset

Despite having a high success rate for the training and testing sets, the next step would be to improve the algorithm so we could use more than one train video at the same time and detect abnormal events of different sets without having to train our application again with a specific video of that set. This methodology is what projects like [12] already do, and it would be interesting to do so we could compare the success rate with the methodology they present.

A possible improvement to add to this module in the future could be to automate the training process and the testing and real use of the algorithm. We have already tested an early version of this idea, but the main problem of doing this is that real footage is needed, and it depends on what is the real velocity that it can be found through the image sequences. With a calibration process, we could start testing new additions in our algorithm and try to improve this application.

Bibliography

[1] "PETS 2006 benchmark data," http://www.cvg.reading.ac.uk/PETS2006/data.html, last visited May-2019.

[2] G. R. Group, "Unusual crowd activity dataset made available by the university of alcala de henares."

[3] "University of minnesota dataset," http://mha.cs.umn.edu/movies/crowdactivity-all.avi, last visited May-2019.

[4] "technoaware website," http://www.technoaware.es, last visited June-2018.

[5] omnitec website, "omnitec website," http://www.omnitec.es, 2019, online; accessed 18 January 2018.

[6] "Nuuo website," http://www.nuuo.es, last visited June-2018.

[7] H. Fradi and J.-L. Dugelay, "Crowd density map estimation based on feature tracks," in *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*. IEEE, 2013, pp. 040–045.

[8] X. Liu, D. Campbell, and Z. Guo, "Single image density map estimation based on multi-column cnn and boosting," in *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2017, pp. 1393–1396.

[9] A. Chmielewska, P. Marianna, T. Marciniak, A. Dabrowski, and P. Walkowiak, "Application of the projective geometry in the density mapping based on cctv monitoring," in *2015 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*. IEEE, 2015, pp. 179–184.

[10] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, "Describing videos by exploiting temporal structure," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4507–4515.

[11] Y. Cong, J. Yuan, and Y. Tang, "Video anomaly search in crowded scenes via spatio-temporal motion context," *IEEE transactions on information forensics and security*, vol. 8, no. 10, pp. 1590–1599, 2013.

[12] S. Wu, B. E. Moore, and M. Shah, "Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 2054–2060.

[13] K. Lin, S.-C. Chen, C.-S. Chen, D.-T. Lin, and Y.-P. Hung, "Abandoned object detection via temporal consistency modeling and back-tracing verification for visual surveillance," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 7, pp. 1359–1370, 2015.

[14] W. Huang, L. Jin, Y. Luo, Y. Li, and T. Cui, "A novel algorithm for abandoned object detection," in *2016 IEEE International Conference on Information and Automation (ICIA)*. IEEE, 2016, pp. 1583–1587.

[15] J. Redmon and A. Farhadi, “Yolov3: An incremental improvement,” *arXiv preprint arXiv:1804.02767*, 2018.

[16] C. Chen and L. Xia, “Recurrent neural network and long short-term memory.”

[17] S. R. Eddy, “Profile hidden markov models.” *Bioinformatics (Oxford, England)*, vol. 14, no. 9, pp. 755–763, 1998.

[18] J. Yamato, J. Ohya, and K. Ishii, “Recognizing human action in time-sequential images using hidden markov model,” in *Proceedings 1992 IEEE Computer Society conference on computer vision and pattern recognition.* IEEE, 1992, pp. 379–385.

[19] D. Helbing and P. Molnar, “Social force model for pedestrian dynamics,” *Physical review E*, vol. 51, no. 5, p. 4282, 1995.

[20] C.-D. Liu, Y.-N. Chung, and P.-C. Chung, “An interaction-embedded hmm framework for human behavior understanding: with nursing environments as examples,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 14, no. 5, pp. 1236–1246, 2010.

[21] Y. Wang, K. Huang, and T. Tan, “Abnormal activity recognition in office based on r transform,” in *2007 IEEE International Conference on Image Processing*, vol. 1. IEEE, 2007, pp. I–341.

[22] E. Dorj and E. Altangerel, “Anomaly detection approach using hidden markov model,” in *Ifost*, vol. 2. IEEE, 2013, pp. 141–144.

