

UNIVERSIDAD AUTONOMA DE MADRID

ESCUELA POLITECNICA SUPERIOR



**Master Universitario en Deep Learning
for Audio and Video Signal Processing**

MASTER THESIS

**Real-time camera operation and tracking for the streaming of
teaching activities.**



**Javier Vinuesa Solana
Advisor: Jesús Bécscos Cano**

June 2021

**Real-time camera operation and tracking for the streaming of
teaching activities.**

AUTOR: Javier Vinuesa Solana

TUTOR: Jesús Bécós Cano

Video Processing and Understanding Lab. (VPULab)

Dept. Tecnología electrónica y de las Comunicaciones

Escuela Politécnica Superior

Universidad Autónoma de Madrid

June de 2021

**This study has been partially supported by the Spanish Government through its
TEC2017-88169-R MobiNetVideo project.**



Resumen (castellano)

La principal razón de este trabajo proviene de la necesidad del Laboratorio en ofrecer a los estudiantes la oportunidad de asistir a un evento desde casa o desde cualquier parte del mundo en tiempo real. El objetivo principal de este trabajo es construir un tracker que funcione a tiempo real para seguir los movimientos del conferenciante. Después construiremos un framework para manejar una cámara PTZ (Pan Tilt and Zoom) basándose en los movimientos del conferenciante. Es decir, si el conferenciante va para la izquierda, la cámara girará hacia la izquierda.

Para abordar este proyecto continuaremos un proyecto desarrollado por Gebrehiwot, A. que consistía en construir un tracker que funcionaba tiempo real. El problema de este tracker es que estaba implementado en Ubuntu y se ejecutaba con una CNN muy compleja que requería el uso de una GPU en nuestro ordenador. Como bien señala Gebrehiwot, A. al final de su informe, no todo el mundo tiene una partición de Ubuntu o una GPU en sus ordenadores, así que el primer objetivo era portar el tracker a Windows. Para lograr este objetivo utilizamos Anaconda Windows, que nos facilitó mucho el trabajo. Después implementamos un backbone ligero para el tracker que nos permitía ejecutarlo en ordenadores con poca capacidad de procesamiento. Una vez realizado todo este proceso, pusimos en práctica el mencionado framework para manejar el movimiento de la cámara PTZ. Este framework utiliza el tracker ligero implementado para seguir los movimientos del profesor/conferenciante y, en función de estos movimientos, la cámara se desplazará y se inclinará automáticamente. Hemos probado este framework en plataformas de streaming como YouTube demostrando que puede mejorar mucho la calidad de las clases online.

Finalmente sacamos conclusiones del trabajo realizado y proponemos trabajo futuro que se podría hacer para mejorar el framework.

Abstract (English)

The primary driving force of this work comes from the Lab's urgent needs to offer students the opportunity to attend a remote event from home or anywhere in the world in real-time. The main objective of this work is to build a real-time tracker to follow the movements of the lecturer. After that we will build a framework to handle a PTZ (Pan Tilt and Zoom) camera based on the lecturer movements. That is, if the lecturer goes to the left, the camera will turn to the left.

To tackle this project we will follow a project developed by Gebrehiwot, A. which involved building a real-time tracker. The problem of this tracker is that was implemented on Ubuntu and running with a very complex CNN which required the use a good GPU on our computer. As Gebrehiwot, A. rightly points out at the end of his report, not everyone has an Ubuntu partition or a GPU on their computers so we started moving the real time tracker to Windows. To achieve this objective we used Anaconda Windows which made our work much easier. After that we implemented a lightweight backbone of the tracker allowing us to run it on computers with a fewer processing power. Once that all this process was done, we put into practice the mentioned framework for handling the movement of the PTZ camera. This framework uses the implemented lightweight tracker to follow the lecturer moves and depending on these movements the camera will pan and

tilt automatically. We tested this framework on streaming platforms like YouTube proving that can greatly improve the quality of online classes.

Finally we draw conclusions from the work done and propose future work to improve the framework.

Palabras clave (castellano)

Aprendizaje profundo, redes neuronales, redes convolucionales, seguimiento de objetos en video, cámaras PTZ.

Keywords (inglés)

Deep Learning, neural networks, convolutional networks, video object tracking, PTZ cameras.

Acknowledgements

Primero de todo, quiero dar las gracias a mi tutor, Jesús, por su gran ayuda y compromiso durante este año.

Agradecer también a mis padres y mi familia por su gran apoyo y confianza durante este año, no solo académicamente, sino en todos los aspectos de mi vida. Nunca lo hubiese conseguido sin vosotros.

Gracias a mis amigos de toda la vida, tanto arenaleros como madrileños, por el apoyo y los buenos ratos que me habéis dado. Habéis hecho que este viaje se haga mucho más ameno. Gracias también a mis compañeros de clase, en especial a Daniel, con el cual empecé la carrera y acabo el máster. Sois unos cracks.

CONTENTS

1 Introduction	2
1.1 Motivation	2
1.2 Objectives	2
1.3 Structure of the report.....	3
2 State of the art.....	6
2.1 Video object tracking.....	6
2.1.1 What is Video object tracking?	6
2.1.2 Differences with Video Object Detection	6
2.1.3 Single Object Tracking vs Multiple Object Tracking	7
2.1.4 Challenges in Single Object Tracking	7
2.1.5 Main components in Single Object Tracking.....	8
2.1.6 Deep Learning Methods for tracking.....	9
2.1.6.1 Taxonomy of Deep Trackers.	9
2.1.6.2 Siamese Region Proposal Network	11
2.1.6.3 Lightweight CNN's	12
2.2 PTZ Cameras	12
3 Development.....	14
3.1 Previous work	14
3.1.1 System Design	14
3.1.2 Drawbacks of the system proposed	15
3.2 Objective 1: Port the system to MS Windows.....	16
3.3 Objective 2: Test lightweight CNN's for target tracking.	17
3.4 Objective 3: Design and implement the camera operation module.....	19
3.4.1 Communication between PC (laptop) and PTZ camera.	19
3.4.2 Implementation of the camera operation module.	20
3.4.2.1 Extension of the camera operation module	22
4 Integration, testing and results.....	23
4.1 System requirements	23
4.2 Setting up the system.....	24
4.2.1 Hardware Set up	24
4.2.2 Software Set up.....	26
4.3 Video Live Streaming.....	29
4.4 Demos.....	32
5 Conclusions and future work.....	33
5.1 Conclusions	33
5.2 Future work	34
Bibliography	36

LIST OF FIGURES

FIGURE 2-1 DIFFERENT TYPES OF STATE IN TRACKING. SOURCE: [2]	6
FIGURE 2-2: REPRESENTATION OF MOTION PROBLEM. SOURCE: [2].....	8
FIGURE 2-3: REPRESENTATION OF MATCHING PROBLEM. SOURCE: [2]	8
FIGURE 2-4: TYPICAL VIDEO TRACKING PIPELINE. SOURCE: [2].....	8
FIGURE 2-5: ARCHITECTURE OF SIAMRPN. IT IS DIVIDED IN TWO, AT LEFT THE SIAMESE NETWORK AND AT RIGHT THE REGION PROPOSAL NETWORK. SOURCE: [7]	11
FIGURE 2-6 : ARCHITECTURE OF MOBILENET. SOURCE: [8].....	12
FIGURE 2-7: PTZ CAMERA	13
FIGURE 3-1: PREVIOUS PROPOSED METHOD: HARDWARE SET UP. SOURCE: [1]	14
FIGURE 3-2: GENERAL OVERVIEW OF THE PROPOSED SYSTEM	15
FIGURE 3-3THIS IS WHAT IT SHOULD APPEAR WHEN YOU TYPE ‘PYTHON’ ON THE ANACONDA PROMPT AND THE ANACONDA IS WELL INSTALLED.....	16
FIGURE 3-4: SIAMMASK ARCHITECTURE. WE CAN SEE THE ADDED BRANCH TO MAKE SEGMENTATION MASK (THE FIRST ONE STARTING BY THE TOP). THIS IS THE TRACKER USED IN AWED’S ALGORITHM.	18
FIGURE 3-5: EXAMPLE OF MANUAL TARGET SELECTION	21
FIGURE 3-6: SYSTEM OF RULES TO HANDLE THE MOVEMENTS OF THE PTZ CAMERA.	21
FIGURE 3-7. ZOOM FUNCTIONALITY INCLUDED IN THE RULE SYSTEM.....	22
FIGURE 4-1: SWITCH CONNECTIONS. THE ETHERNET BLUE CABLE DIVIDE THE INTERNET SIGNAL IN THE TWO WHITE CABLES. THE WHITE CABLES WENT ONE TO THE LAPTOP AND THE OTHER TO THE PTZ CAMERA. THE BLUE CABLE COMES FROM THE ROUTER.	25
FIGURE 4-2: LAPTOP CONNECTIONS. THE ETHERNET CABLE COMES FROM THE SWITCH. THE USB3.0 COMES FROM THE PTZ CAMERA	25
FIGURE 4-3: PTZ CAMERA CONNECTIONS. THE ETHERNET CABLE COMES FROM THE SWITCH. THE USB3.0 IS CONNECTED TO THE LAPTOP.....	25
FIGURE 4-4: PC (LAPTOP) CONFIGURATION.....	26
FIGURE 4-5: HOW YOUR CONNECTION SHOULD BE AFTER CHANGING THE IP ADDRESS OF YOUR LAPTOP.....	27
FIGURE 4-6: MINRRAY CAMERA WEB ACCESS AND CONFIGURATION WINDOW.	27
FIGURE 4-7: EXAMPLE OF THE GLOBAL VARIABLE PYTHONPATH.....	28

FIGURE 4-8: THIS WHERE YOU SHOULD PUT A ‘1’ OR A ‘0’ DEPENDING IF YOU HAVE WEBCAM OR NOT. 28

FIGURE 4-9: HOW TO CREATE A NEW MULTIMEDIA SOURCE IN OBS STUDIO..... 29

FIGURE 4-10: THIS IS HOW YOU HAVE TO FILL IN THE TABLE WHEN YOU CREATE THE NEW MULTIMEDIA SOURCE 30

FIGURE 4-11: CONFIGURATION OF A LIVE EVENT ON YOUTUBE. THE YELLOW PART IS THE STREAM KEY..... 30

FIGURE 4-12: THIS IS HOW YOU MUST FILL UP THE GAPS TO LINK YOUR YOUTUBE LIVE EVENT WITH THE OBS STUDIO..... 31

FIGURE 4-13: BUTTON TO START STREAMING TO YOUTUBE. 31

LIST OF TABLES

TABLE 2-1: COMPARISON BETWEEN HAND-CRAFTED TRACKERS AND DEEP TRACKERS.....	9
TABLE 3-1: LIST OF COMMANDS FOR PAN, TILT, ZOOM AND FOCUS DRIVE.....	19
TABLE 4-1 : SOFTWARE REQUIREMENTS OF THE DEVELOPED SYSTEM.....	23
TABLE 4-2: HARDWARE REQUIREMENTS OF THE DEVELOPED SYSTEM.....	24

1 Introduction

1.1 Motivation

Just a few months ago, the world was brought to a standstill by the arrival of Covid-19. Quarantines began and people had to stop going to their jobs. Covid-19 has strongly affected people's behaviours and routines. This affected the student's community as well. Students had to stop going to class and started with online classes. In the best case, online classes are usually based on a few slides and the teacher commenting on them in the background. This is fine for a couple of classes, but if the online teaching is prolonged, it may not be very productive. So, this work comes from the Lab's urgent to offer students the opportunity to study from home or anywhere in the world in real time with quality classes.

