Facultade de Informática
UNIVERSIDADE DA CORUÑA

TRABALLO FIN DE GRAO
GRAO EN ENXEÑARÍA INFORMÁTICA
MENCIÓN EN TECNOLOXÍAS DA INFORMACIÓN

Development of new functionalities in an application for the analysis of animal behavior based on Artificial Vision

Student: Javier Dopico Pardo
Director: Álvaro Rodríguez Tajes

A Coruña, novembro de 2021.

To my family, for the continuous support

Acknowledgements

First of all I would like thank my parents and sister, for their continuous support not only in this last step but also during the degree.

Also thanks to the director of this project, Álvaro Rodriguez, for his help and the time dedicated to make possible that this project could go through.

At last, also thank you to all the friends I met during the degree, making the path easier, I hope our friendship will stay for a long time.

Abstract

Computer Vision is a scientific field that is nowadays almost everywhere, as it is a field of bast applications, which can go from robotics or traffic flow analysis to medical detection applications, for example. This is why, in this project it was decided to go into a deep review and develop over some of the basic parts that would form a Computer Vision application.

In this case, the application used to develop over and work with was Toxtrac, a program which main functionality is the tracking of animals based on Computer Vision. The aspects taken in account in this project will be Background Subtraction, Camera Calibration and Image Segmentation, improving along the study of these fields, functionalities related to them in Toxtrac.

Resumo

La Visión Artificial es un campo científico que, hoy en día, está presente en casi cualquier lugar debido a su gran variedad de aplicaciones, que pueden ir por ejemplo desde robótica o sistemas de control de tráfico a aplicaciones de reconocimiento médico. Es por ello que en este proyecto se ha decidido hacer hincapié y desarrollar algunas de los campos más comunes de la Visión Artificial usando como base una aplicación que utilizase Visión Artificial.

En este caso concreto, la aplicación será Toxtrac, un software utilizado en el traking the animales basado en Visión Artificial. Los aspectos que se desarrolarán en este proyecto pasan por la Sustracción de Fondo, la Calibración de Imagen y la Segmentación de imágenes, estudiando sobre estos temas en el proceso a la par que desarrollando nuevas funcionalidades o mejorando algún campo en la mencionada aplicación, Toxtrac.

Keywords:

- Computer Vision
- Background Subtraction
- Camera Calibration
- Image Segmentation
- Watershed
- OpenCV
- Toxtrac
- Thresholding

Palabras clave:

- Visión artificial
- Sustración de fondo
- Calibración
- Segmentación de Imagen
- Algoritmo de Watershed
- OpenCV
- Toxtrac
- Umbralización

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Introduction

Vision, is the capacity to see or the perception of the physical realities through sight. In humans, this ability resides in the eyes, that are the organs that make possible to receive all the ”information” that surrounds ourselves, by detecting light rays, then, eyes are capable of converting this into impulses able to travel to the brain (through the optic nerves), finally combined with the capabilities of our brain, we can make sense out of it.

Computer Vision[1] is a field of study in computer science which its main objective is to develop techniques so that computers are capable to see and understand the real content of any image or videos that get analyzed.

Computer Vision took its first steps between the sixties and the seventies when, firstly, Larry Roberts in his PhD thesis in 1960 came up with the possibility of getting 3D information from a 2D perspective image of some polihedric figures. Taking this idea, lot of researchers, worldwide, continued investigating and developing it.

It is a field with a big variety of applications, such is it that it has converged with other close-by disciplines, as image processing or computer graphics.

In reality, computer vision is a very complex field in computing, we could even say that there are really few challenges that have been fully solved. The main reason could reside in that, in comparison, human vision is, basically, too complex and is pretty difficult to reach that same level. In fact, there are some skills we have naturally that directly are not even in the next horizon of computer vision research.

The problem resides, partly, in the processes of understanding an image, where the capability of the human brain goes much further than where a computer can get. Hardware limitations are also a big issue, because, how could a computer be able, for example in an person or face recognition tool, to process and operate with big amounts of information as the brain does to retrieve the results in a **real-time** speed?.

As a way to understand this, an example, if we meet a person that we know, which we have stored his face, human brain works in a way that will not get just the result of knowing who he is by only fixing exactly his face with the image we might have in our remembers (that evidently wont be exactly the same as factors like the illumination in the street, any kind of disturbance like clothes or his hands that may hide any point of his face or even physical changes since your last memory...) but also will be able to get the result by matching any kind of recognisable feature.

That is, when trying to operate in any situation (like the one proposed) until this point of detail in real-time where the objective gets its difficulty really increased.

Nowadays, computer vision already has a vast number of functional applications, such as object detection or tracking any element of an image, being able of discarding the rest of the image that may disturb our result.

In fact, big part of this project will be about researching about this kind of applications on Computer Vision.

All the project will be based in the job that is, in part, already developed in the application **Toxtrac**, a free software, right now in its 2.96 version, used internationally for tracking a big variety of organism movements, from insects, fishes or rodents, for example.

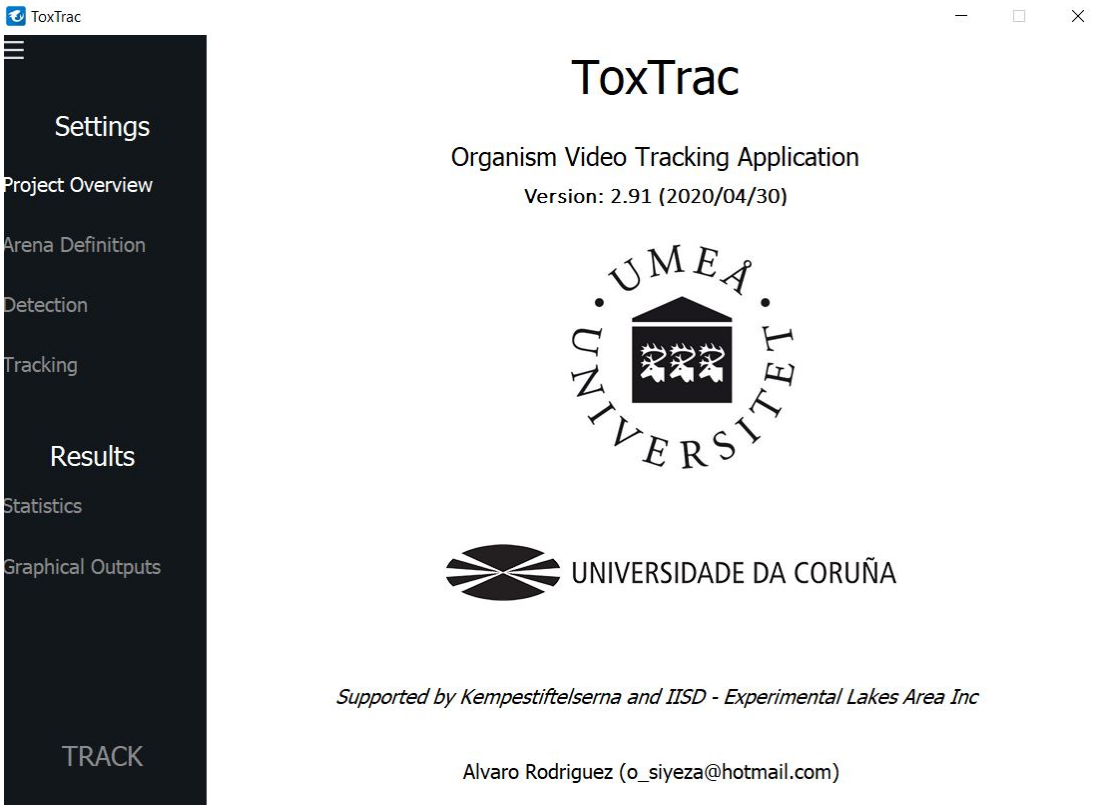


Figure 1.1: Toxtrac main screen

Toxtrac is a tracking application used to study and analyze the movement and reactions to internal or external impulses of the organisms recorded, facilitating the understanding their general behaviour. Some of the main features of Toxtrac are the capability to track more than one element at the same time in various arenas or no needing to specify the organism shape...

1.2 Motivation

In general, computer vision nowadays its even more required in almost every sector, with lots of different applications as in robotics but also in medicine o biology, for example.

Some of the reasons of why computer vision is so popular could reside in the speed on doing the analysis, avoiding human participation, so, automating the tasks that in other context, would be tedious. This is why, by consequence, computer vision also reduces costs.

In our particular case, we talk about animal behaviour analysis, and this has big variety of applications on disparate fields, from pure ecologic and biologic terms to psychologically ones. But, what is going to be done in our project, could in reality be surrounded in a complete different discipline.

This is like that due to the fact that the process of creating or improving a computer vision system works as a trial and error path, checking all the new challenges from all the points of view the developer can get, analyzing one or another algorithm depending on the needs of the situation... and, no matter the process would be probably similar, each problem would be different in the way to achieve the final correct solution.

1.3 Objectives

The main objectives of this project will be:

- To analyze in detail the Toxtrac tool, learn about it functionalities that already has and the ones that will be probably modified in this work.
- Study the background subtraction model already implemented in Toxtrac and analyzing and implementing a new alternative.
- Study the camera calibration module of Toxtrac, go inside the interface implementation and improve parts in it in order to help the potential users in the calibration process.
- Study, analyze and code a new segmentation model, in order to help basically the tracking app Toxtrac in new kind of scenarios with elements that could cause difficulties in the identification of the main organism.

1.4 Structure

This dissertation will be separated into 7 **Chapters** explained down here. After all these chapters, there is also included an extra section which sums up all the Bibliography used in it.

1. **Introduction**. Chapter in which is described why the project was proposed, the context of it and briefly talk about the main objectives.

2. **Basic theoretical fundamentals**. Here, the main fields occupied in this project will have a theoretical explanation sufficiently deep in order to give the reader the basic notions about them and understand what will be done after.

3. **Technologies fundamentals**. Now, what is going to be explained is all the tools used while working on this project.

4. **Planning**. In this chapter, the methodology and planning chosen will be explained, plus the resources that participate in the project and the final costs.

5. **Development of the Work**. In this chapter, it is going to be explained, step by step and detailed, the development of each one of the modules in which the project was divided.

6. **Conclusions**. Here, all the conclusions drawn from the project will be exposed.

7. **Future Work**. Final chapter where its included some ideas of paths that could be explored after finishing this project.

Chapter 2

Basic theoretical fundamentals

In this chapter we will talk about the main three theoretical fields in which the research done in this project was based. By this way, we will try to make clear the basic concepts about them and facilitate the comprehension of the dissertation and the job done. Those fields will be:

- **Background Subtraction**
- **Camera calibration**
- **Image Segmentation**

2.1 Background Subtraction

When we talk about **artificial vision**, usually, one of the first, basic tasks to do with the video to analyze is to locate in picture the objects that are in movement (detecting changes), differentiating between the background and the foreground elements, this is, by definition what it is usually called **Foreground Detection**.

One of the main approaches of this detecting technique, is what we are going to touch here, **Background Subtraction**. This is basically isolating the foreground parts detected in order to allow its extraction for further work.

Background Subtraction has many applications, from Traffic Monitoring system or Human Recognition to **Tracking** applications (objects, humans, animals, etc...), this is going to be the main point on why we are touching this Background Subtraction approach.