[23] S. Sharma, R. Kiros, and R. Salakhutdinov, “Action recognition using visual attention,” *arXiv preprint arXiv:1511.04119*, 2015.

[24] J. R. Medel and A. Savakis, “Anomaly detection in video using predictive convolutional long short-term memory networks,” *arXiv preprint arXiv:1612.00390*, 2016.

[25] D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “C3d: generic features for video analysis,” *CoRR, abs/1412.0767*, vol. 2, no. 7, p. 8, 2014.

[26] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, “Learning spatiotemporal features with 3d convolutional networks,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4489–4497.

[27] I. Laptev, “On space-time interest points,” *International journal of computer vision*, vol. 64, no. 2-3, pp. 107–123, 2005.

[28] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, “Action classification in soccer videos with long short-term memory recurrent neural networks,” in *International Conference on Artificial Neural Networks.* Springer, 2010, pp. 154–159.

[29] G. Thung and H. Jiang, “A torch library for action recognition and detection using cnns and lstms.”

[30] O. Boiman and M. Irani, “Detecting irregularities in images and in video,” *International journal of computer vision*, vol. 74, no. 1, pp. 17–31, 2007.

[31] V. Patraucean, A. Handa, and R. Cipolla, “Spatio-temporal video autoencoder with differentiable memory,” *arXiv preprint arXiv:1511.06309*, 2015.

[32] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, “Learning temporal regularity in video sequences,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 733–742.

[33] K. Min, L. Yang, J. Wright, L. Wu, X.-S. Hua, and Y. Ma, “Compact projection: Simple and efficient near neighbor search with practical memory requirements,” in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition.* IEEE, 2010, pp. 3477–3484.

[34] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, “Robust real-time unusual event detection using multiple fixed-location monitors,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 3, pp. 555–560, 2008.

[35] R. Mehran, A. Oyama, and M. Shah, “Abnormal crowd behavior detection using social force model,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE, 2009, pp. 935–942.

[36] Y. Zhang, L. Qin, R. Ji, H. Yao, and Q. Huang, “Social attribute-aware force model: exploiting richness of interaction for abnormal crowd detection,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 7, pp. 1231–1245, 2014.

[37] C.-Y. Cho, W.-H. Tung, and J.-S. Wang, “A crowd-filter for detection of abandoned objects in crowded area,” in *2008 3rd International Conference on Sensing Technology.* IEEE, 2008, pp. 581–584.

[38] R. K. Tripathi, A. S. Jalal, and C. Bhatnagar, “A framework for abandoned object detection from video surveillance,” in *2013 Fourth National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG).* IEEE, 2013, pp. 1–4.

[39] P. Viola, M. Jones *et al.*, “Rapid object detection using a boosted cascade of simple features,” *CVPR (1)*, vol. 1, pp. 511–518, 2001.

[40] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[41] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[42] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, “Selective search for object recognition,” *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[43] P. F. Felzenszwalb and D. P. Huttenlocher, “Efficient graph-based image segmentation,” *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[44] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[45] R. Girshick, “Fast r-cnn,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[46] S. Ren, K. He, R. Girshick, and J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” in *Advances in neural information processing systems*, 2015, pp. 91–99.

[47] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[48] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in *European conference on computer vision.* Springer, 2016, pp. 21–37.

[49] J. Redmon and A. Farhadi, "YOLO9000: better, faster, stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 2017, pp. 6517–6525. [Online]. Available: https://doi.org/10.1109/CVPR.2017.690

[50] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[51] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[52] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," *arXiv preprint arXiv:1602.07360*, 2016.

[53] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning," in *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, 2016, pp. 265–283.

[54] "Darkflow repository."

[55] R. Kromann, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," 2016.

[56] G. Bishop, G. Welch *et al.*, "An introduction to the kalman filter," *Proc of SIGGRAPH, Course*, vol. 8, no. 27599-23175, p. 41, 2001.

[57] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision.* Springer, 2014, pp. 740–755.

[58] "Unusual human activity detection repository," https://github.com/Poornav/Unusual-Human-Activity-Detection, last visited May-2019.

[59] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Scandinavian conference on Image analysis.* Springer, 2003, pp. 363–370.