To improve the quality of the classes the Lab thought of PTZ cameras. PTZ cameras are special cameras which can Pan 360 degrees, Tilt 90 degrees and Zoom automatically. The idea was to develop a framework to handle the PTZ camera movements to follow the lecturer movements to make it seem as if the student was in a face to face classroom (although obviously it will never be the same). So, with that idea in mind, the Lab, more specifically Gebrehiwot, A., developed a first version of the framework [1]. This framework worked very well but had some problems when it came to implementation. The two main problems were the complexity of the model and that it only worked under Ubuntu.

This project planned to solve both problems. Many lecturers don't have an Ubuntu partition on his laptop so the first thing will be porting the system to Windows so that a larger number of users can use it. Also, many lecturers don't have a GPU installed on his laptop so the second thing to do will be to develop a lightweight framework to track the professor. If these two objectives are achieved, we will be able to offer good quality online classes.

1.2 Objectives

Port the system to MS Windows.

The first objective of this Thesis is to port the existing system to MS Windows. We will try to run the actual tracking module in Windows Anaconda. This objective is important because some people do not have an Ubuntu partition on his PC, so to make that this algorithm could be used by as many people as possible we will port it to Windows.

Test lightweight CNN's for target tracking.

The actual tracking algorithm used to manage the PTZ camera is a state of the art one. Is a very good algorithm but it has the problem that we need a good GPU to run it. Not all the people have a good GPU, so we will try to change the tracking algorithm for a lightweight one to avoid this limitation.

Design and implement the camera operation module.

Once we have decided our lightweight algorithm, we will implement the camera operation module. This camera operation module will manage the movements of the PTZ camera, based on the movements of the lecturer.

1.3 Structure of the report

This report has the following chapters:

- Chapter 1: Motivation, objectives and organization of the thesis.
- Chapter 2: State of the art. In this section we will talk about the state of the art of the latest tracking algorithms, lightweight CNN's and PTZ cameras.
- Chapter 3: Development. In this section we will talk about how we have developed and achieved our three objectives. We will talk also about the previous work of Gebrehiwot, A.
- Chapter 4: Integration, testing and results. In this section we will show how to run the tracker in a laptop from scratch. We will also show how to make a Video Live Streaming (for the teaching activities) using the tracker.
- Chapter 5: Conclusion and future work.

2 State of the art

2.1 Video object tracking

2.1.1 What is Video object tracking?

Video object tracking is the task of estimating over time the state of one or more arbitrary objects of interest in video sequences. The target or object state x_k are variables describing its properties (e.g. location, shape appearance, structure...) for each k frame in the video [2]. We can see different types of target states in **Figure 2-1**. We usually locate the target with a bounding box (four variables for the object state). In video object tracking it is used also the recursive estimation. Recursive estimation means that once you have estimated the first state x_k , you will use this information to estimate the next state, x_{k+1} .

Video object tracking is widely used in lot of applications as autonomous driving, human-computer interaction, surveillance...

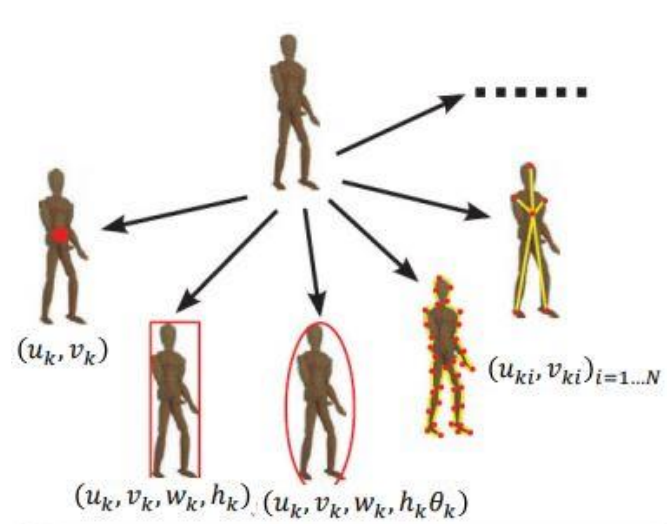


Figure 2-1 Different types of state in tracking.

Source: [2]

2.1.2 Differences with Video Object Detection

The terms "tracking" and "detection" are often used interchangeably. The purpose of detection is to find one or more objects in a given image, but the purpose of tracking is to find these objects throughout a video while keeping track of which object is which. To track an object, you must first give the tracking algorithm with an image of the object and this is either done by a detection algorithm (Detection-based trackers) or manually (Detection-free-trackers). [3]

Some people may think that we can perform an object tracking algorithm by making an object detection algorithm in each frame, but this is a very naïve way to perform. Tracking is necessary for many reasons:

- Detection is computationally expensive.
- Objects for which no detector has been trained can be tracked using detection-free trackers.
- With tracking we can maintain identities along the video.
- Changes in light, motion blur, change in scale, occlusions (when the subject is partially or totally obscured by another object for a period of time in the video), poor image quality... might all be managed with tracking. [3]

2.1.3 Single Object Tracking vs Multiple Object Tracking

In Single Object Tracking (SOT), the bounding box of the target in the first frame is given to the tracker. The tracker's purpose is to find the same target in all of the other frames. Because the first bounding box is provided, Single Object Trackers are classified as detection-free trackers. They should be able to track any item without any prior experience with it. Siamese network-based trackers and Correlation Filter-based trackers are the top performers for short-term tasks.

In Multiple Object Tracking (MOT) there are multiple objects to track. The tracking algorithm is supposed to figure out how many objects are in each frame and then maintain track of their IDs. MOT is a more challenging topic, and demonstrating an explicit class of algorithms that outperforms the rest is more challenging.

In this Thesis we will perform Single Object Tracking.

2.1.4 Challenges in Single Object Tracking

There are some challenges that could affect the performance of our Single Object Tracking algorithm. The main challenges are:

- **Object Modelling:** One of the major tasks of object modelling is to find an appropriate visual description that makes the object distinguished from other objects and background [4].
- **Changes in shape and appearance:** As the camera angle changes, the appearance of an object can change. During successive video frames, deformable objects such as humans can change their shape and appearance. The perspective effect, in which things farther away from the camera appear smaller than those closer to the camera, can also modify the appearance and shape.
- **Illumination changes:** Changes in lighting can have a significant impact on the look of an object. In an indoor (artificial light) environment, an object may appear differently than in an outside context (sun light). Even the time of day (morning, afternoon, evening) and the weather conditions (cloudy, sunny, etc.) can affect lighting.
- **Shadows and reflections:** For a shadow on the ground that behaves and appears like the target, some aspects such as motion, shape, and background are more sensitive. Reflections of moving objects on smooth surfaces might generate the same problem.

- **Occlusions:** Occlusion happens when one object is occluded by another item or when an object is occluded by a background component. The objects and their features become ambiguous under occlusion. Before and after the occlusion, the tracking algorithms must be able to determine the individuality of the objects involved in the occlusion. [4]

2.1.5 Main components in Single Object Tracking

Single Object Tracking mainly relates to:

- **Motion Problem** (i.e. prediction): identify a limited search region where the object is expected to be found with high probability. We can see that on **Figure 2-2**.



Figure 2-2: Representation of Motion Problem. Source: [2]

- **Matching Problem** (i.e. detection or location): identify the target state in the next frame within the designated search region. We can see that on **Figure 2-3**.

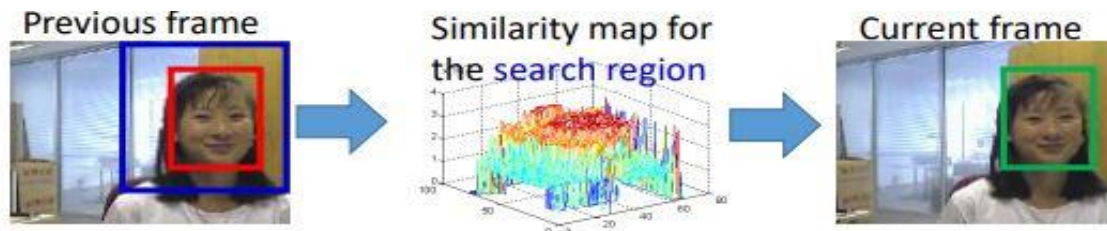


Figure 2-3: Representation of Matching Problem. Source: [2]

The typical pipeline that we can see on Video Tracking Algorithms is the one shown on **Figure 2-4**

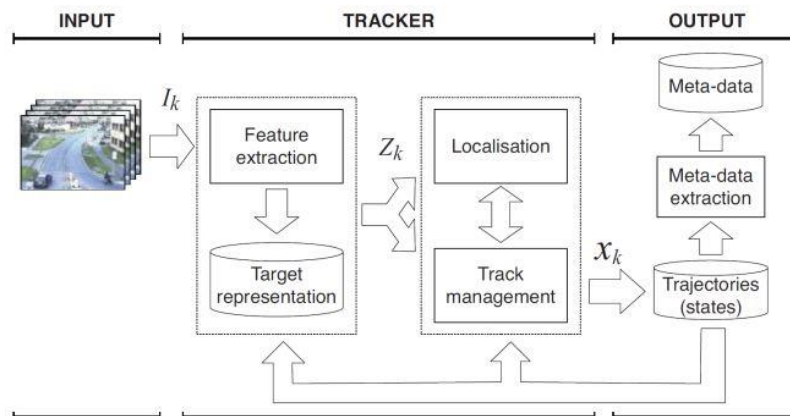


Figure 2-4: Typical Video Tracking Pipeline. Source: [2]

The main objective is to estimate the target state over the time (e.g. shape, position). In all tracking problems you have to make three choices. First of all, you have to select an object representation method for your target (the features that you will use to represent your target). Second, you have to select a searching process to generate candidate locations. Finally you have to select a similarity measure to find the best candidate (from the ones generated on the previous step) for your target.

All in this section is a summary extracted from [2].

2.1.6 Deep Learning Methods for tracking.

In this section we are going to see some Deep Learning Methods to perform Single Object Tracking. Deep Learning methods have been lately adopted in visual tracking and it have become a major achievement. In **Table 2-1** we can see a comparison of Hand-Crafted (HC) and deep trackers over the dataset VOT2017 [5]. The measurements are: expected average overlap (EAO), accuracy (A) and robustness (R). Only one tracker used hand-crafted features in VOT2020 [19].