In real scenarios, external agents can appear and as noise, difficult this task, as in a recording with changes in the light over the frames, background elements that may create different textures as places with water, wind or grass... Is in this kind of cases, why to solve this, there have been developed solutions as "background subtraction **algorithms**".

In general, background subtraction methods, are proposed in a way that they get the background model (background image) that will be used as reference in the subtraction by calculating it from a sample of previous frames.

Of course as there is a study in this field, no matter the basic principle of its work is, as said, similar, there exists a big variety of **methods**[2] that change in the way of calculating it.

2.1.1 Running Gaussian Average

This model is one of the simplest methods in background subtraction. It offers to maximize the speed of execution and low memory requirements allowing real-time performance, while it still is a sufficient reliable method (but far for being the most accurate).

As explained in 1997 by Richard Wren et al.[3], it is based on applying a Gaussian density function taking on the last n pixel's values. The Gaussian density function in order to avoid starting again for each frame it gets computed a running average as:

$$\beta_t = \alpha I_t + (1 - \alpha)\beta_{t-1}$$

Taking into account that β_{t-1} is the background image at $t - 1$, α is a value between 0 and 1 that fixes the speed of ignoring background information.

Then, if $|I_t - \beta_t| > \sigma_t$, (Being σ_t the standard deviation), I_t will be qualified as **foreground**.

2.1.2 Temporal Median Filter algorithm

This Temporal Median Filter[4] method is one of the most popular applied in Background Subtraction, it bases are founded in the calculus of the **median** in order to get the background image.

Introduced in a study of Lo and Velastin [5] dealing with videos with high illumination changes through frames (2001), as said, it just gets the median in the last k frames in the video recorded in order to construct the model to use.

It is true that, as said by Cucchiara et al. [6] in an article in 2003, this method proved to be really effective and quick due to the no-so-big computational requirements of the operations performed (in comparison to other techniques as the Gaussian techniques with more complex models) but, at the same time, there should be pointed out its main disadvantages, as it needs a kind of buffer to storage the N number of frames selected to calculate the median and also that the adaptability on the model construction is less noticeable as there is no way of calculating any kind of deviation that may help.

2.1.3 Mixture of Gaussians

In the algorithms proposed until now, it is true that both models constructed the new background model when changes in the scenario happened, pretty quickly. But, it could happen that this changes in the background object changed again even quicker than the speed the background model algorithm is performed. That is because scenario was being treated as if there was only one value for the background.

That is why, multi-valued background model were proposed by Stauffer and Grimson in [7], 1999. By this approach multiple background objects could be managed at once, even defining both foreground and background values.

The history of a pixel value during the T frames defined it is obtained by a Mixture of (K) Gaussian distributions. The **probability** of getting an x value in particular is obtained by:

$$P(x_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t})$$

Where K, of course, was the number of distributions defined, usually a value between 3 or 5, defined by the computational power, $\omega_{i,t}$ is the estimation of the weigh of the i Gaussian in the mixture at time t, $\mu_{i,t}$ is the mean of the i Gaussian in the mixture at time t, $\Sigma_{i,t}$ is the co-variance matrix of the i Gaussian in the mixture at time t and η is a Gaussian probability density function (**pdf**), as seen in the first explained method.

For computational reasons the standard deviation is assumed to be the same for the three channels, getting $\Sigma_{i,t}$ simplified to

$$\Sigma_{i,t} = \sigma_i^2 \mathbf{I}$$

This, at the same time that makes the method miss in accuracy, it gains in simplicity, avoiding a much more complex matrix.

After this computations, all the distributions get ranked following the result of

$$\frac{\omega_i}{\sigma_i}$$

and it is assumed that the higher and compact is the distribution the more likeliness there will be to belong or not to the background. Then the first B distributions (following the just explained order) are chosen as background if they follow the following in-equation:

$$\sum_{i=1}^{B} \omega_i > T$$

In which T is a selected threshold.

In each new t that comes, a new x_t value has to be assigned to the best fitting distribution.

This is done, from all the distributions that follow this in-equation $(x_t - \mu_{i,t})/\sigma_{i,t} > 2.5$. Also the parameters ($\mu_{i,t}$, $\sigma_{i,t}$,$\omega_{i,t}$) have to be estimated to update the mode. These parameters are only updated for the fitting distribution, following equations similar to the ones seen in the Running Gaussian Average section.

If there is no matching, last ranked distribution will be replaced by a new one centered in x_t with high variance (σ_i) and low weight (ω_i).

2.1.4 MOG-2

Now, the next method explained will be the one that Toxtrac implements nowadays.

Specifically, Toxtrac (As specified in its Code Guide) runs with an algorithm based on the previously explained Mixture of Gaussian method (MOG) but improving it.

This ideas where exposed by Z.Zivkovic in two articles, from 2004[8] and 2006[9]. The main point in which this improvement was based is that the own algorithm selects the correct number of Gaussian distribution for each pixel. Due to this characteristic, it provides high adaptability in scenarios with high illumination altering through frames.

This algorithm works with Bayesian probability to calculate in each pixel how similar is an x pixel in a specific frame in order to qualify it, by this criteria, as **Foreground** (FG), or as **Background** (BG), expressed as a formula like:

$$p(BG|x) = \frac{p(x|BG)p(BG)}{p(x|BG)p(BG) + p(x|FG)p(FG)}$$

In general, we could assume that we do not know anything about the objects belonging to the foreground so, as a standard base: En general, podemos asumir que no conocemos nada sobre los objetos del primer plano por eso, de base:

$$p(BG) = p(FG) = 0.5$$

or if we have information enough about the scenario we could fix a predefined value.

The background model gets identified as $p(x|BG)$ and will be decided if a pixel is background if $p(x|BG)$ is higher than the value.

This Background model is obtained from a sample formed by the observations during a T time chosen in a way it could be enough to get addapted to the changes

$$X_{ij} = x_{ij}(t - T), ..., x_{ij}(t)$$

For each new sample, the "training" data set gets updated by adding those samples while discarding the older ones, while a fixed set value does not get surpassed. The values of the pixels that do not fit as the background distribution, get tagged as Foreground until there

appears a Gaussian, with new data that makes it get included as Background.

This Mixture of Gaussian models have as main inconvenient that if an object of interest for the analysis, classified as Foreground, remains for enough time static, could happen that the model changes the object tag as Background until it ”moves”. In particular, this was a point taken into account to decide a new Background subtraction option could be useful in the tool.

2.2 Calibration

There is a difference between the point in the three dimensional point of an object and the corresponding image pixel of the image a camera has recorded, Camera calibration helps to get this relationship.

This relation gets defined by a series of geometrical parameters and the process to obtain them and solve the equation to get this equality of points, its **Camera Calibration**

Theoretically, Calibration gets defined by three basics, the **camera model**, the **distortion model** and of course the **calibration methods** proposed in each problem so then get the model to proceed and the mathematical parameters needed in the modeling of the solution.

Camera model

is the mathematical process of calculating how a point in the 3-D world will be settled in the 2D image coordinates.

The main Camera Model used in this kind of processes is the model of **pin-hole**[10], also called projection perspective. It is also the simplest one known. It resides in the concept of the pinhole camera, which is just a simple totally-opaque camera with no lenses and with a small hole from where the light can go through forming a projection of the image (but inverted, creating an effect usually called obscure camera).

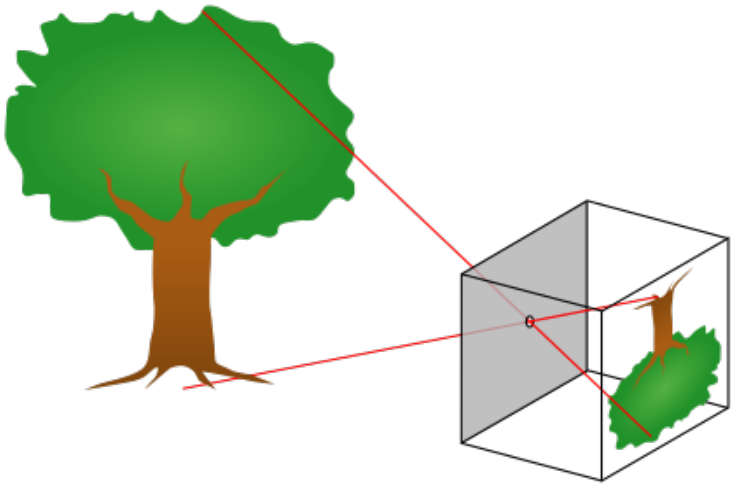


Figure 2.1: Example of how the Pinhole ideal camera would work.

This projection perspective model is based on **Projective geometry**, a mathematical field that studies over geometrical figures and its incidence geometrical properties but apart from the measurement or distance concepts. This could be understood easily as the geometrical obtained when a person is observing a point while situated in that point. Indeed any imaginary line that connects to the eye, will be seen by the observer as just a point in the projective plane, as it would be impossible to ”see” all the points behind the ”observed”.

Distortion model

. In general, a distortion in this field would relate to any kind of deviation to the original rectilinear projection. So any kind of imperfection caused in this process will need from the distortion model and its parameters to surpass these errors. These kind of distortions may come for example from the own lenses.

Main steps using the Pin-hole Camera Model

The whole calibration process, will follow approximately the next steps, when using the previously mentioned **pin-hole** camera model:

- First step is to realize about the position of the camera and calculate the rotation of it respect to the analyzable object. Then, with this we will be able to calculate, starting from the coordinates in the 3 coordinates axis of the real-world map, the same point coordinates in the map system of the camera. Apart from the transform itself that leads

us to this point, it also needs the **Rotation Matrix** R and the **Translation Matrix** T.

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = R_{3x3} \times \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + T_{1x3}$$

- Next, a transform has to be made in order to go from the camera coordinates system to the ideal camera coordinates system. The ideal camera used as reference in most of the models is the **pinhole camera**, a photographic camera without lenses, that by definition could be an opaque box with a tiny orifice from where the light can go through (the pinhole itself), and a photosensitive material. Then, the transform will be used to convert the obtained coordinates from last step into the coordinates in 2D of an image.

  It will follow the next parameters, where M is the transform matrix, build by $f(f_x f_y)$, which represents the **focal length** (the distance between the lenses and the camera sensor) and $c(c_x c_y)$, which is the optical center (where the point would be projected from the middle of the lens, in the coordinate system of the image). M will be something fixed while the focal length is not varying.

$$M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = M_{3x3} \times \begin{bmatrix} \frac{X_c}{Z_c} \\ \frac{Y_c}{Z_c} \\ 1 \end{bmatrix}$$

- Finally, last step would be to calculate and obtain the difference between the position obtained in the last transformation and the real positioning in the image.

2.3 Image Segmentation

Image Segmentation is how it is called the process of dividing an image in different regions, basing this division in the characteristics of the pixels, in order to identify objects, limits or borders so then the image can be simplified and favour or facilitate its analysis.

Segmentation is applied in lot of different fields, from health applications to even the movie industry, with such examples as medical instrumental that can identify from an scanner any kind of problem in bones, organs or muscles, or when producing a movie, in scenes that make use of "chroma" and then separate the actor from the background when editing it.