[60] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern recognition letters*, vol. 31, no. 8, pp. 651–666, 2010.

[61] "Información sobre gnu/linux en wikipedia," http://es.wikipedia.org/wiki/GNU/Linux [Último acceso 1/noviembre/2013].

[62] "Página de la aplicación gcc," http://savannah.gnu.org/projects/gcc/ [Último acceso 1/noviembre/2013].

[63] "Página de la aplicación make," http://savannah.gnu.org/projects/make/ [Último acceso 1/noviembre/2013].

[64] "Página de la aplicación pycharm," https://www.jetbrains.com/pycharm/ [Último acceso 21/May/2019].

Appendix A

User Manual

A.1 How to run the application

There are two main modules in this project:

- In one module, we have the main python wrap of YOLO and the application has been built up around it. Given that two out of three parts work with YOLO for the moment, the main file that leads this application can be found there too. This file counts with the functions that are used mainly by the Stampede and Lost Bag Detector from this project.
- The second module takes care of abnormal crowd activity, and it contains the functions that calculate the optical flow density in the image and the procedures to train the software to enable the identification of abnormal crowd activities.

There are also other important modules that need a mention too because they contain important functions to make this framework possible:

- Kalman module contains the class based on Kalman filters that allow us to predict the position of the backpacks when they are not found by YOLO.
- The objects module has the classes of person and backpack, which are necessary to create the objects that contain the information of people and bags respectively in the videos analyzed by the Lost Object Detector module. This module also has functions related to update these classes and the function used to link backpacks to people.
- The configuration module has been created to ease the process of saving images and do other minor tasks that were repetitive in the process and were allocated in this file.

To run the application, we will need to open a console in the folder called obj/python, and it will be necessary to have the inputs as it is shown below:

python3 darknetandheatr.py -i /path/to/video –mode choose -kp -pb

Described with a double line, we can see the inputs we have available for our application. The possible values that these input accept are:

- Input(–i)

  We have to put a path that leads to the folder where we have the images of the desired video.

- Mode(–m)

  Through mode input, we can choose the different modes available for our application. The modes available right now are:

  - lostobjectdetection, which activates the mode that detect people and bags and raises an alarm when the application finds an abandoned bag.
  - heatmap, which activates the mode draws the people's trace. In case we have an image in the folder *base*, that image will be use to display the heatmap(this can be useful if we don't want to see the traces over the people but in a empty image).
  - Stampede or StampedeT, which trains and calculates the thresholds for detecting abnormal crowd activity or raises an alarm when the algorithm detect that the optical flow is higher than the threshold calculated.

- Use kalman with people(–kp)

  Through using kalman with people, we enhance the reidentification process when YOLO does not detect people in the images by using kalman filters. We just need to put the –kp to activate it.

- Use kalman with bags(–kb)

  Through using kalman with bags, we enhance the reidentification process when YOLO does not detect people in the images by using kalman filters. We just need to put the –kp to activate it.

- Use quick position prediction with bags(–pb)

  Through using quick position prediction with bags with bags, we enhance the reidentification process when YOLO does not detect people in the images by roughly prediction the bag position with the gradient of movement of the person who is linked to the bag. We just need to put the –kp to activate it.

- show prediction(–sp)

  Given that sometimes it can be too much information on screen, we believe that there is no need to show the predictions in the images. This is actually an internal process that help us with the reidentification but it does not help to the person who is watching the footage. In case we want to watch the prediction, we should use –sp to activate it.

To end, results will be allocated in the folder results. In the folder Heatmap, we will save temporary binary images in the folder temp.

In case we put an image in the folder heatmap/base, that image will be used to draw the heatmap in it. If there is any image in the folder, the heatmap will be drawn always in the current image displayed.

A.2 Tools and resources

The tools we have needed to develop this project have been:

- PC compatible.
- GNU/Linux O.S. [61].
- C/C++ gcc compiler [62].

- Make compiler [63].
- Pycharm IDE [64].

The libraries we needed for this project are:

- Numpy, to perform array operations
- CV2, to perform operations with the opencv library.
- OS, to perform operations related to paths and files.
- PIL, to perform some operations related to images.
- RE, to perform some string operations.