	Tracker	Baseline			Realtime		
		EAO	A	R	EAO	A	R
HC trackers	CSRDCF	0.256	0.491	0.356	0.099	0.477	1.054
	STAPLE	0.169	0.530	0.688	0.170	0.530	0.688
	KCF	0.135	0.447	0.773	0.134	0.445	0.782
	SRDCF	0.119	0.490	0.974	0.058	0.377	1.999
	DSST	0.079	0.395	1.452	0.077	0.396	1.480
Deep trackers	CF2	0.286	0.509	0.281	0.059	0.339	1.723
	ECO	0.280	0.483	0.276	0.078	0.449	1.466
	CCOT	0.267	0.494	0.318	0.058	0.326	1.461
	SiameseFC	0.188	0.502	0.585	0.182	0.502	0.504
	MCPF	0.248	0.510	0.427	0.060	0.325	1.489

Table 2-1: Comparison between Hand-Crafted trackers and Deep trackers (extracted from [2])

2.1.6.1 Taxonomy of Deep Trackers.

Following [2] we can divide Deep Trackers according to their: Tracking strategy, Architecture, Network exploitation, Network training, Network objective and by their Network output.

Looking the tracking strategy we can find two types of trackers:

- **Detection-based trackers:** They perform tracking by detection. These trackers learn the possible positions of the target in the training phase. They have to re-detect the object at every frame and update the classifier.
- **Correlation-based trackers:** Correlation is applied between the observed data and the target model. These trackers operate in the frequency domain to manage computational cost.

Deep trackers can be also divided by their Network architecture. The most common architectures for deep trackers are:

- **CNN:** This was the most common architecture between 2015 and 2017. This architecture extracts the features better than handcrafted methods. It is focus on target modelling and matching.
- **Siamese:** It is the most popular architecture since 2017. It is based on modelling the target and matching. It is focused on achieving a trade-off between accuracy and speed. This architecture is the architecture that we are going to use in this Thesis.
- **RNN:** This architecture tries to model the target and features avoiding pre-trained CNN's. Due to its complexity it is limited on training.
- **GAN:** This architecture is focused on improving training modelling by enriching training samples. The drawback is that it is hard to train and evaluate.

We can divide also Deep Trackers by their Network Exploitation. There are two types:

- **Deep of-the-self features:** Is the preferred option. They use the network as a feature extractor. These networks have been previously trained on non-tracking still-image datasets. Features from different layers of the network are used to track the object.
- **Deep Custom Features:** This type of Network Exploitation is growing now. It is an end to end training. These trackers do not use pre-trained Networks. They make a specialised training for visual tracking and they are focused on designing, training and adaptation of tracking features.

There are also two types of training for Deep trackers:

- **Offline Training:** The tracker is previously trained on a large dataset to obtain generic target representations. The drawback is that it is limited to the 'classes' that we have on the Dataset, e.g., if you want to track an object that was not on your database, the Network would not extract a good representation of the target; but if your object was on the Database, this tracker would perform very well on real-time applications.
- **Online Training:** These trackers adapt their parameters according to the variation of the target appearance. They are prone to overfitting and may lead to drift. They have limitations for real time.

We can also divide the trackers by their Network Objective:

- **Classification-based objective function:** It is like a two-class (or binary) classification problem over object proposals. These trackers use a classification loss function.
- **Regression-based objective function:** Tracking is reformulated as optimizing L1 or L2 loss functions.

Finally, we can divide Deep Trackers by their Network output. The two more typical are:

- **Confidence map:** it is a probability density function on the output image, assigning each pixel of the new image a probability, which is the probability of the pixel colour occurring in the object in the previous image. [6]
- **Bounding Box:** Is an area defined by two points (x, y) and the width and high of the box.

This classification of Deep Trackers is extracted from [2].

2.1.6.2 Siamese Region Proposal Network

It is the architecture that we are going to follow in this thesis. As we can see on 2.1.6.1 Siamese based tracking is the most popular architecture since 2017. Siamese region proposal network (Siamese-RPN) is an end-to-end tracker trained off-line with large-scale of image pairs. Specifically, it consists of a Siamese subnetwork for feature extraction and a region proposal subnetwork including the classification branch and the regression branch. In the inference phase, the proposed framework is formulated as a local one-shot detection task. [7]

Siamese Network: It consists of two branches which are to be used as feature extractor for the two images that are passed as input (the template frame and the detection frame). In each of the branches will be the same Convolutional Neural Network to extract the features. Continuously the features extracted will be merged to generate a single output as we can see on **Figure 2-5**. Siamese Networks are common due to its accuracy and speed. It can be seen as a regression method using the bounding box predicted in the last frame as the only proposal. Siamese Networks are also very robust to the rapid movement of objects. The two branches share CNN parameters, so both images will perform a very similar transformation.

Region Proposal Network: Before RPN, traditional proposal extraction methods were time consuming. These networks are a type of network that arose to accelerate the process of region proposal. This was achieved through the enumeration of several anchors and the sharing of convolutional features. This type of networks is fast and achieves high quality results. RPN is able to extract more accurate proposals thanks to the supervision of the foreground-background classification and bounding box regression. The RPN used in this thesis consists of a pair-wise correlation section and a supervision section. The supervision section has two branches, one for foreground-background classification and the other for proposal regression. First, the pair-wise correlation section will divide the transformations obtained with the Siamese network in two. We will have a transformation of the template branch on the classification branch and another one on the regression branch, as we can see in **Figure 2-5**. The same will occur to the transformation obtained from the detection branch. The correlations between the two transformations will be performed in the two supervision branches. For this purpose, the convolution of the transformation of the template branch with the transformation of the detection branch will be performed both in the classification branch and in the regression branch. The output we obtain in the classification branch is an output with two thousand channels which represent the positive or negative activations of each anchor. In this branch we utilize a softmax loss function. The output of the regression branch consists of four thousand channels with dx, dy, dw ad dh of the distance between the anchor and the ground truth. [7]

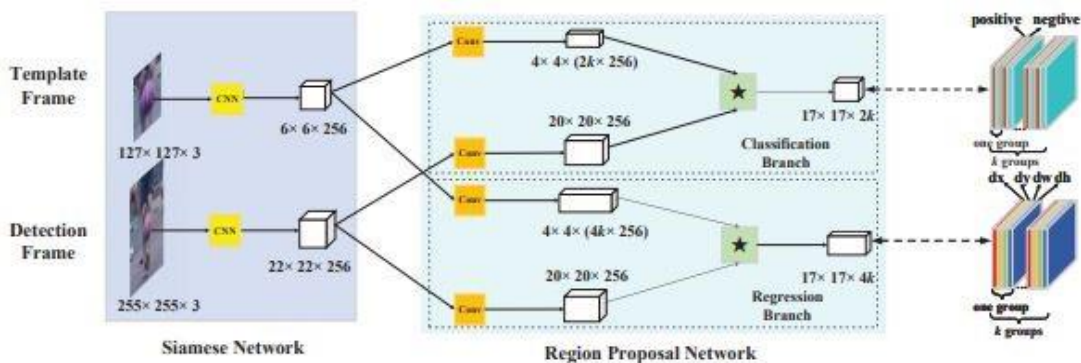


Figure 2-5: Architecture of SiamRPN. It is divided in two, at left the Siamese Network and at right the Region Proposal Network. Source: [7]

2.1.6.3 Lightweight CNN's

Deep CNN-based object tracking algorithms are more and more used in Artificial Intelligence (AI) applications. However, it still very difficult to deploy large CNNs architectures on small devices with limited hardware resources, because they consist of millions of parameters, which make them computationally very exhausting. Lightweight CNN architectures are proposed as a solution to make the deployment of deep neural networks on small devices feasible. [8]

In this section we are going to see the Lightweight Network used on our tracker. This Network is called MobileNet.

MobileNet is an efficient CNN architecture for mobile and embedded vision systems. It splits the convolution into a depth wise separable convolution followed by a pointwise convolution to build a lightweight deep neural network. Furthermore, it introduces two simple hyper-parameters that give us the possibility to build small and low latency models that can be easily matched to the design requirements for mobile and embedded vision applications. One of the hyper-parameters is the width multiplier that allows us to thin the number of channels, while the second hyper-parameter is the resolution multiplier that reduces the spatial dimensions of the feature maps. We can see its architecture on **Figure 2-6**. [8]

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

Figure 2-6 : Architecture of MobileNet. Source: [8]

2.2 PTZ Cameras

The term PTZ camera has two uses within the video security and surveillance products industry. First, it is an acronym for pan-tilt-zoom and can refer only to the characteristics of specific surveillance cameras. Second, "PTZ cameras" can also describe a whole category of self-tracking cameras, in which sound, motion - or a combination of these factors - triggers camera, focus and field-of-view changes.

PTZ cameras can rotate around two axes, one horizontal (360 degrees) and one vertical (90 degrees), as well as zoom in and out to focus on an area or object manually or

automatically. These cameras have been widely used on video-surveillance. Nowadays due to Covid, video conferences are increasing and to improve the quality of these, PTZ cameras are starting to be used, which give a more realistic experience. We can see a PTZ camera on **Figure 2-7**. [9]



Figure 2-7: PTZ Camera. Source:[9]

3 Development

3.1 Previous work

This thesis continues the work developed by a previous project. This previous project is titled ‘Real-Time Target Tracking to Position a Mobile Device’ [1]. The objective of this project was to obtain a fully portable system capable of automatically track the presenter (lecturer) and creates a far more appealing live presentation. [1]

The primary driving force of this previous project comes from the Lab’s urgent needs to offer students the opportunity to study from home or anywhere in the world in real-time. The main objective of this project was to build a prototype for real-time lecturer tracking with the aim of live-lecture video streaming. Therefore, this work focuses on developing a real-time active tracking framework to position a mobile camera precisely w.r.t the target of interest. The mobile camera was a PTZ camera, the same we have studied on section 2.2 and the target of interest was a lecturer. The tracking output was processed to control the PTZ mobile camera.

3.1.1 System Design

In this section we are going to see the system design of the previous work. For a moving target, tracking algorithms using a PTZ camera are a bit complex compared to using a static camera. The primary reason is, PTZ camera has pan, tilt, and zoom control, and it can rotate 360 degrees on its axis. Thus, each acquired frame captures a dynamic background scene in terms of position and location, which makes it difficult to relay in simple tracking algorithms that consider a static background. The proposed system fulfils a smooth camera movement like a professional human cameraman does. The proposed method is composed of three modules: acquisition module, tracking module and streaming module as we can see on **Figure 3-1**.

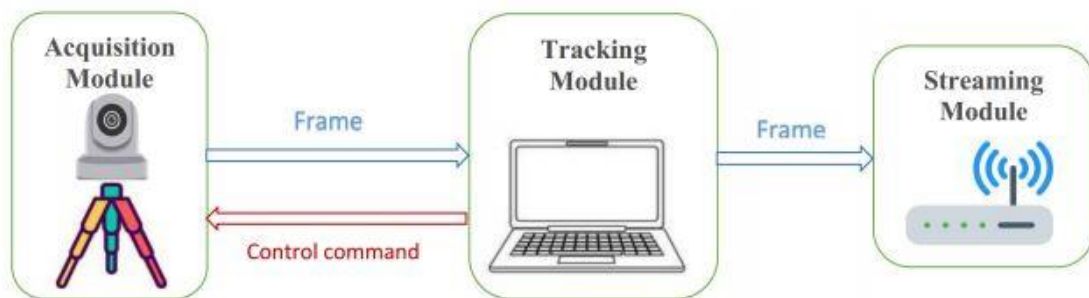


Figure 3-1: Previous proposed method: Hardware Set Up. Source: [1]

Acquisition Module: The acquisition module contains a PTZ camera (section 2.2). This module is responsible for capturing video frames and delivering them to the tracking module, and also receives back a control command, i.e., pan, tilt and zoom from the tracking module to accurately position the PTZ camera according to the tracked target (lecturer) position and orientation. [1]

Tracking Module: The tracking module contains a laptop with an on-board tracking algorithm. This module is responsible for processing each received frame from the acquisition module and initiate the tracking process. It tracks the target (Lecturer) in real-

time and sends back control signal in order to position the PTZ camera precisely w.r.t the location and orientation of the target in the PTZ camera field of view (FOV). The overall hardware configuration of the system is depicted in **Figure 3-2**. The laptop equipped with an on-board tracking algorithm utilizes the video frames from the PTZ camera and apply the tracking algorithm to locate the target (Lecturer) and then send a control commands, i.e., pan, tilt and zoom back to the PTZ camera aiming to accurately position the PTZ camera enabling to record a good quality lecture and stream the videos online over the Wi-Fi connection. [1].