The most basic option of segmentation (even sometimes, the first step in a more complex process) is called **thresholding**.

Thresholding in definition, is the operation of changing the value any pixel in a original image which intensity ($I_{i,j}$) is higher or lower a prefixed constant T by this way.

- If $I_{i,j} < T$, then replace with a black pixel.
- If $I_{i,j} > T$, then replace with a white pixel.

The way the T constant value is selected, is fixed by the individual working with it, but in reality, there already exist techniques to do a complete automatized thresholding seeking for the value that goes more appropriate in each circumstance. Those, were explained in a research paper by Sezgin and Sankur in 2004[11] and divided according to the criteria they use to calculate the threshold:

- **Histogram-shape techniques**, peaks, valleys or curvatures of a smoothed histogram are analyzed.
- **Clustering-based methods**, gray-level images get clustered in two pieces, background and foreground, or as a mixture of two Gaussians.
- **Entropy-based methods**, whose algorithms use the entropy obtained in different ways.
- **Object attribute-based methods**, checking the similarities between a binary image and its gray-level version.
- **The spatial methods**, checking for similarities between pixels and their correlations
- **Local methods**, adapting the T value depending on the pixel of the image.

Now, again, about image segmentation, it can be divided in many ways, depending on diverse criteria. For example, we could say, all image segmentation methods can be named

into one of these 3 types, which differences could be said to exist in their complexity on the way they work[12]:

2.3.1 Binary Segmentation

When in the segmentation we only differentiate between 2 elements, foreground and background using the basic method of **thresholding**, mentioned above, we call this a binary method. Pixels above the thresh turn white (foreground) while the rest stay black as background.

Usually, the input of a thresholding operation is an image, no matter if coloured or gray-scaled, while, the output, as said will be a binary image.

2.3.2 Assisted segmentation

Or also called **semi-automatic** methods, where, usually, the user makes a kind of tagging on a region of the image so that the algorithm then, executes the segmentation due to that parameter. In this category there are three big examples of what this is.

- **Watershed**, 1979. This algorithm takes the image as a topographic map, created taking in account the intensity from a gray-scaled image used as reference. The algorithm job would be, similar to the effect of flooding the topographic map created from the image with a liquid and letting the liquid to separate the image in regions, depending on where the liquid passes or not using the higher peaks of the map as dams[2.2].

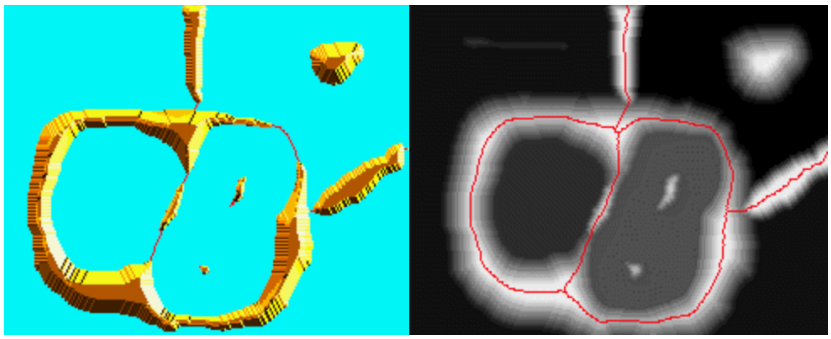
s 

Figure 2.2: Example of a topographical map generated and being ”flooded” with the resultant Watershed segmentation over the original image.

The manual point on this algorithm resides in the solution reached to avoid getting over-segmented regions on the resultant images[13], where the flooding could be kind of controlled by putting the own user some markers. Nowadays there are algorithms and functions that also can help us in generating those Markers like the one used in the Prometheus Project[14] or using the distance transformation of an image.

- **Grabcut**, 2004[15]. With surprising simplicity, GrabCut is able to segment an image with the help of human iteration. In this case, the user just has to point an area over the image, the tool will segment the image using this area as a reference, leaving the non-framed area as safe background, like in figure 2.3

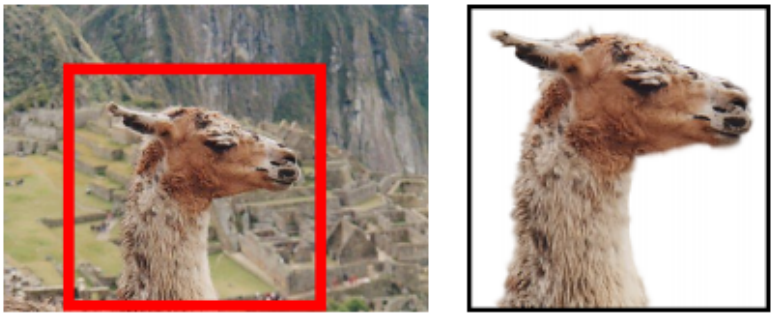


Figure 2.3: Example of how GrabCut operates and what human input it needs.

- **SIOX**, 2006[16]. Or Simple Interactive Object Extraction is another semi-automatic segmentation method implemented in some famous image-editing programs as **GIMP**. The user, from an RGB original image, selects using any kind of selection tool the program that implements the SIOX technology has and marks any regions of interest to be cataloged as foreground.

2.3.3 Segmentation based on Machine Learning

. More or less, in 2012, **Machine Learning** got into the computer vision studies and big and relevant results started to appear. In general, this appearance marked a before and after in image segmentation because of it high performance in solving really complex problems leading almost every previous solution to get obsolete. Only a few options were maintained useful, as the simple thresholding or the recently explained, Watershed and Grabcut.

There are two[17][18] main distinct types of machine learning, **Supervised** and Unsupervised M.L and based on data available or the context at hand, the interested user will choose to train an algorithm using a specific learning model.

Supervised Machine Learning

In Supervised ML, the context surrounding the algorithm is that the user can introduce in it data **labeled** ("knowing the answers") in a way that the algorithm trains with training samples formed by input and output. Could be seen as giving a paper of questions and answers, so

then, in the future, with a real problem in front, the algorithm predicts the result of it from the data learned in that way in the learning process.

Supervised machine learning is usually seen in two different ways, depending on how the output of the algorithm is, in **Classification** problems or **Regression** problems.

In **Regression** problems the output data are continuous values, as what to expect the value y to be if we have these or those arguments in front, as could be a problem of checking the price of an alleged apartment with some particular kind of information about it as its position in the city or its size.

In **Classification** problems the output data are discrete values identifying it from a cluster or a group of something. Like classifying a figure as a square or a circle, for example its training data set here could be pictures of shapes labeled with which shape is each one.

Unsupervised Machine Learning

In Unsupervised ML techniques, as an opposite to the supervised, the training data set would be just input samples, where the algorithm would need to, in a way, understand and get information out of it, As asking questions where the algorithm does not know the answer.

Unsupervised have different ways of organizing data,

- **Clustering**: One of the most used, where the model gets information about similar points (characteristics) of the training set and uses this to create and divide the output in groups or clusters.
- **Anomaly detection**: As its name says, the model is able to detect big discrepancies in the data set.
- **Association**: By training, the model can reach a point of knowing for some key characteristics, the predictable result.

Applications in Image Segmentation

As said firstly Machine Learning applied in Image Segmentation is almost the vanguard in this field, most of them based in the idea of **Deep Learning**, a sub-field in Machine Learning dedicated to create algorithms inspired in the structure and functionality of the brain.

The most basic model of Deep Learning is the so called **Artificial Neural Network** (ANN) but the most used nowadays in terms of image segmentation is the **Convolutional Neural Network** (CNN). CNN was mostly introduced in 1998 by Lecun et al. [19] and it is the main Deep Learning Architecture in computer vision-related fields. Segmentation methods based on CNN are usually trained by Supervised Machine Learning, in contexts were big amounts of labeled samples can be thrown. One of the main fields where Deep Learning architectures for image segmentation are used is in medicine applications.

2.3.4 Other Taxonomies

Other way of classifying segmentation methods resides in knowing how they really work more in detail.

Color Segmentation

It is basically, classifying image pixels by their colour. Of course, as by this definition, the most simple way of this would be the **thresholding**. Other known methods[20] are the **Histogram Thresholding** or the **Clustering Thresholding** for example.

In the Histogram technique the different segments are obtained checking the histogram and looking for the peaks.

While, in the Clustering one, the pixels are organized depending on their color, directly creating clusters and then, use those to continue classifying the pixel images.

Region-growing methods

Along with the original image, a selection of seeds is used as input. Then from those seeds, in each iteration of the algorithm, they follow a path over the neighbour pixels, comparing them following a criteria, usually the similarity in their intensity level between the intensity level of the regions alongside, by this way, at the end of the ”execution” each pixel on the image has a region assigned and the image ends segmented.

Graph partitioning methods

where the image gets modeled as an undirected weighted graph, then the pixels are seen as the nodes and the weights of the edges are the similarity between neighbour pixels. Then the graph can be segmented into clusters by on similar nodes and the output is each one of the segmented parts of the original image.

Model-based segmentation

it is based on the hypothesis that the elements to segment in the image are going to be tending to a geometrical shape. Then it is matter of seeking a model that seeks for this shapes in the image or its possible variations.

Multi-scale segmentation

it is calculated at different scales and then it is spread from larger scales to more fines. The complex on how this method segments is arbitrary. Any kind of region found must be connected with the others in any way.

2.3.5 Mathematical Morphology

As an extra tool, and commonly used in processes of image segmentation, mathematical morphology will help into this task.

It was born in mid 1960s[21] by the studies and work of Jean Serra and Georges Matheron in the Centre de Morphologie Mathématique (CMM), based in Paris. Its main points reside in the **set-theory** and some of its main applications reside in image processing and segmentation tools like edge detection.

Main approach used with Mathematical Morphology are the morphological transformations, which basic functionality will be to get conclusions in the comparison between an image and an **Structuring Element**, which is a simple binary image that has a shape (usually an square, circle or cross) and makes possible to see how this element fits or not in the image compared.

In OpenCV[22], which has modules of Mathematical Morphology that makes this process easier in coding, this structuring element (or kernel) is one of the inputs in this field main functions and lets the user easily choose the shape and size of this element.

The most basic Morphological operations all follow the principle of Translation-Symmetry[23] which basically explains, in geometrical terms, the movement of a figure without any rotation (It implies, that at least in one direction, the geometrical shape will be infinite, as a repeating wallpaper, for example).

Those most basic and used operations are **Erosion** and **Dilation** but from which other popular operations come, like **Opening** and **Closing**

Erosion

Is and operation which basic idea will be to erase those regions where, in the comparison with the kernel, do not fit, ending into a more skinnier version of the figure of the original image.

Theoretically expressed, Erosion of an element A, letting A being expressed within the Euclidean Space E and taking in account that B will be the structuring element, is the set of all A points in which, during the movement (through a movement vector, z) of B through all points of A (expressed as B_z), B remains in the space covered by A.

It can be mathematically expressed as:

$$\text{Erosion of } A \text{ by } B = A \ominus B = \{z \in E | B_z \subseteq A\}$$

As a graphical example this could be a possible output on using the Erosion operator over the next figures, being $\epsilon(X)$ the result of the erosion, as the set of all points of the original figure that remain there after the eroding process.