Streaming Module: The streaming module is in charge of broadcasting the video to the student's devices. It is composed of the OBSstudio program and Youtube, plus the router and Ethernet cables needed to make connections. The signal from the PTZ camera is sent to the OBSstudio via RTSP. Then, from OBSstudio, the signal is sent to Youtube via RTMPS. Finally, students will connect to the Youtube link and will be able to watch the live broadcast. To see how to make a streaming from scratch go to section 4.3

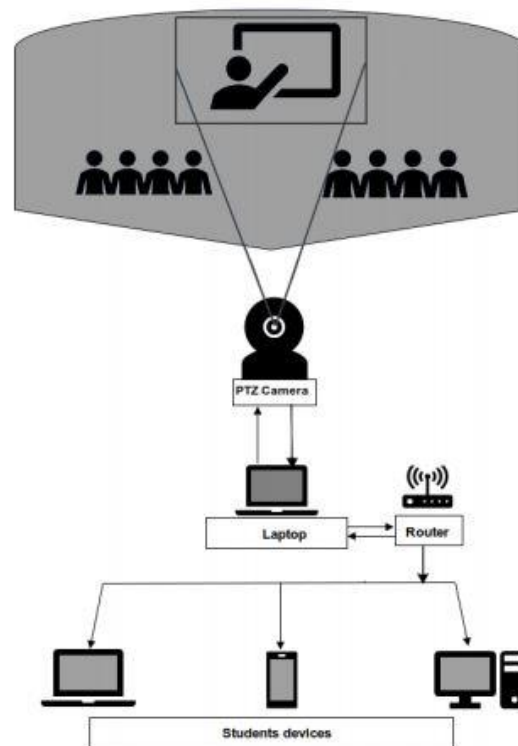


Figure 3-2: General overview of the proposed system. Source: [1]

3.1.2 Drawbacks of the system proposed

The system proposed in [1] has some drawbacks.

The first problem is that the system only works in Ubuntu. This is a drawback because some lecturers (the final users of the system) do not have an Ubuntu partition on his laptop.

For this reason, our first objective is port the system to MS Windows and try to run the tracking module there.

The second problem is about complexity. The tracking algorithm used is too complex to run it without GPU. Most of the laptops do not have a GPU. For this reason, our second objective is to test lightweight algorithms and CNN's for target tracking and try to run it without a GPU.

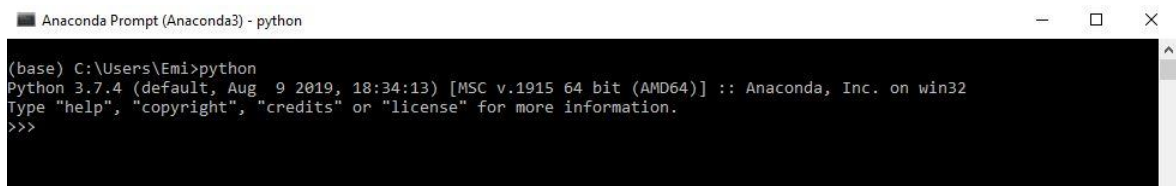
The third problem is that the system proposed does not use the zoom functionality of the PTZ camera. We want our tracking algorithm to look like a man anchoring the camera. For this reason, our third objective is to design and implement the camera operation module. This camera operation module will have the zoom functionality incorporated.

3.2 Objective 1: Port the system to MS Windows

In this section we are going to see step by step how we have implemented objective 1. As we have mentioned before, the tracking algorithm implemented by Gebrehiwot, A. in [1] only works on an Ubuntu partition. To solve that we are going to port the system to MS Windows. To do that we had following the next steps:

1. Download Anaconda for Windows: The first step is to download Anaconda on Windows. Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS [10]. With Anaconda we can run python code in a Windows partition in an easy way. To install Anaconda we have followed the steps included in [11] which are:

- Visit Anaconda Downloads page: www.Anaconda.com/downloads
- Select Windows from the three options.
- Download. You have to select 64 bit version or 32 bit version depending on your Windows version. With that step you should have downloaded the .exe installer.
- Open and run the .exe installer. You have to accept the license.
- Open the Anaconda Prompt. Anaconda is the Python distribution and the Anaconda Prompt is a command line shell (a program where you type in commands instead of using a mouse). To check if everything is good installed, open the Anaconda Prompt and type 'python'. You should see something like **Figure 3-3**. With the interpreter running, you will see a set of greater-than symbols '>>>' before the cursor. [11]



```
■ Anaconda Prompt (Anaconda3) - python
(base) C:\Users\Emi>python
Python 3.7.4 (default, Aug 9 2019, 18:34:13) [MSC v.1915 64 bit (AMD64)] :: Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 3-3 This is what it should appear when you type 'python' on the Anaconda prompt and the Anaconda is well installed

2. Install the Environment: The second step of this objective is to install the same environment that we have on Ubuntu. An environment is a tool that helps to keep dependencies required by different projects separate by creating isolated python virtual environments for them [12]. So, we are going to get all the packages installed on the Anaconda Ubuntu environment (the packages that allow us to run the Gebrehiwot, A. tracker) and we are going to install them on our new Windows Anaconda environment. To do that you have to follow the next steps:

- Activate the Ubuntu environment of Gebrehiwot, A. tracking algorithm.
- Create a .yaml file. In this file we will have all the packages installed on Awed's environment. To create this file you have to run the next command on the Anaconda Prompt (once that the environment is activated): `'conda env export > environment.yaml'`. All the packages that we need will be on the environment.yaml file.
- Now we have to install all this packages on our Anaconda Windows Environment. To do that we open the Anaconda Prompt in our Windows partition and we create a new environment from the file generated in the previous step. To do that we run on the prompt the next command: `'conda env create -f environment.yaml'`. Now we have the same environment / packages on the two partitions. [12]

3. Install and run the tracker: This is the final step. Once you have all the packages that you need installed you have to port the tracker to Windows. This is very easy. You only have to take and copy all the files that we have on our tracking algorithm project on Ubuntu and paste it in the same way to the new project created in Windows. After that you can run the tracker on Windows.

3.3 Objective 2: Test lightweight CNN's for target tracking.

In this section we are going to see how we have implemented objective 2. As we have mentioned before, the Gebrehiwot, A. tracker is a state of the art algorithm, which works very well but with the drawback that a GPU is needed to run it. Most of the laptops do not have a GPU installed. As our algorithm will be run on a laptop most of the times, we need to make it simpler to run it on a CPU.

The tracking algorithm used in Gebrehiwot, A. tracker is called SiamMask [13]. This algorithm is based on Siamese Networks, the most popular architecture for deep trackers since 2017, as we mentioned on section 2.1.6.1. SiamMask it is an improvement/evolution of the SiamRPN proposed on section 2.1.6.2. SiamMask unlike existing tracking methods that rely on low-fidelity object representations, argue the importance of producing per-frame binary segmentation masks. Binary segmentation masks are images of pixels, where the pixels of the target are '1' and the background pixels are '0'. To this aim SiamMask show that, besides similarity scores and bounding box coordinates, it is possible for each response of a candidate window (RoW) of a fully convolutional Siamese network to also encode the information necessary to produce a pixel-wise binary mask. This can be achieved by extending existing Siamese tracker with an extra branch and loss [13], as we can see on **Figure 3-4**.

The SiamMask architecture is a State of the art method in terms of accuracy. The problem for us, is that is too complex to perform without GPU. The thing is that our problem is not very complex or difficult. We just need to implement a Single Object Tracker, where the target will be in a controlled environment. Probably we will not have complex challenges as occlusions or distractors (aka similar objects) so we will not need a State of the art tracker. It is better to use a simpler one that allows us to have de same performance but with much less parameters.

For this reason we have selected the SiamRPN tracker (section 2.1.6.2). To implement this tracker we have follow the instructions from [14]. The steps are as follows:

- Step 1: Install the framework. To do that you have to run the next commands on your Anaconda Prompt:


```

      'cd ~'
      'git clone https://github.com/STVIR/pysot.git'
      'export PYTHONPATH=~/youruser/pysot:$PYTHONPATH'
      'bash install.sh /opt/anaconda3.7 pysot'
      
```
- Step 2: Get the tracker Model. In this step we download the model zoo of our tracker model. To do that you have to run the next commands on your Anaconda Prompt:


```

      'wget http://wwwvpu.eps.uam.es/~jcs/DLVSP/pysot_nets/siamrpn_mobilev2_l234_dwxcorr/siamrpn_mobilev2_l234_dwxcorr.pth'
      'mv siamrpn_alex_dwxcorr.pth ~/pysot/experiments/siamrpn_mobilev2_l234_dwxcorr/'
      
```

In this step we have chosen MobileNetV2 (section 2.1.6.3) as the tracker model because it is a lightweight CNN and it has less parameters than other CNN's. This will help us to make a simpler module for tracking which is one of the objectives in this thesis.

Once you have done these two steps, you will have installed the SiamRPN tracker. If you want to test the tracker with the webcam of your laptop you can run the next command on the Anaconda Prompt:

```

python tools/demo_ptz_zoom.py \
--config experiments/siamrpn_mobilev2_l234_dwxcorr/config.yaml \
--snapshot experiments/siamrpn_mobilev2_l234_dwxcorr/model.pth

```

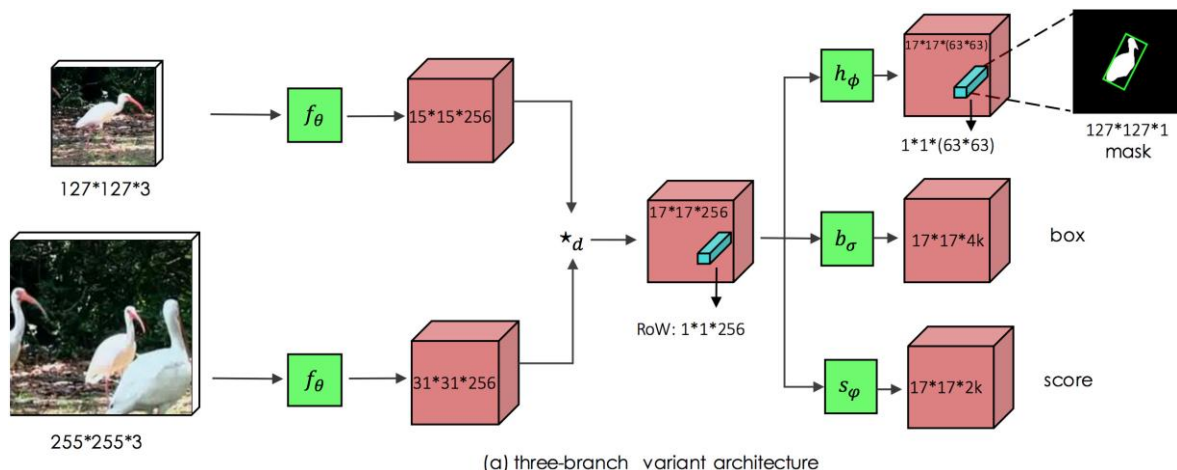


Figure 3-4: SiamMask Architecture. We can see the added branch to make segmentation mask (the first one starting by the top). This is the tracker used in Awed's algorithm. Source: [13]

3.4 Objective 3: Design and implement the camera operation module.

In this section we are going to see how we have implemented objective 3. Once we have our tracking algorithm installed, we will implement the camera operation module. This camera operation module will be in charge of the PTZ camera movements. The PTZ camera must follow the movements of the lecturer as we have seen on section 3.1.1.

3.4.1 Communication between PC (laptop) and PTZ camera.

First of all, we are going to see how we can handle the movements of the PTZ camera. To access the camera from a PC/laptop one can use the USB 3.0 cable or the Ethernet cable, both of them serve different access to the camera. With the USB 3.0 it is possible to read the video frames from the camera and using the Ethernet cable it is possible to access the Camera configuration setup from any browser and also control the PTZ motion of the camera. During the project the Camera was assigned an IP address of 192.168.1.163, anyone who wants to access the camera for the first time, they have to configure the PC (laptop) that they are using to be in the same network with the PTZ Camera IP address.

The PTZ camera allows to be remotely controlled via IP by sending VISCA commands. In this project a socket programming with Pan, Tilt and Zoom commands has been written aiming to control the Pan, Tilt and Zoom motion of the camera. The Pan, Tilt and Zoom commands are sent over the dedicated VISCA IP address: 192.168.1.163 and port: 1259. The specific code written to handle the camera movement can be found inside pypot/tools/minnary_ptz_control_ZOOM.py. The specific VISCA commands used to control the camera are shown in Table 3-1. [1]

Command	Function	VISCA Command Package	Note
Pan Tilt and Zoom	Up	81 01 06 01 VV WW 03 01 FF	VV: Pan speed
	Down	81 01 06 01 VV WW 03 02 FF	0x01 (low speed)
	Left	81 01 06 01 VV WW 01 03 FF	to
	Right	81 01 06 01 VV WW 02 03 FF	0x18 (high speed)
	UpLeft	81 01 06 01 VV WW 01 01 FF	
	UpRight	81 01 06 01 VV WW 02 01 FF	WW: Tilt speed
	DownLeft	81 01 06 01 VV WW 01 02 FF	0x01 (low speed)
	DownRight	81 01 06 01 VV WW 02 02 FF	to
	Stop	81 01 06 01 VV WW 03 03 FF	0x14 (high speed)
	Absolute Position	81 01 06 02 VV WW 0Y 0Y 0Y 0Y 0Z 0Z 0Z 0Z FF	YYYY: Pan Position ZZZZ: Tilt Position
	Relative Position	81 01 06 03 VV WW 0Y 0Y 0Y 0Y 0Z 0Z 0Z 0Z FF	YYYY: Pan Position ZZZZ: Tilt Position
	Home	81 01 06 04 FF	
	Reset	81 01 06 05 FF	
	Zoom in	81 01 04 07 02 FF	
	Zoom out	81 01 04 07 03 FF	
Zoom stop	81 01 04 07 00 FF		
Focus in	81 01 04 08 02 FF		
Focus stop	81 01 04 08 00 FF		

Table 3-1: List of commands for Pan, Tilt, Zoom and Focus drive. Source: [1]

3.4.2 Implementation of the camera operation module.

In this section we are going to see the ‘system of rules’ that we have implemented to handle the movements of the PTZ camera based on the lecturer movements. Up to this point we have our tracking algorithm installed on our laptop and a PTZ camera connected to it. The PTZ camera and the laptop can communicate between them sending VISCA commands through a socket as we can see on section 3.4.1.

The first thing that our algorithm does is to reset the camera to the Home position. The Home position is the intermediate position of both horizontal and vertical. We reset the Zoom position too. This is done to always start in the same position, not in the position in which the last user of the camera left it.

The next thing is to select the target. This can be done in two ways: manually or automatically. If the user selects “manual target selection”, a pop-up window will appear and the user will have to mark the target with a bounding box as we can see on **Figure 3-5**. If the user selects “automatic target selection”, the target will be selected by the algorithm. To do that we pass the frame to an object detector. After that we apply a filter to keep only the ‘person’ object. The bounding box generated by the object detector, will be the bounding box to track in the rest of the algorithm. If the object detector detects more than 1 person, a pop-up window will appear and the user will select one of them. It works better with the manual target selection.

After that a Zoom will be done to improve the camera framing. If the room is big the Zoom will be larger than in a small room.

Once we have the target selected and the zoom applied, we start tracking. At every frame of the video we will have the position of the target in a bounding box. The bounding box will be represented by the x and y coordinates plus the width and height as we can see on **Figure 2-1**. To handle the movements of the PTZ camera we will focus on the movement of the bounding box. The first thing we are going to do is to calculate the centroid of the bounding box (cX and cY will be the coordinates of the centroid). With the coordinates of the centroid we will calculate the location of the target in the frame (lX and lY). After that we define the target area by dividing the frame into 6 by 5 grid regions (rX and rY). Then we define the rules:

- If $(lX > rX)$ we pan to the right.
- If $(lX < -rX)$ we pan to the left.
- If $(lY > rY)$ we tilt down.
- If $(lY < -rY)$ we tilt up.

These rules are very simple. They are based on a grid region and when the target changes the grid in where it is, we move the camera. We can see this ‘system of rules’ on **Figure 3-6**.

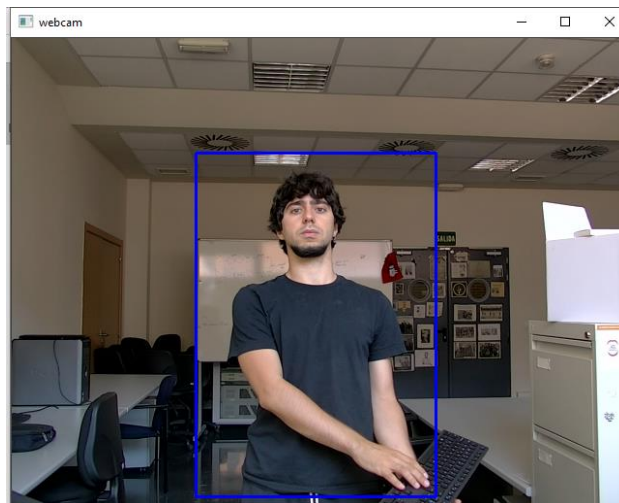


Figure 3-5: Example of Manual Target selection

```
#print(frame.shape())
fw = 640
fh = 480

#ahora vamos a calcular el centroide de la
cX = bbox[0] + (bbox[2]/2)
cY = bbox[1] + (bbox[3]/2)
#print('Centroides cX ' + str(cX))
#print('Centroides cY ' + str(cY))

# calculate the location of target in the
lX = round(float(cX - fw / 2)) # (en el e
lY = round(float(cY - fh / 2))

# define target area by dividing the frame
rX = fw / 6.0
rY = fh / 5.0

# PTZ camera control
if (lX > rX):
    data = bytes.fromhex(pan_R_re_pos)
    s.sendall(data) #

elif (lX < -rX):
    data = bytes.fromhex(pan_L_re_pos)
    s.sendall(data)

elif (lY > rY):
    data = bytes.fromhex(tilt_D_re_pos)
    s.sendall(data)

elif (lY < -rY):
    data = bytes.fromhex(tilt_U_re_pos)
    s.sendall(data)
```

Figure 3-6: System of rules to handle the movements of the PTZ camera.

3.4.2.1 Extension of the camera operation module

Once the algorithm was developed, we thought of an extension for it. It occurred to us that we could make an improved version of the tracker for those users who had a GPU in their laptops. (It could be used also by users with a good CPU, such as Intel i7 8700 or higher)

This extension is based in the zoom functionality. Up to this point, the rule system we were using only handled the Pan and Tilt functionalities. The new rule system includes rules that handle Zoom in and Zoom out automatically, based on the size (T_size) of the selected bounding box. T_size is calculated by subtracting y_max and y_min of the bounding box. We included this two rules to the system:

- If (T_size > 2.5*rY) we make a Zoom Out.
- If (T_size < 2*rY) we make a Zoom in.

We can see that on Figure 3-7. To see a comparison between the two versions implemented go to section 4.4.

```
#-----  
elif (T_size > 2.5 * rY): #original=4 #con 2.  
    if cuenta_zoom > 0:  
        data = bytes.fromhex(Z_out) # EN REA  
        s.sendall(data)  
        time.sleep(0.3) # con 0.3 va guay  
        data = bytes.fromhex(Z_stop)  
        s.sendall(data)  
        cuenta_zoom = cuenta_zoom - 1  
    else:  
        cuenta_zoom = cuenta_zoom  
  