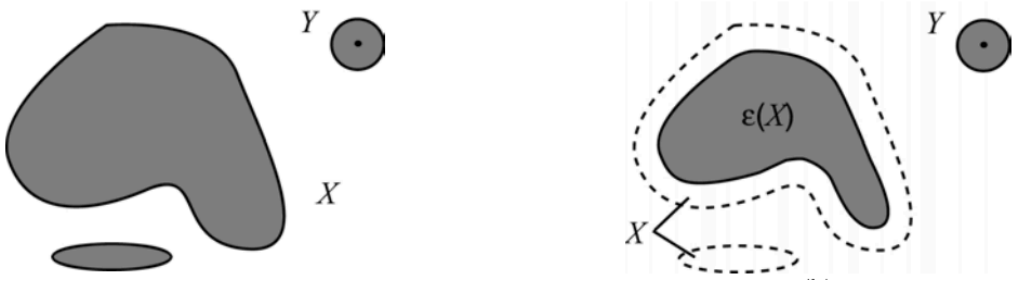


Figure 2.4: Example of Erosion working over a figure.

Dilation

Is the second fundamental Morphology operation, and works as the opposite of **Erosion**, ending by thickening the boundaries of the figures in the image where the operation was made.

Theoretically expressed, Dilation applied on element A working with the structuring element B, being A an element defined within the Euclidean space E, will result into the set of points that are part of the geometric space of the element B (During the movement of B through the movement vector z) while its center point resides inside A.

It can be mathematically expressed as

$$\text{Dilation of } A \text{ by } B = A \oplus B = \bigcup_{b \in B} A_b$$

It is Dilation a commutative operation so:

$$\text{Dilation of } B \text{ by } A = A \oplus B = B \oplus A = \bigcup_{a \in A} B_a$$

As a graphical example, this is a possible output of applying a Dilation function over the first figure, being $\delta(X)$ the dilation result, as the set of points of the structuring element, that, while moving on the Euclidean Space, contain any point of the original figure.

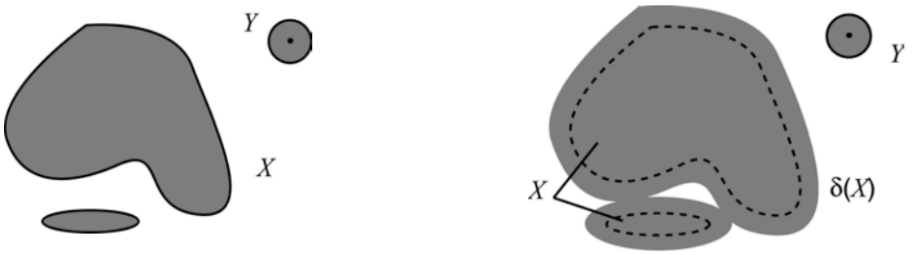


Figure 2.5: Example of Dilation working over a figure.

Opening

Usually in morphology it is impossible to apply the inverse to the Erosion Operation nor Dilation, so an approach that would let the user get a similar figure to the original would be to apply, after Erosion a Dilation and vice-versa.

This operation[24][25] is also one of the most used in Mathematical Morphology. Opening of A by B is the result of applying the erosion of A by B followed by the dilation of the result obtained by B.

So, it can be mathematically expressed as

$$\text{Opening of } A \text{ by } B = A \circ B = (A \ominus B) \oplus B$$

An easy example of when to use an Opening operation could be as, in the example below, when trying to get the same image as the original but without any kind of detail that is smaller (or thinner) than the interesting figure, so playing with the kernel shape and size you might get the desired result. Big point is that the same kernel is used for both eroding and then dilating.

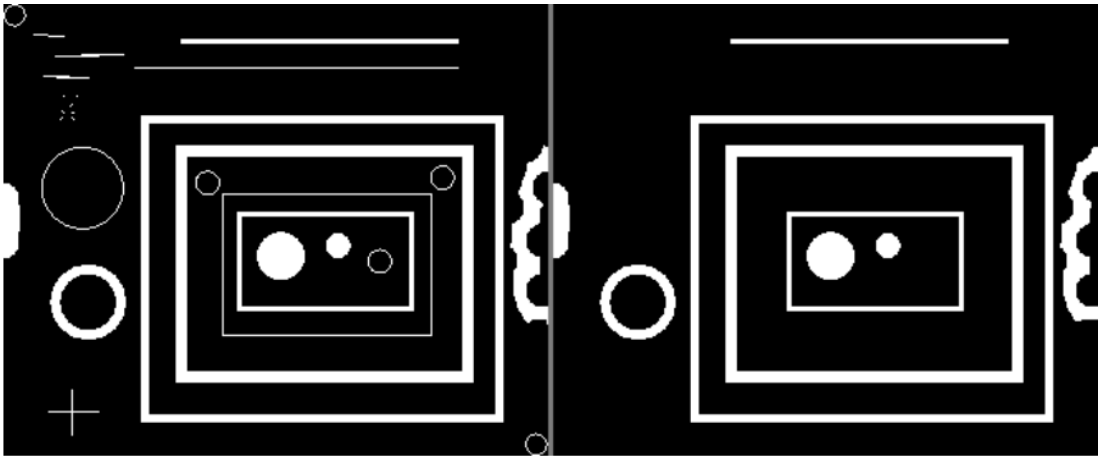


Figure 2.6: Example of an Opening operation working over the figure, erasing the thinner elements .

Closing

As counterpart to the Opening operation, Closing of the element A by kernel B will be the result of applying firstly the dilation of A by B followed by the erosion of the result obtained by B. Again, using the same structuring element B for both operations.

So the mathematical equivalent for Closing operation is

$$\text{Closing of } A \text{ by } B = A \bullet B = (A \oplus B) \ominus B$$

An example of when to use Closing[26], to explain it in a simple graphical way (see the image bellow), could be in an image where the figure or figures have any kind of holes or spaces between them that we want to fill. That is where the dilation part of the operation works, as engrossing the "walls" of the figures that surround those interested holes seeming them to fill. After that, with that problem solved, the erosion part of the closing operation will fix the possible thickening of the figure, that we of course did not meant to generate.

Figure 2.7: Example of the closing operation working over the left figure, showing a similar scenario to the one proposed, trying to fill those holes..

Extra Morphological operations

Although the four techniques above are the most basic and used ones, there are more options that still follow the Mathematical Morphology basis.

- **Skeletonization** or Medial axis transformation, an operation that will basically try to get rid of most of the pixels from the figure identified of the image and letting generate a kind of skeleton shape that will preserve the extension of the figure (and the connection between all the possible ends of the figure) in all directions as a difference with thinning by eroding. Basically as the own method says by its name, creating a kind of skeleton from any figure that is treated.
- **Hit or Miss** transformation, is a really basic morphology operation based on the erosion method but know incorporating a second structuring element, called, for example C.

Both structuring elements now are disjoint elements such as that B and C never have common points in the Euclidean space.

The result of the operation will be the set of points where B fits in with A while C does not (completely). It is mathematically expressed as

$$\text{Hit or Miss transformation of } A \text{ by the pair } B \text{ and } C = (A \ominus B) \cap (A^c \ominus C)$$

Being A^c the set of points that are not in A.

- Other extra but popular operations could be the **Pruning transformation**, the **Top-Hat transformation** or **Granulometry**.

Chapter 3

Technological Fundamentals

This chapter is meant to be a brief explanation of every single tool used in the development of this project. Some of the ones pointed out will be **C++**, **OpenCV** or **Qts Designer** among other.

3.1 C++

C++ is a programming language created in 1979 by danish Bjarne Stroustrup with the main intention of improving the already existing textbfC language with some tools that make possible object manipulation.

Figure 3.1: C++ logo.

In fact its name, C++ comes from seeing this new language as a C improvement, as a C extension.

In Stroustrup words [27], it was preferred to work ”over” an already known language as C, no matter its cons (that of course, were already acknowledged) than working from scratch on this new language that finally ended being C++. C was the chosen one to ”improve” due to 4 characteristics.

1. It is a quite low-level programming language and its versatile.
2. It fits in most of the programming tasks of multiple systems.
3. Can run everywhere and any kind of thing
4. Fits into the UNIX environment.

Again with C++, some of the main characteristics obtained with it where that:

- It is an **Object Orientated** Programming Language
- **Simplicity**. As it is C++, it makes easier to understand it in depth for programmers, also, taking in account that, as it is like C (but with classes), if it is learned by a C programmer it will feel almost the same.
- **Agility**. Its characteristics make C++ a programming language with a quicker compilation and execution times than its "rivals" like Python or Java.
- As it is a **High-Level** programming language, it makes that the "language used" gets more easily comprehended as it is more similar to human English.
- **Popularity**. Nowadays we can say it is a widely extended programming language. It has loads of applications, as it is common to see databases, as MySQL for example, written down in this language, the same happens with famous Operating Systems, compilers or even more basic user experiences as applications related to games or web browsers.

3.2 OpenCV

OpenCV[28] is a cross-platform library available for Windows, Mac or Linux that lets the user develop computer vision apps in real time.

Initially developed by Intel, first version was released in year 2000 and thanks to the scientific community has been growing until nowadays. Originally, it was created for the language we are using, C++ and later based on the original API, other options for Java or Python where created. Nowadays it is available in loads of Operating systems as Windows, FreeBSD, OSx, Linux...

Some of the most commons applications for OpenCV are **image segmentation**, facial recognition, tools for managing scenarios no matter if 2D or 3D, tracking tools, gesture recognition or even it is seen in robotics.

OpenCV is the best choice when talking about these kind of applications due to its efficiency in computational terms for coding applications that need real time outputs. Along with

this performance promises, OpenCV is also the best solution in terms of managing Computer Vision applications with high simplicity coding methods while offering complex results.

It has over 500 different functions covering all kinds of purposes Computer Vision application could need. Also, as Machine Learning in this field is getting more and more useful, OpenCV included a sub-library full of more functions that could cover this area.

In fact, OpenCV[29] can be divided into 5 modules,

1. **CV**, which includes all the components related to image processing and high-level computer vision algorithms.

2. **MLL**, the already mentioned Machine Learning Library, with clustering and statistical methods.

3. **HighGUI**, with I/O methods and useful functions for storing and loading videos and images.

4. **CXCORE**, that includes the basic data structures, the XML support, for example.

5. **CvAux**, contains some extinct areas and also algorithms in its experimental phase.

3.3 Visual Studio

Microsoft **Visual Studio**[30] is an integrated developing environment (IDE) produced by Microsoft. It comes with its own Code Editor, Debugger and visual Designer and supports more than 30 different programming languages while letting the user create both native and managed code.

Visual Studio is an extendable IDE, letting the user connect plug-ins from other tools in order to expand its own functionalities, like source control services as Git, being this one of the most common.

3.4 Qt Designer

The Qt Designer [31] is a tool integrated into the Qt framework (1992), used to design graphical interfaces to applications on different platforms.