    #data = bytes.fromhex(Z_stop)  
    #s.sendall(data)  
    # print('zoom out000000',f)  
elif (T_size < 2 * rY):  
    if cuenta_zoom >= 0 and cuenta_zoom <= 5:  
        data = bytes.fromhex(Z_in) # EN REAL  
        s.sendall(data)  
        time.sleep(0.3) # con 0.3 va guay  
        data = bytes.fromhex(Z_stop)  
        s.sendall(data)  
        cuenta_zoom = cuenta_zoom + 1  
    else:  
        cuenta_zoom = cuenta_zoom
```

Figure 3-7. Zoom functionality included in the rule system

4 Integration, testing and results

4.1 System requirements

In this section we will describe the software and hardware needed to run the system.

Software requirements: The software infrastructure employed during the project is completely written in python; in particular the Python version 3.7.10. The software system has a set of components; the majority of them are related to the fields of computer vision and deep learning. The most prominent unities are shown in **Table 4-1**. All necessary libraries can be found on the ENV_pysot_zoom_cpu.yml file located inside the pysot directory. [1].

For the extended version of the tracker we have developed other environment. This environment has the same libraries as the other environment but with the GPU versions. We have remove the gluoncv library because it causes conflicts with other libraries. For this reason the extended version of the tracker doesn't have automatic target detection. All necessary libraries can be found on ENV_pysot_zoom_gpu.yml file located inside the pysot directory.

	Type	Version	Description
Python	Programming lenguaje	3.7.10	A general-purpose and high-level programming language, used for developing Data Science including machine learning, data analysis, and data visualization
PyTorch	Deep Learning framework	1.8.1	Open-source Machine Learning framework based on the Torch library, supports a tensor computation (like NumPy) with strong GPU acceleration.
TorchVision	Part of the Pytorch Framework	0.9.1	The torchvision package consists of popular datasets, model architectures, and common image transformations for computer vision. [17]
OpenCV	CV library	4.5.1.48	Provides a common infrastructure for computer vision applications.
Numpy	Python library	1.19.5	Fundamental library for Python with packages for scientific computing, multi-dimensional arrays, with high-level mathematical functions.
Keras	Deep Learning framework	2.3.1	Keras is an open-source software library that provides a Python interface for artificial neural networks [15]
Windows	OS	10	It is a desktop operating system developed by Microsoft. [16]
Anaconda	Environment	4.9.2	Open-source distribution of the Python programming languages for scientific computing, that aims to simplify package management and deployment.

Table 4-1 : Software requirements of the developed system

Hardware requirements: The software system has been developed on a laptop equipped with Intel Core i7-10750H with 2.60GHz CPU, Nvidia GeForce GTX 2060 8GB, GDDR5, with 16GB RAM. The PTZ camera described on section 2.2 with Pan, Tilt and Zoom functionalities has been used as video source module. The communication between camera and laptop through IP address has been described on section 3.4.1. Detailed description regarding the PTZ camera and used hardware can be found in **Table 4-2**.

	Type	Description
Computer model	Laptop PC	Laptop with on-board GPU is used for processing the tracking algorithm and to position the PTZ camera
Camera model	UV950A-12-U3 Minrray PTZ camera	A portable Minrray PTZ camera is used
Camera control	PTZ control	Onboard with pan, tilt and zoom control functionality
Camera PTZ communication	IP address	Ethernet cable is used for control data communication between the Camera and the computer or router
Camera data communication	Serial port	USB cable is used for data communication between the Camera and the computer
Camera power source	DC	USB cable is used for data communication between the Camera and the computer

Table 4-2: Hardware requirements of the developed system

4.2 Setting up the system

In this section we are going to see how to run the system from scratch. We will divide this section in two. On part will be to set up the hardware and the other to set up the software.

4.2.1 Hardware Set up

As we have seen on section 3.4.1 the communication between laptop and PTZ camera will be with the USB3.0 and the Ethernet cables. We will use a switch and three Ethernet cables to be in the same subnetwork. The signal entering port 1 (coming from a router) is divided to the ports 2 and 3 of the switch. The Ethernet cable coming out of port 2 goes to the laptop and the Ethernet cable coming out of port 3 goes to the PTZ camera. The USB 3.0 connects the PTZ camera with the laptop. We can see this connections on **Figure 4-1**, **Figure 4-2** and **Figure 4-3**.



Figure 4-1: Switch connections. The Ethernet blue cable divide the internet signal in the two white cables. The white cables went one to the laptop and the other to the PTZ camera. The blue cable comes from the router.



Figure 4-2: Laptop connections. The Ethernet cable comes from the switch. The USB3.0 comes from the PTZ camera



Figure 4-3: PTZ camera connections. The Ethernet cable comes from the switch. The USB3.0 is connected to the laptop.

4.2.2 Software Set up

Once you have all the connections ready, we can start with the software.

The first thing to do is install Anaconda. We can see how to download Anaconda on section 3.2.

After that, as we have said in section 3.4.1, during the project the Camera was assigned an IP address of 192.168.1.163, anyone who wants to access the camera for the first time, they have to configure the PC (laptop) that they are using to be in the same network with the PTZ Camera IP address. This can be done by manually changing the IP address of the PC (laptop), to a similar network address (e.g 192.168.1.26) as shown in **Figure 4-4**. Even if you are connected by Ethernet cable, you should still be connected to the internet via Wi-Fi. The Ethernet cable is used to move the camera, not to have internet connection. We can see that on **Figure 4-5**. Then by opening a browser and putting the camera IP address 192.168.1.163, it will present a screen which asks for "Username" and "Password" as shown in **Figure 4-6**, please fill admin on both the "Username" and "Password", ones logged in it is possible to modification any configuration. [1]

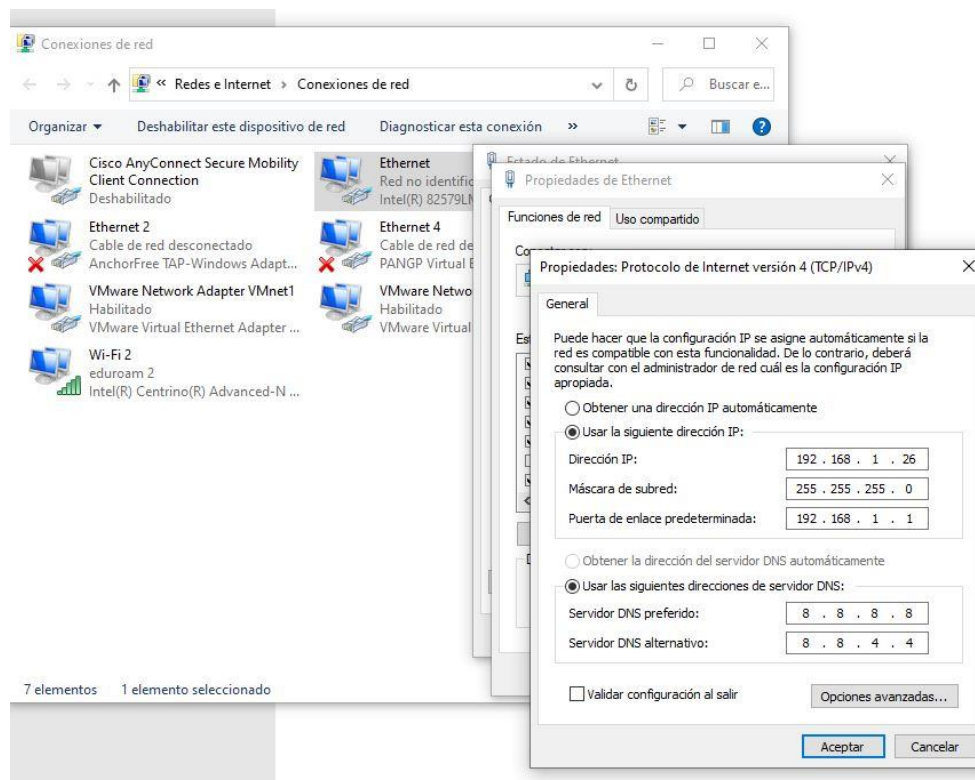


Figure 4-4: Pc (laptop) configuration

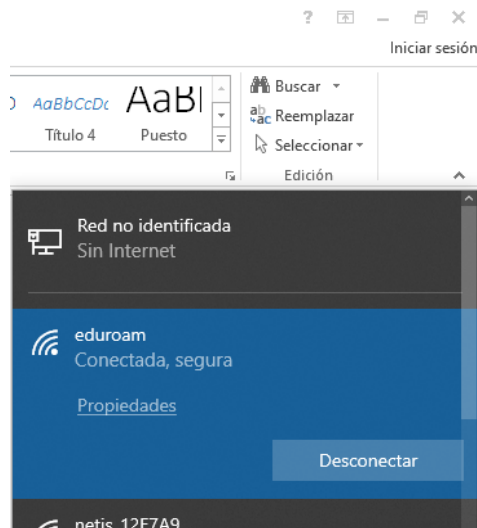


Figure 4-5: How your connection should be after changing the IP address of your laptop.

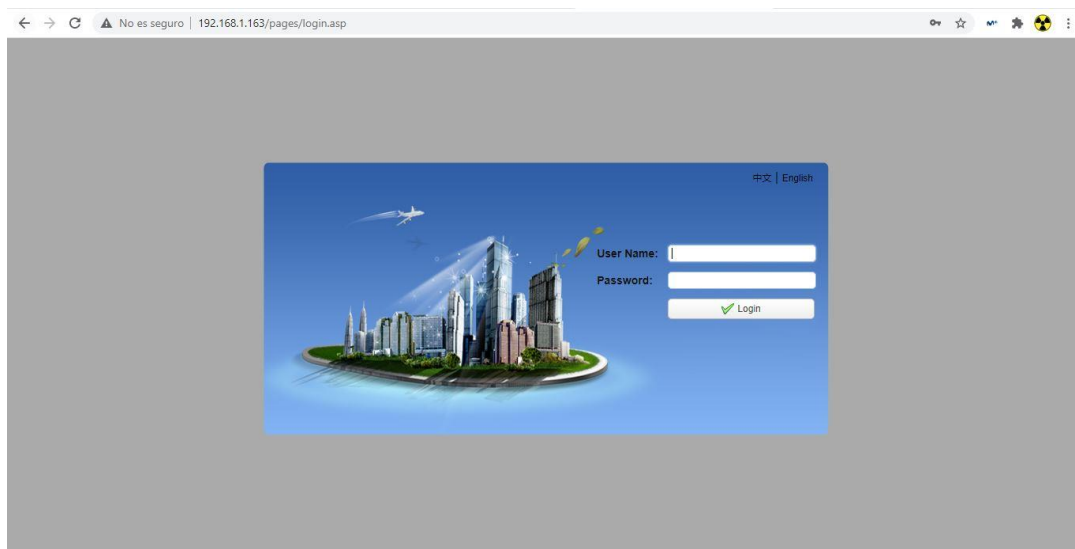


Figure 4-6: Minrray Camera web access and configuration window.

Once you have changed the IP address of your laptop, you should create the Conda Environment with all the libraries needed to run the algorithm. To do this you should follow this steps:

- Go to the downloaded zip and extract the pysot folder to you Desktop.
- Open the pysot folder.
- Open Anaconda Prompt and run the next command:
`'conda env create -f ENV_pysot_zoom_cpu.yml'`
- Activate the Environment:
`'conda activate pysot_cpu'`

- Export the PYTHONPATH.
To do that, you have to create a global variable: *Control panel > Click on the Advanced system settings link and then click Environment Variables.*

You have to create a new variable named PYTHONPATH. This variable should be the path to your pysot project (the folder where are all the files), as we can see on **Figure 4-7**.

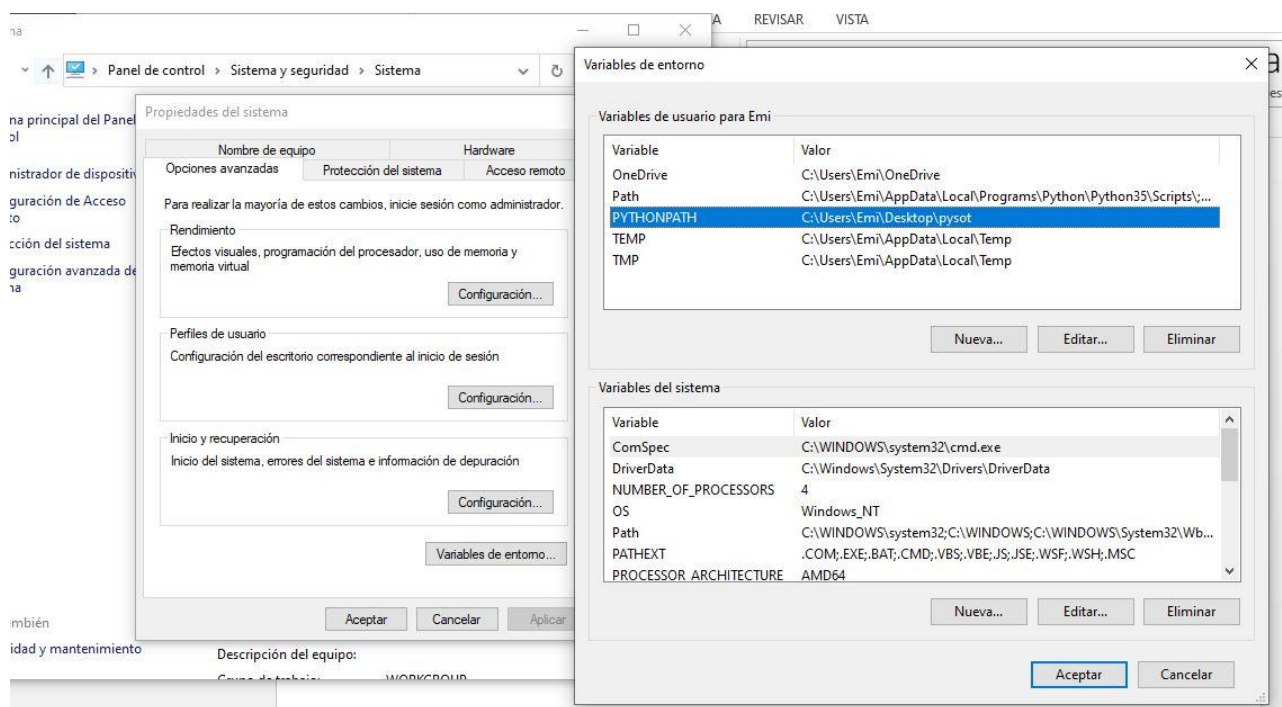


Figure 4-7: Example of the global variable PYTHONPATH.

Now is time to run the tracker. To do that you must follow these steps:

- Check if your laptop has a webcam installed. This is to prevent the software from analyzing the images from the webcam instead of the images it receives via USB from the PTZ camera.
Once you have checked it go to the file: `pysot/tools/demo_ptz_zoom.py` line 41.
If you have a webcam installed put a '1' on the function.
If you don't have a webcam installed put a '0'.
We can see that on Figure 4-8.