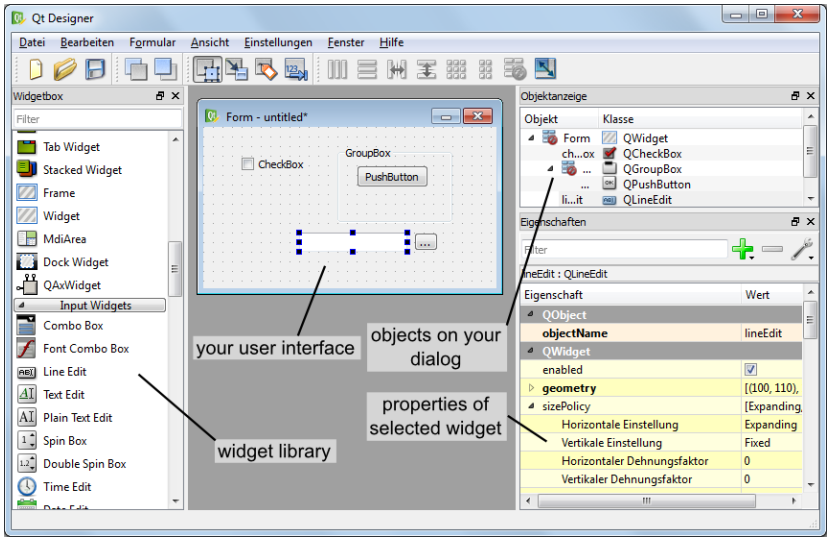


Figure 3.2: Qt Designer environment and main tools..

All the elements integrated in the Qt Designer get perfectly integrated in the programming code, once there, Qt lets the user, coding its behaviour or characteristics dynamically.

3.5 VirtualDub2

An improved version [32] of the original **VirtualDub** is a free and open-source video editor that lets the user reed and write over the most famous video formats (AVI, MP4, MOV...) and work easily in a frame by frame situation.

3.6 GIMP

Called after "GNU Image Manipulation Program", and born in 1996, **GIMP** [33], is a cross-platform, open-source application used in image editing, created in a way is keen to be used with third party plugins.

3.7 Latex

LaTeX[34], 1984, by Leslie Lamport, based on **TeX**(an older low-level language), is an open-source writing system focused on the creating of documents that require high-level typography quality, that is why is commonly seen in papers, articles or books.

Some of its main characteristics are the easiness of typing mathematical **formulas**, the facility to work with figure references no matter how large is the document and the bibliography system. **OverLeaf** was the Latex editor used.

Chapter 4
Planning

In this section of the dissertation, an explanation about the methodology and planning used in the project will be done. Also, a brief addition about the **human and material resources** that took part.

4.1 Methodology

As this project is a research type one, a typical decision to make in terms of choosing methodology is to take, for these kind of projects, an **AGILE** one.

AGILE is a group of software development methodologies that propose an incremental and iterative approach to software design. Agile bases its idea in providing the ”customer” with a functional product from early stages of the development already. This decision, making the client to be more involved in the process, creates, in general, higher quotes on satisfaction on clients.

Agile methodologies appeared approximately in 2001, with the appearance of the **Manifesto for Agile Software Development** and the following of these values:

- That **Individuals and interactions** are over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- Finally, **Responding to change** over following a plan

While making clear, in that same Manifesto, that, although it is clearly said above that the ”left part of the sentence” elements are the ones most valuated, the ”right part of the sentence” ones, are not excluded at all and still are taken into account.

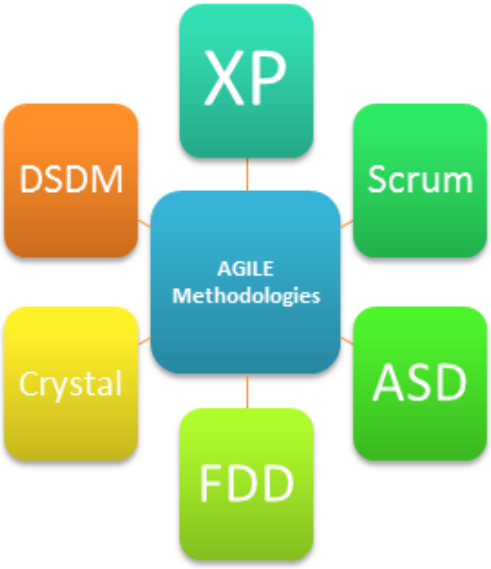


Figure 4.1: Graphic with some of the most popular AGILE methodologies.

4.1.1 SCRUM

From all the different varieties of AGILE methodologies (figure 4.1), the most popular one is **Scrum**, which is the one followed in this project.

Scrum is a framework that will help individuals and organizations in the development of complex projects. Some of the main points of Scrum are the main intention of **seeking for work optimization**, **continuous job checkpoints** that are sent to the client, **fluent client - work-group communication** that will make possible some **flexibility** to the original requirements from the client... There are more basic points in Scrum ideals, but the ones just presented are the reason why these method was selected.

The main benefits that appear using of Scrum methodology are

- Increase of the **productivity** in the team due to the work structure of the method.
- Increase of the **quality of the final product** and more **fidelity** in the following of all the requirements, as the flexibility and the constant communication with the client makes the team to be able to change the immediate work focus.

On the other side, some of the possible disadvantages of Scrum could be

- Projects might be **more extended in time than the original proposition**, due to the way Scrum leads with project calendars.
- The **continuous number of reunions** could reach a point of being a disadvantage if the crew members are not accustomed to this way of working.

- Related in part with last point, Scrum could reach into a big **challenge** to implement in **larger groups**.

To understand Scrum methodology completely we have to talk about all the pieces that form it, the **roles**, the **artifacts** and the **events** or ceremonies.

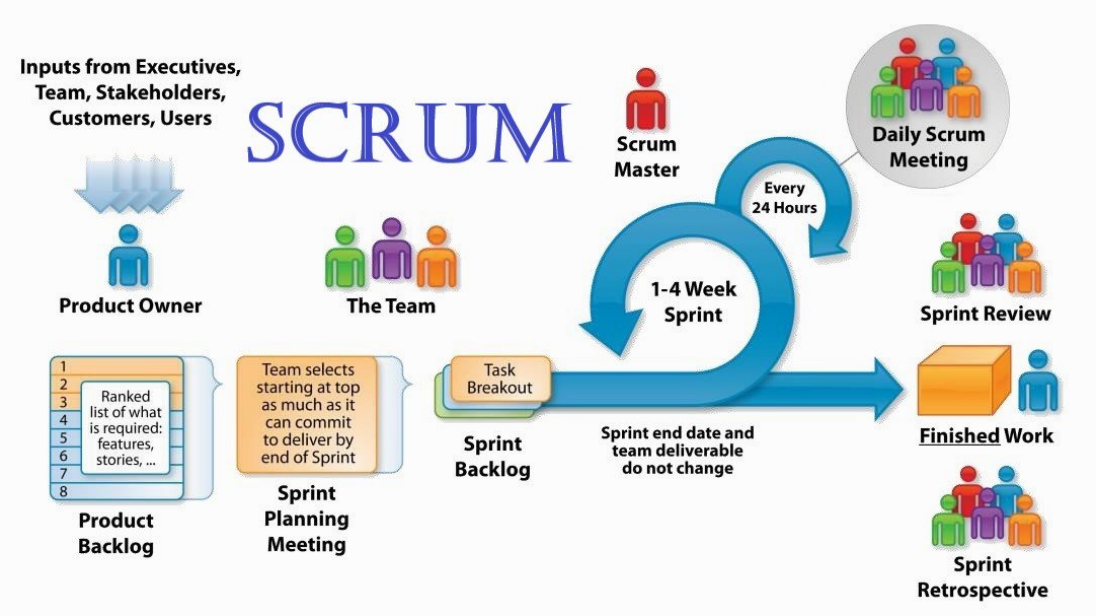


Figure 4.2: Graphic showing all SCRUM basics.

A **Role**, in Scrum terms, each basically each one of the individuals that participate more or less directly in the project. Is basic in Scrum to manage people establishing them with these role tags. There are three main roles possible.

- **Scrum Master**: Is an expert in the methodology basics that plays a role of a coach to assure that the project team is working according to Scrum rules.
- **Scrum Team**: Or Project-Development team are the ones that mainly work on the implementation of the project, they do not have different hierarchies. The length in a generic team might go from one or two to seven or eight members, really different depending on the context.
- **Product Owner**: Is like a figure that appears representing the client. It takes a role of spokesperson but he understand deeply the product needs and characteristics. The ideal point is that this role is just played by a single person to facilitate the communication. Is the one that maintains the **Product Backlog**, that will be explained later.

In Scrum process, there are three types of **artifacts**, the already mentioned **Product Backlog**, the **Backlog Sprint** and the **Product Increment**.

- **Product Backlog**: Is a kind of a list of tasks proposed by the Product Owner, going from new functionalities to error solving. This list is not closed, being more and more complete as the project goes on.
- **Sprint Backlog**: Is a subset of the Product Backlog. It is the work the Developers want to do during a Sprint in order to achieve the Sprint Goal, being the sprint goal the main purpose of a sprint.

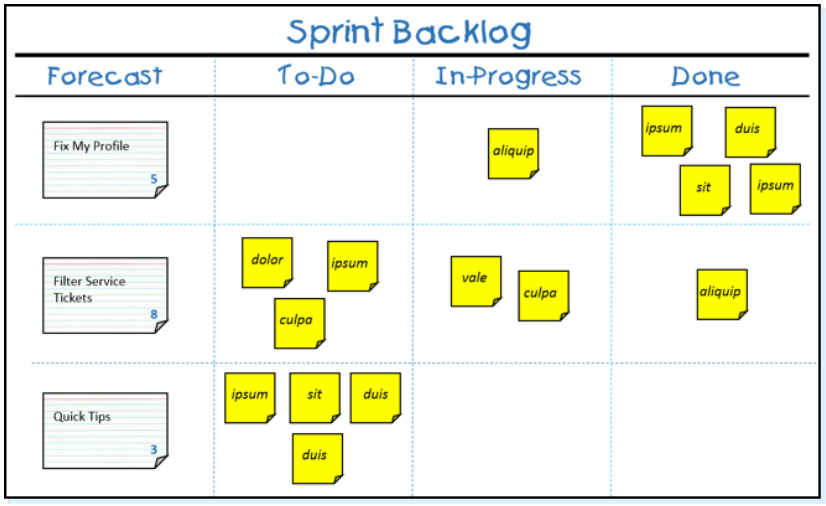


Figure 4.3: Example of how a Sprint Backlog could be represented in a board

- **Product Increment**: Is the final result or output of each one of the sprints. A valuable increment is the one that is usable. The output that comes from a Sprint Backlog is the result of all the elements of it that can be assured as DONE. The elements that do not fulfill this, will be postponed to the next sprint.

Finally, to complete all the possible elements of the scrum methodology, we have to talk about the **Events**.

- **Sprint**: Is each (fixed) period of time in which the Scrum Team works on an specified task. Sprints are the most pure base of Scrum and most of the other Agile Methodologies. A Sprint just finishes when the task is fully fulfilled and the next sprint will start immediately when last one gets marked as completed.
- **Sprint Planning**: Is a reunion that usually takes place at the very beginning of an Sprint in order to plan and organize the Backlog and what the Sprint Goal will be.
- **Daily Scrum**: Is a reunion planned almost daily for getting to know what has been done the day before, what will be done this same day and to unlock any kind of drawback appeared. Usually is a no-more-than 15 minutes reunion.

- **Sprint Review**: Is a reunion planned at the end of each sprint to get a general idea of what functionalities have been developed in this sprint. Product Backlog gets updated in this reunion.
- **Sprint Retrospective**: Is another reunion planned at the of each sprint but, in this case, is the Scrum Team which identifies as a kind of analysis to get what went worse than expected and get the points in which to focus a future improvement in effectiveness. Is the event that concludes an sprint.

4.2 Work Approximation

Initially, in this project it was planned to divide the work into 7 sprints (plus extra contingency sprint, 8) of 2 weeks or 3 weeks each.

The initial plans were to start the project, more or less at the beginning of October, so, as to do the initial approximation, the starting date of this project will be **1/10/2020**.

So, the initial approximation table of the sprint list will end like this, taking in account the number of Sprints predicted.

Table 4.1: Initial Sprint Approximation

Sprint	Initial-Date	End-Date
1	1/10/2020	15/10/2020
2	16/10/2020	1/11/2020
3	2/11/2020	25/11/2020
4	26/11/2020	10/12/2020
5	11/12/2020	7/1/2021
6	8/1/2021	30/1/2021
7	31/1/2021	20/2/2021
8	21/2/2021	14/3/2021

- **Sprint 1**: Analyze and study the **OpenCV** environment and getting set up the material resources that will be used in the project. The main objective here will be to study the basics chapters on the OpenCV page, how it really works, tutorials on the most used functions and even learn how to set up an environment. Also, I will receive a virtual machine with the basic tools needed almost ready to work coding on it.
- **Sprint 2**: Analyze and study about the **Toxtrac** tool, how it works, the basic functionalities and understand its code.

  Now, on the environment received from the director of the project, check how the Toxtrac code is settled in Virtual Studio, and study on an overview how the interesting

fields for my job are structured and coded in order to understand the necessary to start working. Also will be necessary to read all the guides, both for Code and for User purposes.

- **Sprint 3**: Develop a new Background Subtraction method apart from the one **Toxtrac** already had.

  Along with the project director, discuss and decide what is the objective method and develop it. Trial and error process carried out firstly in a test environment before putting it in Toxtrac code.

- **Sprint 4**: Study and analyze about Calibration methods, and what users could demand in **Toxtrac** calibration tool.

- **Sprint 5**: Improve and develop new functionalities in Calibration section of **Toxtrac** app.

  Create then new options using for it QT designer, generally looking for the simplicity of the Toxtrac Calibration Tool for the hypothetical user.

- **Sprint 6**: Develop an image segmentation tool using different approximations and choose, by results obtained, the most convenient to polish fully implement.

  First approaches will follow assisted segmentation techniques, if the results do not reach to fulfill minimum requirements a whole new approach using techniques as texture segmentation ideas or background subtraction will be developed.

- **Sprint 7**: Write the dissertation. Along with the writing itself, periodic reunions with the director will take place. Also in order to complete it, there should take place a job of looking for articles or books create a bibliography.

- **Sprint 8**: Contingency Sprint. It is a good practice to plan an extra sprint just if there occurs something unexpected at last minute.

At the end, due to foreign issues, the dates could not be followed, due to the fact that both human resources that participated in the project had other occupations or personal issues that made some sprints postponed. But, the working load, in terms of hours or days worked, were completed as if nothing would had happened.

4.3 Resources and costs

In this section, the human and material resources used in the project will be explained briefly.

4.3.1 Human Resources

In this project just two individuals were involved, the student and the tutor, now I will introduce the roles in which they both took part in this work.

- **Tutor:** The tutor will get a role of **adviser**, checking in all the sprints the progression done and revising the project. Will also help the student in elaborating the project planning and virtually directing it as a Director of the Project.
- **Student:** Will alternate between two roles, one as a **developer** and the other one, as an **analyst**. Will be in charge of the project development itself, studying sometimes and coding others. As the involvement in this kind of projects to the student is so big, it will end being also a Director of the Project along with the tutor.

4.3.2 Material Resources

Hardware:

- **Processor** - Intel Core i7-4720HQ
- **RAM** - 16 GB
- **GPU** - Nvidia GTX 960M

Software: Content of this part is fully explained in **??**

4.3.3 Costs

Finally, to get what this project could cost, is a matter of getting the salaries of the Human Resources and the costs produced of getting all the Material Resources.

Firstly, the Material Resources were all received for free as the computer used was ceded by the own tutor and all the programs or libraries used were completely free.

Usually, the big costs come from the Human Resources values and its hours spent in the project.

In this case was planned a workload of 3 hours per day for both roles the student had.

For sprints 1, 2, 7 and half of 4, the student's role was the Analyst role.

For sprints 3, 5, 6 and half of 4, the student's role was the Developer role.

Due to the issues explained in last part of Work Approximation subsection, Sprint 8 finally counted as working time, will be showed as half part Analyst and half part Developer.

In relation to the tutor role as an advisor that controlled all the project, we will count its job as an average of 4 hours per Sprint.

The salaries taken as reference for each role will be trivial but in proportion, tried to adjust it to the real salary comparison between roles.

Having all this information in account, the **Breakdown of the Project Costs** will be, like showed in table 4.2

Table 4.2: Final Costs Approximation

Role	Hours Spent	Salary (€/hour)	RoleCost
Analyst	186	26	4836 €
Developer	240	20	4800 €
Advisor	36	40	1440 €
TOTAL	**462**		**11076 €**

Chapter 5

Work Done

Usually, in this kind of projects in which an application is being developed or improved as in this case, for a final user, a previous step done before even starting is the requirements Analysis. In fact, big part of a software-product final success comes from this chapter and how it is carried out.

In our case, all the processes related to the analysis or identification of the requirements was done between student and tutor, who already knew his own application (Toxtrac) and what could be needed or improved. Sometimes this step is carried out by a process of interviews or chats between a bigger development team or with a representation of the final client, showing their concerns and needs regarding the application they use.

5.1 Functional Requirements

In Software Engineering, we talk about **Functional Requirements** [35] as any statement that defines any kind of functionality in a system or application, basically, this set of Requirements will let the user know what the system will be able to offer and also in same cases, specifying what the system should not do.

In this project, the Functional Requirements have been defined as:

- **New Possibility** for the user to choose how he wants to perform the **Background Subtraction** on Toxtrac. Originally, Toxtrac, as an already-working application, had its own way of calculating a Background Model in order to perform its tracking functionalities. A new method will be implemented.

- **Develop new tools in the Calibration Module** of Toxtrac. The application will now include new functionalities in the Calibration part in order to simplify its usability for the user, focusing in a graphical point of view of the calibration paradigm.

- **Develop a new Image Segmentation** module from scratch. Focusing in some situations presented in terms of new videos requirements, create a working unit of Image Segmentation that will let Toxtrac in the future work with them, separating the animals from the rest of the frame.

As with these Requirements being set, Use Case Diagrams for each Module could be drawn. Here, we will be able to schematize how Toxtrac's functionalities, related to those modules topics, will be completed among all the improvements we propose at the beginning of this project.

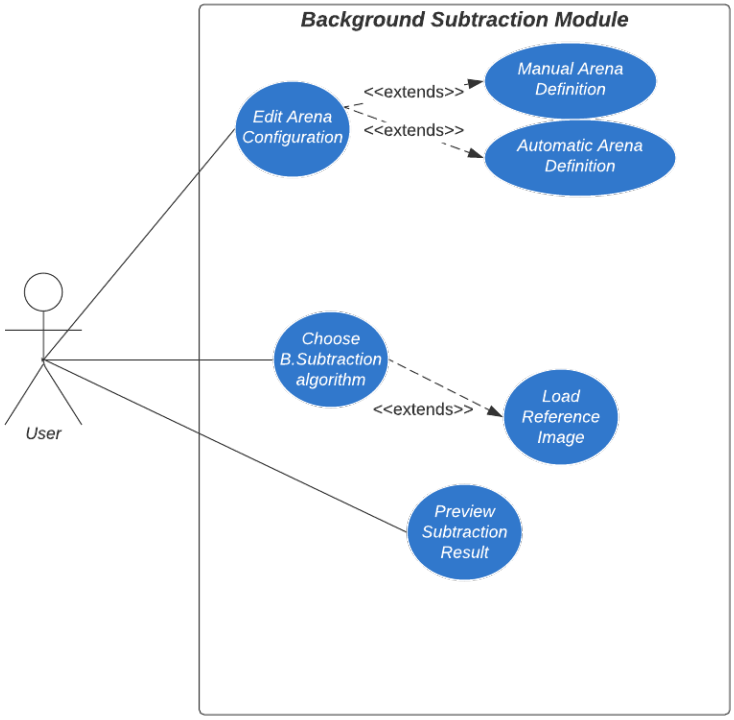


Figure 5.1: Use case Diagram related to the B.Subtraction Module of the project in Toxtrac

Figure 5.2: Use case Diagram related to the Calibration Module of the project in Toxtrac

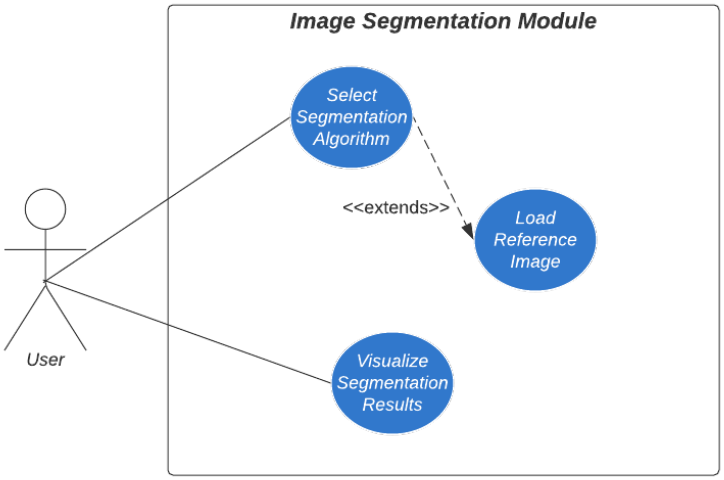


Figure 5.3: Use case Diagram related to the Image Segmentation Module of the project in Toxtrac

5.2 Non-Functional Requirements

Or **Quality Requirements**[36], are those that explain **how should the system perform** more than what should the system do, trying to ensure the quality of it and its use.

The ones defined for this project have been:

- **Intuitiveness** and Usability. As the application should improve in this terms after our job, making it easier for the user to interact with it and to understand how things are being done ”underneath”.
- **Reliability**. Ensuring that everything that will be developed and introduced, will provide correct and clear results for the user, making it a tool to rely on.
- **Scalability and maintainability**. As the capability of making the software ready and easy to fix if it is needed or to modify and add new or already implemented functionalities.

5.3 Background Subtraction

5.3.1 Context and Planification

As explained in previous chapters, Toxtrac is the app used as reference in this work. As an already functional tracking software, it has a Background Subtraction module that it is used

as a complement for the tasks of the application. This B.S. Module works over a method called MOG2, a method based on Mixture of Gaussians.

It was proposed a new version, to complement with this MOG2 method.

The most negative point of this method it already has implemented was that in situations where the foreground object, the animal, stood still for enough time, it will identify it as background and so there will be frames of no-detection.

The new idea proposed to implement was founded to avoid Background Subtraction algorithms that generate a reference model.