```
def get_frames(video_name):
    if not video_name:
        cap = cv2.VideoCapture(0)
```

Figure 4-8: This where you should put a '1' or a '0' depending if you have webcam or not.

- Once you have checked that, you can run the tracker with the next command:


```
python tools/demo_ptz_zoom.py --snapshot
./experiments/siamrpn_mobilev2_l234_dwxcorr/siamrpn_mobilev2_l
234_dwxcorr.pth --config
./experiments/siamrpn_mobilev2_l234_dwxcorr/config.yaml'
```
- Now you have to follow the instructions that appears in the command line.

4.3 Video Live Streaming

In this section we are going to see step by step how to make a Video Live Streaming once that you have the tracker installed. This could be useful to stream the lectures.

First of all we will need to install a program called OBS studio. OBS Studio is a free and open source software for video recording and live streaming. To download this program you must follow the next steps:

- Go to www.obsproject.com.
- Select Windows from the Home page.
- Open the downloaded file (.exe).
- Follow the steps of the .exe and agree the license. (as a common program).

Once you have OBS Studio installed we will link it to the PTZ camera. We will do this Via RTSP (point to point to a physical decoder). To do that you must follow the next steps:

- Open OBS Studio.
- Create a new multimedia source as in **Figure 4-9**.

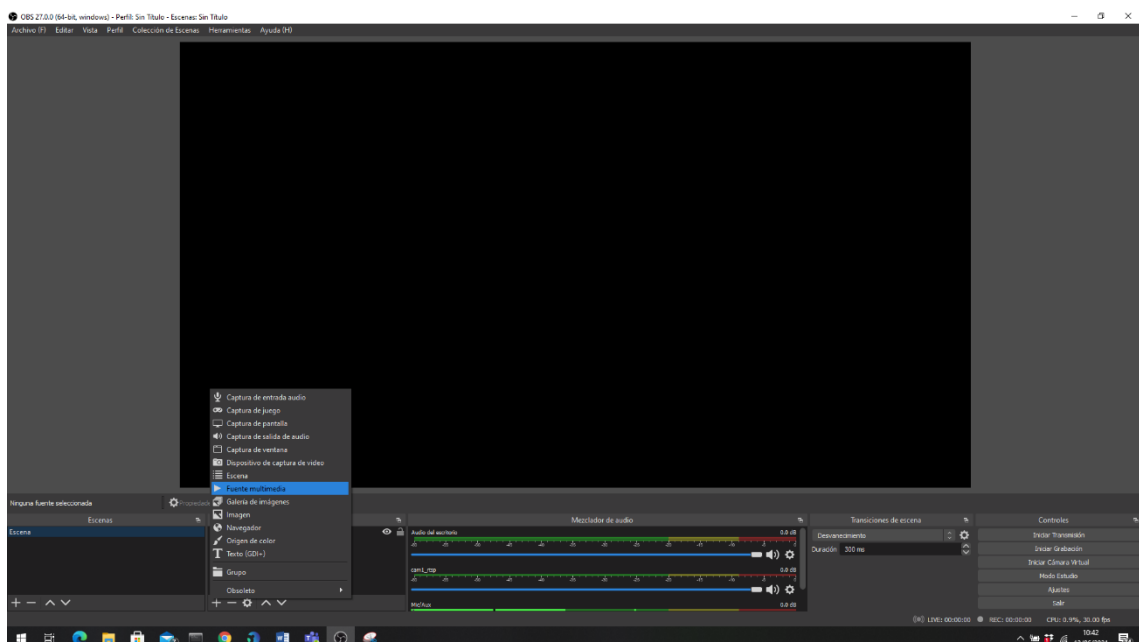


Figure 4-9: How to create a new multimedia source in OBS Studio.

- A pop-up window is open and you must select ‘create new’
- After that a new pop-up window is open. You must fill it with the IP of the camera (192.168.1.163) followed by the Stream Name (live/av0). You must fill it exactly as we can see on **Figure 4-10**. Then you press ‘ok’ and the PTZ camera will be linked with OBS Studio.

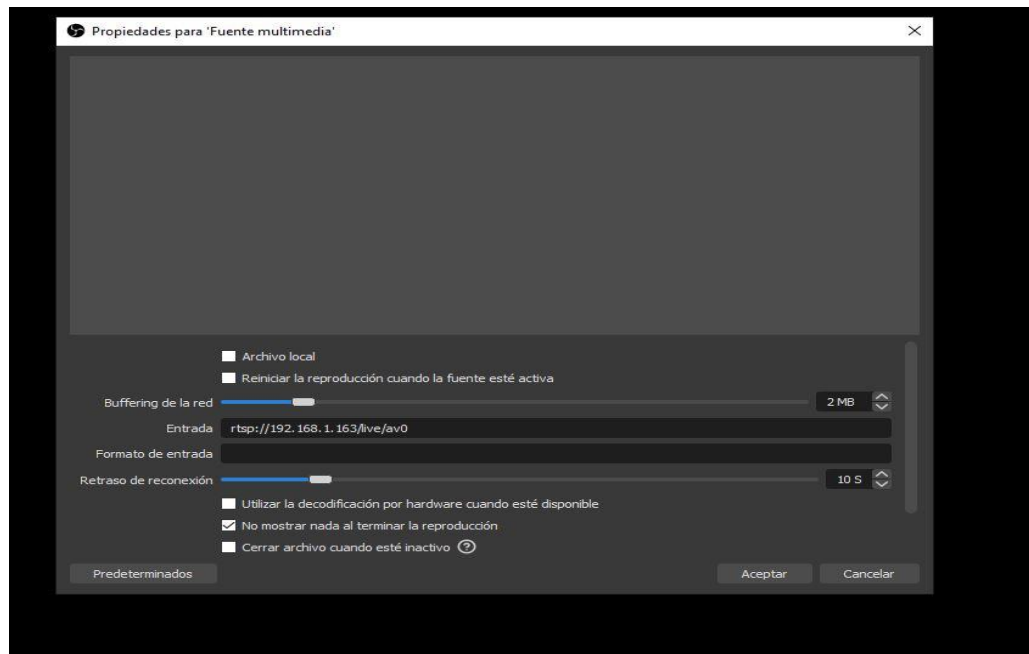


Figure 4-10: This is how you have to fill in the table when you create the new multimedia source

Now we have the PTZ camera and the OBS linked. The next step is to link YouTube with OBS to start streaming. You must follow the next steps:

- Create a live event on YouTube as it can be shown on **Figure 4-11**.
- Copy the Stream Key (the highlighted part in yellow on **Figure 4-11**).

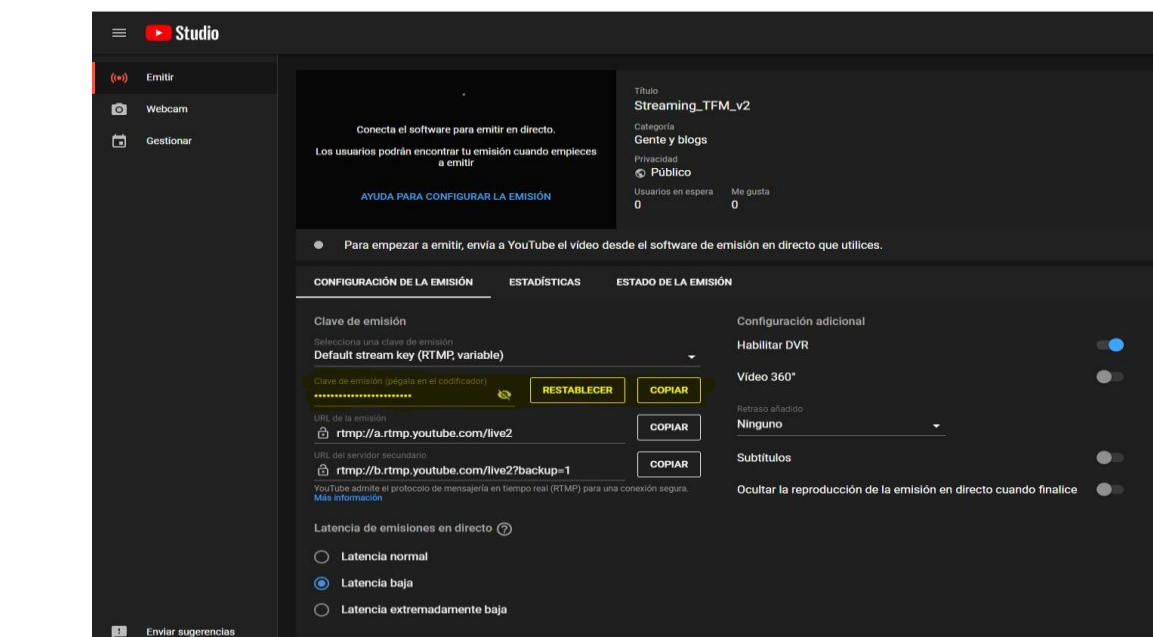


Figure 4-11: Configuration of a Live Event on YouTube. The yellow part is the Stream Key.

- The next step is done in OBS Studio. Go to settings and then to stream (inside settings). You must fill up the table as shown in **Figure 4-12**. You must paste your Stream Key in the Stream key gap.

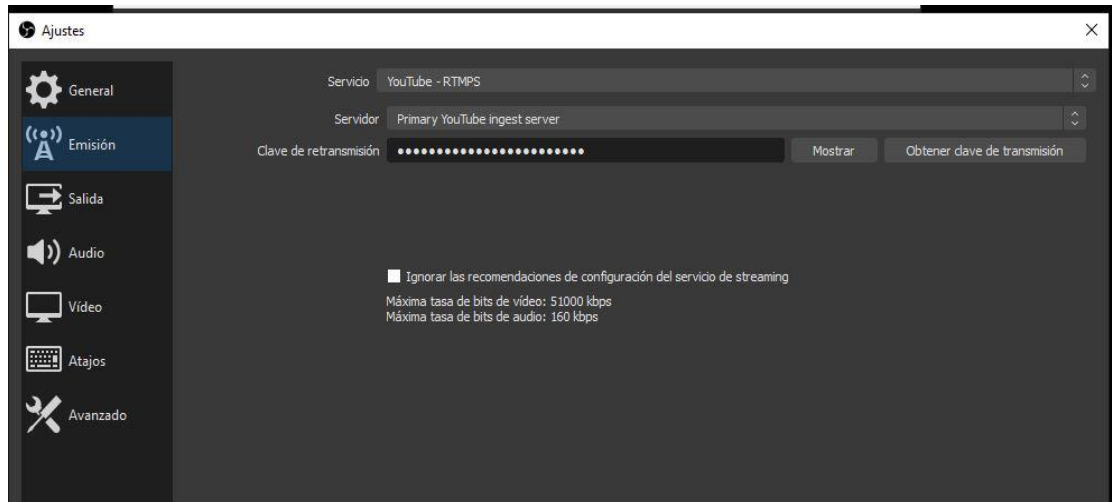


Figure 4-12: This is how you must fill up the gaps to link your YouTube live event with the OBS Studio.

- After that you can start streaming clicking the button shown on **Figure 4-13**.

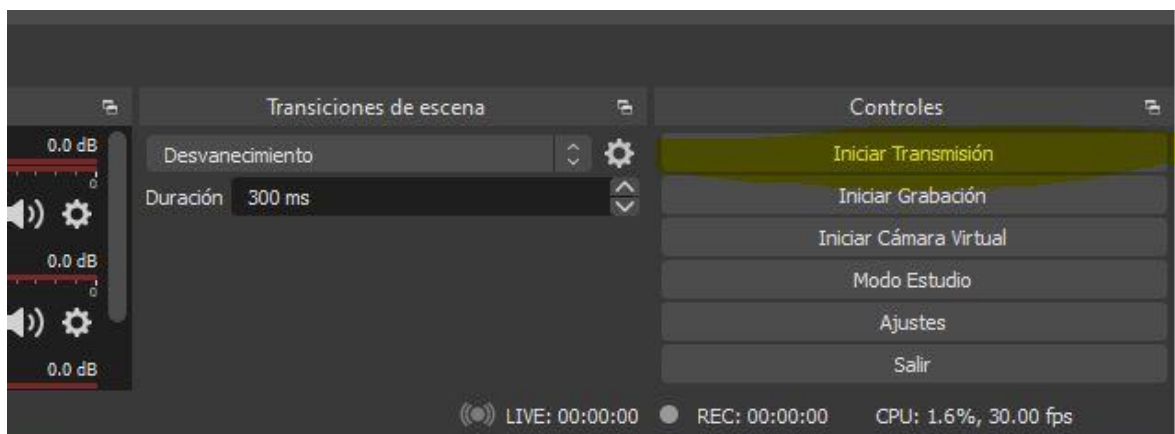


Figure 4-13: Button to start streaming to YouTube.

- After that the signal will arrive to YouTube via RTMPS and you will be doing a video live streaming on YouTube.

4.4 Demos

In this section we are going to leave the link to two videos in which you can see a demo of the project implemented.

Demo with CPU: <https://youtu.be/aJAs-dAxKSQ>

Demo with GPU: <https://youtu.be/KzUjNTt7ZpY>

5 Conclusions and future work

5.1 Conclusions

Our life has changed radically in the last year. Due to the spread of covid-19 many jobs were done remotely and many students started online teaching. The focus of this thesis will be online teaching, more exactly, in the improvement of students online classes.

For this purpose, the university purchased the PTZ cameras. These are special cameras that can pan 360 degrees, tilt 90 degrees and zoom automatically. The goal was to make online classes of higher quality and make it seem as if the student was actually in class. So, with this in mind Gebrehiwot, A. developed a framework that handles the movement of the PTZ camera based on lecturer's movements [17]. The framework developed was so good and uses a state of the art tracker to follow the teacher movements but it has two main problems. The first problem is that the framework only works under Ubuntu and the second problem is that the framework is too complex to run it without a GPU. So, this Thesis continues the work developed trying to fix these two problems. In addition to this, it will also focus on improving the camera operation module.

The first objective we went for was to port the system to Windows. If the framework only works under Ubuntu is a problem. It is a problem because many teachers, who are the end users of the system, do not have an Ubuntu partition installed on their laptops. To tackle this problem we downloaded Anaconda for Windows. Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment [17]. With Anaconda we can run python code in a Windows partition in an easy way. The main problem to achieve this goal were some libraries that worked on Ubuntu but not on Windows. To solve this, we either looked for a previous version that worked on Windows and was compatible with the rest of the libraries versions or we did not install these libraries. Fortunately this did not happen too much and the libraries that did not work were not necessary for the main function of the framework. So in general this was not a very difficult objective to achieve.

The second objective we went for was to make lightweight framework. The framework that Awed implemented was based on SiamMask [13]. SiamMask is a State of The Art algorithm for tracking in Real Time. This algorithm worked very well tracking the movements of the lecturer. The problem was that running this algorithm required a lot of computational power because it was very complex and had many parameters. Basically you needed a GPU in your laptop for the algorithm to work properly. This is a problem because as with Ubuntu partition, most lecturers do not have a GPU in their laptops. To solve this we replaced the framework based on SiamMask by a framework based on SiamRPN. SiamRPN was like a previous version of the SiamMask. It has less parameters and it is less complex. This is not a State of the Art algorithm but it is more than enough for the work it need to do. After all, is nothing more than a Single Object Tracking problem that does not require the best tracking algorithm since we are in a controlled environment and with not too much challenges. Achieving this objective was somewhat

more complicated than the previous one, since we have to study different algorithms and once chosen the right one, we had to fit it to the camera's control system.

The third objective was to design and implement the camera operation module. The camera operation module managed the movements of the PTZ camera based on the movements of the lecturer. We will do tracking with the algorithm developed in objective two, and based on the movements of the target we move the camera. This makes the online classes much more dynamic. To tackle this project we have based our module in the one developed by Awed. We divide the frame in to a grid and when the bounding box change the grid we move the camera in that direction. In addition, we zoom in to adjust the image of the teacher and the blackboard in the best possible way. This objective has also been difficult because inventing a 'system of rules' to manage the camera movements is not an easy task.

Reviewing the objectives we had at the beginning of the thesis, we can see how they have been achieved. In addition, we have shown how video live streaming can be done for the use of the developed software. This will help the teachers to improve online teaching.

5.2 Future work

Observing the final visual results it can be said that the developed algorithm is good but not perfect. As future work we would like to improve it as much as possible to improve the quality of the online classes. So, in this section we are going to propose a series of improvements that can be made to the thesis.

The first improvement proposed is to develop a dataset for moving cameras (like the PTZ camera). One of the problems was that when comparing tracking algorithms we did not have numerical results to compare them. We only relied on visual results. This was because we did not have a dataset with ground truth to evaluate them. All the datasets we had for tracking were developed for fixed cameras, so our algorithm, which was developed for the PTZ camera, could not be tested correctly. To solve this problem, we think that the development of a database with labels for moving cameras could be a good option.

The second improvement we have come up with to improve the quality of the online classes is to add more cameras to the class. If we add a second camera to the class (does not have to be a PTZ camera) we could have a camera tracking the teacher's movements and another camera fixed on the blackboard so that when teacher writes on it, it can be read correctly.

If the use of GPU's becomes more widespread (and cheaper) and laptops start to incorporate them, an improvement of the rule system could be made. For example, pose detectors could be added to track the teacher's gestures as well as his movements. Also the automatic target selection could be improved. If we achieve to select a more accurate automatic bounding box we can skip the step of selecting the manual bounding box.

With this three improvements we think that online classes could be improved.

Bibliography

- [1] Gebrehiwot, A., 2020. Real-Time Target Tracking to Position a Mobile Device
- [2] San Miguel, J., 2021. DLVSP – Video Tracking: single object – Fundamentals.
- [3] Trinh, C., 2019. A tour of Video Object Tracking—Part I: Presentation. Available at: <https://medium.com/@cindy.trinh.sridykhana/a-tour-of-video-object-tracking-part-i-presentation-8a8aa9da9394>
- [4] Singh Jalal, A. and Singh, V., 2012. The State-of-the-Art in Visual Object Tracking.
- [5] Matej, K. et al., 2018. The Visual Object Tracking VOT2017 challenge results.
- [6] Stack Overflow. 2021. What is consistency map (confidence map)? Available at: <https://stackoverflow.com/questions/21086082/what-is-consistency-map-confidence-map>.
- [7] Li, B. and Yan, J., et All 2018. High Performance Visual Tracking with Siamese Region Proposal Network.
- [8] Taberkit, A. and Bouguettaya, A., 2019. A Survey on Lightweight CNN-Based Object Detection Algorithms for Platforms with Limited Computational Resources.
- [9] Es.wikipedia.org. 2021. Cámara PTZ - Wikipedia, la enciclopedia libre. Available at: https://es.wikipedia.org/wiki/C%C3%A1mara_PTZ
- [10] En.wikipedia.org. 2021. Anaconda (Python distribution) - Wikipedia. Available at: [https://en.wikipedia.org/wiki/Anaconda_\(Python_distribution\)](https://en.wikipedia.org/wiki/Anaconda_(Python_distribution)).
- [11] Kazarinoff, P., 2021. Installing Anaconda on Windows - Problem Solving with Python. Problemsolvingwithpython.com. Available at: <https://problemsolvingwithpython.com/01-Orientacion/01.03-Installing-Anaconda-on-Windows/>.
- [12] Conda.io. 2021. Managing environments — conda 4.10.1.post28+e567fcd1b documentation. Available at: <https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#creating-an-environment-from-an-environment-yml-file>.
- [13] Wang, Q., Zhang, L. and Berinneto, L., 2019. Fast Online Object Tracking and Segmentation: A Unifying Approach.
- [14] San Miguel, J., 2021. DLVSP – LAB3 - Tutorial: SiamRPN Tracker deployment and evaluation.
- [15] En.wikipedia.org. 2021. Keras - Wikipedia. Available at: <https://en.wikipedia.org/wiki/Keras>.

[16] Techterms.com. 2021. Windows Definition. Available at: <https://techterms.com/definition/windows>.

[17] torchvision — Torchvision master documentation. Pytorch.org. 2021. Available from: <https://pytorch.org/vision/master/>

[18] En.wikipedia.org. 2021. Anaconda - Wikipedia. Available at: <https://en.wikipedia.org/wiki/Anaconda>.

[19] Matej, K. et al., 2020. The Visual Object Tracking VOT2020 challenge results.