In this case, the final user, will have the possibility to choose between Toxtrac's original background subtraction method and this new one, in which he will have to upload to the program an image that is going to be used as Background Model itself. By this approach, we get a complete different operation for reaching the same objective as with the MOG2 method but, leaving out algorithm-generated Background reference images.

First of all there has to be put into context what kind of videos does Toxtrac work with in general, and show up this example in particular. Videos will be, ideally, sets of one or more arenas recorded in concrete situations as mostly constant illumination, an homogeneous background colour and recorded positioning the camera as in an static overhead shot.

In this next example in particular, the video sample will be a recording of four arenas at the same time, with one **ant** in each arena. Thanks to the system implemented in Toxtrac of manually selection the most interesting arena of the video will be selected for this process. (It also implements by default an automatic arena detection system but, in this case, it is less interesting, to speed up this process),

Basically, here we are pointing out the interest in the arena where the animal makes more movement through the frames that get analyzed, so that the position changes are important enough to differentiate it better.

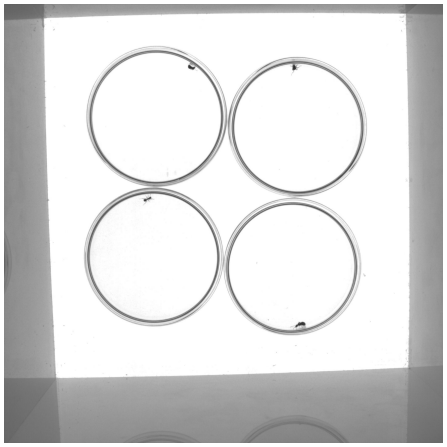


Figure 5.4: Example of video recording used as input.

Figure 5.5: Selected arena using the Toxtrac's manual selection tool

5.3.2 Implementation

Images are represented in OpenCV as arrays (Mat Objects) that consist on rows of pixels, being each pixel color represented by another array of values.

Again, the idea proposed to implement was, theoretically, to use a selected background image as Background Reference Model. Then, this method would work as for each image-frame of the video thrown as input, calculate the difference against the Background Image and calculate as mask from this saved difference.

As an schematic point of view, this diagram 5.6 will show the Architecture of this method,

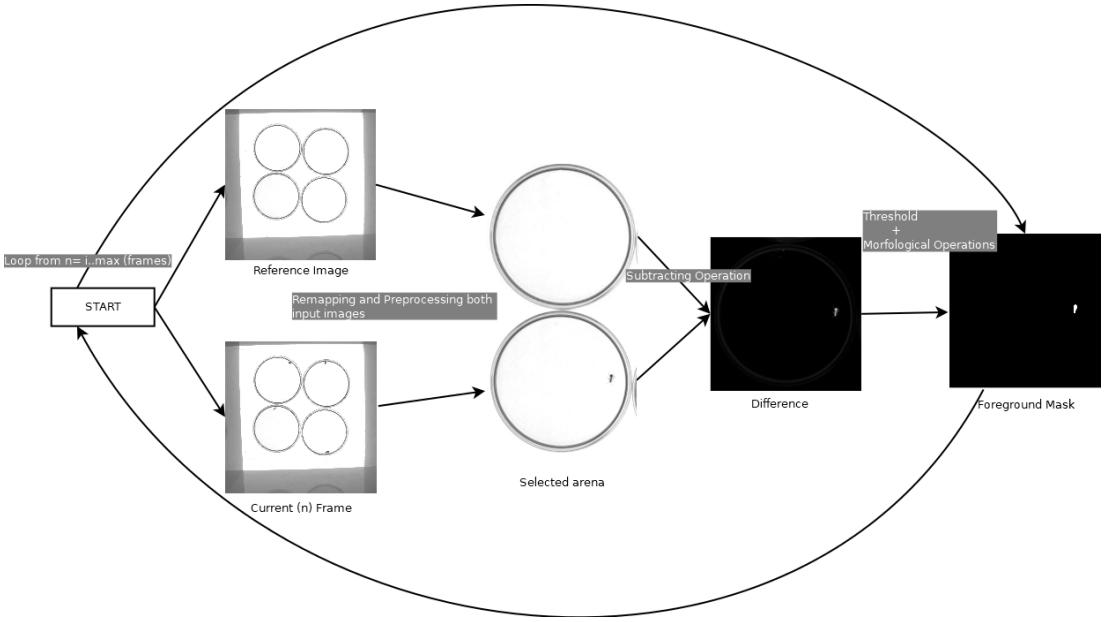


Figure 5.6: Diagram on the function of the new Background Subtraction Method

So, taking in account this idea, there has to be decided what kind of operation gets computed at the point of calculating the difference. We came out with two operators valid as options, **subtract** or **absdiff** operators.

Both functions on OpenCV do really work in a similar way, subtracting one array from the other, element, by element in both, with the particularity that in absdiff what it gets computed is the **absolute difference**.

In this context and checking both performances, the option decided to use was the absdiff operator.

So, next, there has to be clear that in both operations mentioned, the difference is calculated by the subtraction of both arrays **element by element**, in this case, the Mat types that represent both images in each operation.

Absdiff needs to calculate the absolute difference of the two arrays to be from same type and size. And works like this:

$$dst(I) = saturate(|src1(I) - src2(I)|)$$

Theses are the reasons why, as in Toxtrac, images from the video input receive a previous treatment before operating with them, this same treatment steps have to be done also to the background reference image.

Firstly, as the operation will be done between two **gray-scaled** images, that conversion has to be done by the $cv :: cvtColor$ function turning the Color Model from $RGB2GRAY$.

After that, the next treatment to the background image will be using the $cv :: normalize$ function, where it gets secure that both images will be defined on the same type and mapped in the same range according to the interval set by the alpha and beta arguments on the function itself, and taking into account the normType value also, usually (as in this case) $NORMMINMAX$.

Also, as when selecting the arenas in Toxtrac, in this case manually, we re-calculate which arenas are the interested ones, this remapping has to also be done to the background image, in order to follow this reference that will be used in the whole video frames.

In this case, for the scenario in figure 5.7, the background image used will be this one below, in detail.

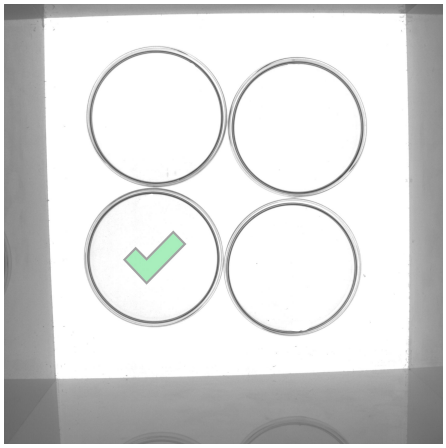


Figure 5.7: Background used in this example with the arena that is going to be selected.

Figure 5.8: Background image after remapping function just getting the wanted reference to subtract

So, absdiff will be used now to get the difference in each frame between the background and the frame that corresponds, the result 5.9, is a highlight of the ant position in each frame.

As Toxtrac has a working pattern, the result can not be left as it is, so now, with the animal highlighted in every moment, it is time for **segmenting** the image in order to generate a mask out of this process.

Firstly, a threshold is operated over the just obtained result and finally, by using mathematical morphology we refine the result obtaining the wanted figure 5.10.

Figure 5.9: Basic result of the subtraction itself

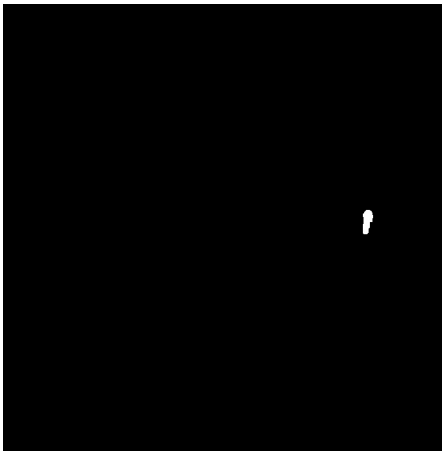


Figure 5.10: Generated mask after segmenting the previous result. (Same Frame)

5.4 Improving Toxtrac's Calibration Module

5.4.1 Context

In this module, as explained before, the job was focused first in analyzing and comprehend how Camera Calibration worked in general in Computer Vision environments and so in Toxtrac in particular, then I had to develop new functionalities or modify some that were already in, all related to the User Interface in the Calibration area of Toxtrac.

Originally, Toxtrac's Calibration tool was given with just two panels. The first panel 5.11 was for making the user able to perform a Calibration just using a Calibration Pattern as the typically used of an image with a Chess Board. In the same screen the user could upload an image and set the parameters related to the pattern on the screen itself such us the number of rows, columns or if it was the case of the Chess-board pattern, the size of each box. Also, the user could apply the calibration then in that same screen or reset it.

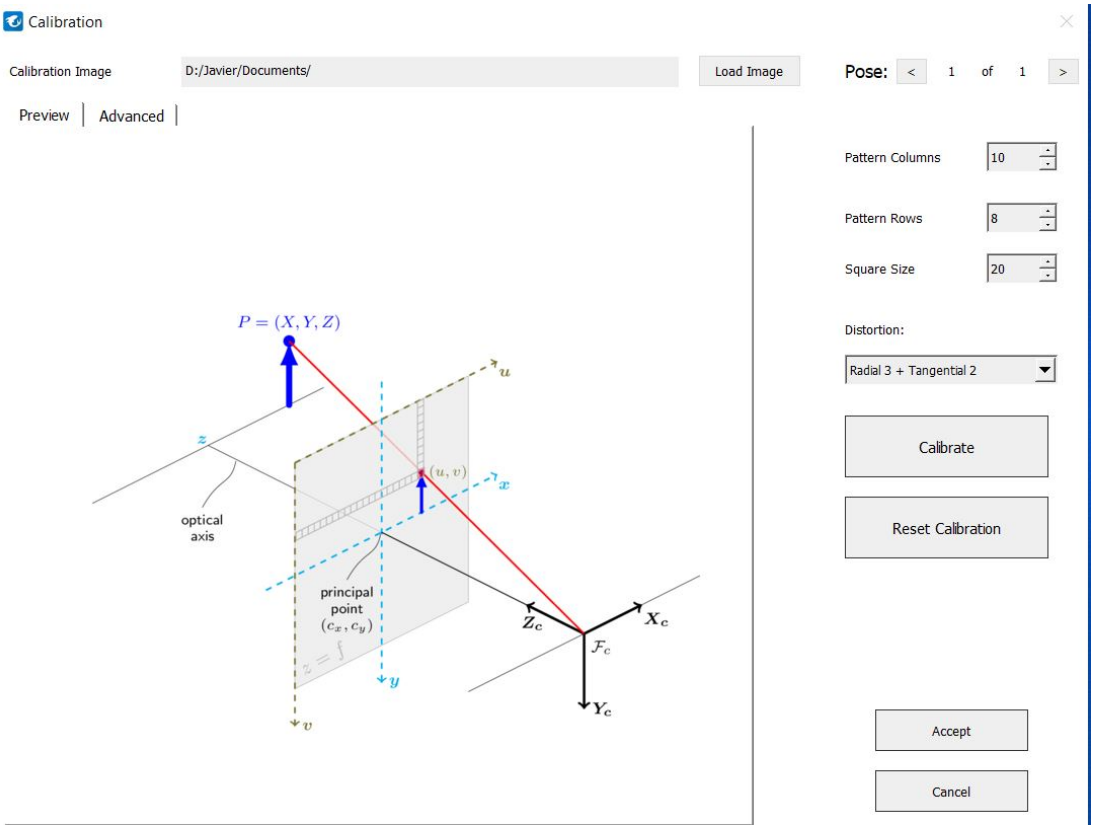


Figure 5.11: First tab in the original Calibration Tool, for Pattern-based Calibration

In the second panel of the Toxtrac's Calibration tool, the user had the possibility to introduce manually and directly in the matrices the values needed to configure the Camera Model, this panel was called Advanced panel and with the objective of using it if the user lacked of a Calibration Pattern to use or if he knew already the values to introduce.

This tab 5.12, so, was formed by an editable Camera Matrix, where the user should edit the parameters f_x and f_y of it related to the millimeters-to-pixel scale (horizontal and vertical scale in that order). Also, there were inputs to edit the extrinsic parameters such us the Rotation Matrix (in order to determine the orientation of the camera) and the Translation vector (in order to get the relative position of the camera).

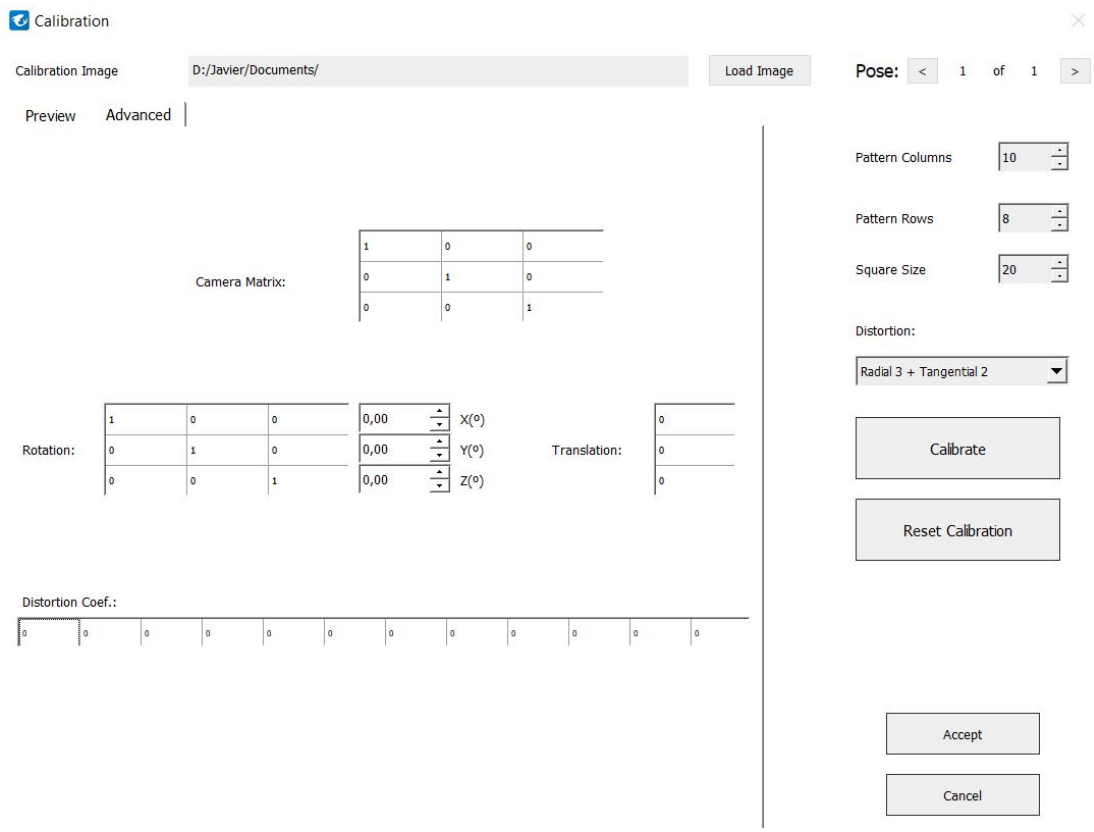


Figure 5.12: Second tab in the original Calibration Tool, introducing Manual Calibration

5.4.2 Planification and Implementation

First idea thrown, was in a direction focused to improve the user experience, making the Calibration Process clearer and easier to understand.

Firstly, the basic idea was to create a **Simple** 5.13 new tab in which any new idea that will fulfill this last statement would be included. Starting from the inclusion of two options that will let the user to edit those Camera Matrix parameters just explained, f_x and f_y, in an easier way.

The difference between this first two options was just the way the relation of the pixel-mm scale was seen, if the idea was to introduce how many pixels fitted in a millimeter or the other way around. The point here was not just adding this two extra boxes for each option, but also make clear, for inexperienced users, what did each of the f_x and f_y mean, as in a written small legend along each text box, because, of course, the edited value, once, saved would be also updated in the Camera Matrix itself in the Advanced Panel.

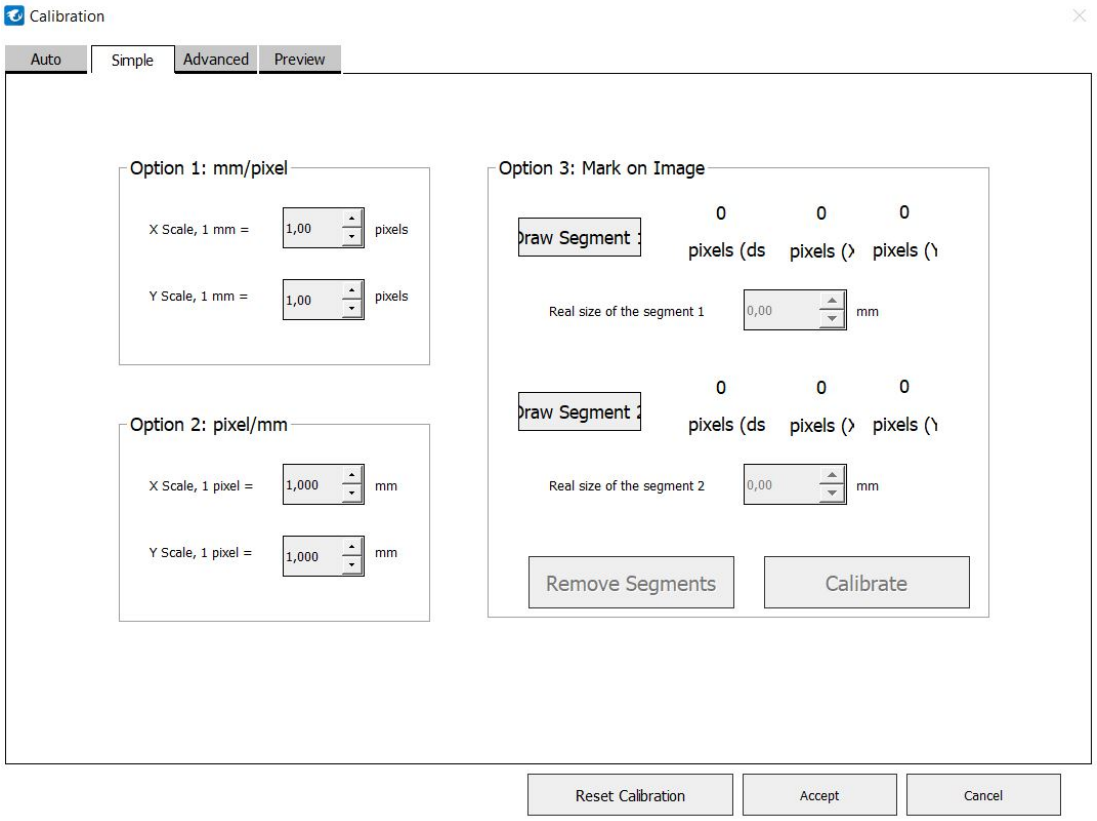


Figure 5.13: New Simple panel in the Calibration Tool.

In this same tab, another, third, option was included, called "Mark on Image" 5.13. In this new method, we add a new way of selecting the f_x and f_y parameters in the Camera Matrix. Now, it was implemented two buttons (one per parameter in the Camera Matrix) that will open the user a new window with a Reference Image from the video (a frame) that the user chose to analyze.

Now, the point of this option will be to draw two segments (one in each new window) by clicking in whatever two points the user wants 5.14. User has or, at least, should draw segments in which he knows its actual length in the real world in millimetres. So, then, by closing this popped-up windows where the segments were drawn, the length in pixels will be showed in the indicators (Absolute distance, distance in the X distance vector and distance in the Y vector). Then its just job of the user to measure in the real world the selected vectors and complete the boxes down below in order the program can calculate the parameters on the Matrix.

Figure 5.14: Example on how segments are drawn, highlighted for clarity in this picture with red ovals

Once clicked in the **Calibrate**, the user will be able to see 5.15 the new parameters both in the Option 1 and 2 of this same tab and also in the Advanced representation (on the next tab) of the Matrix itself.

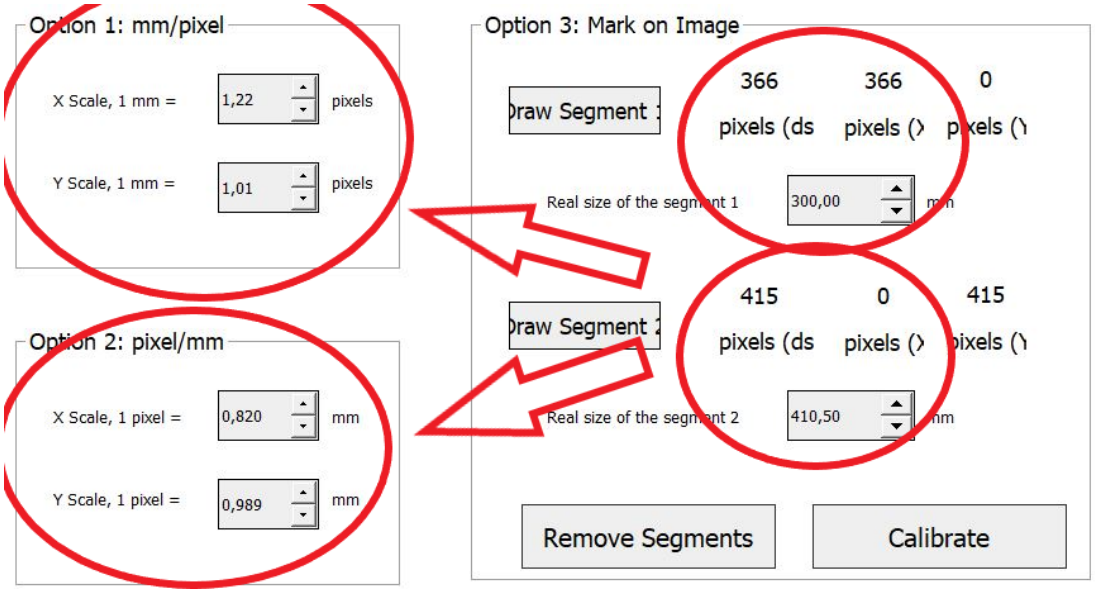


Figure 5.15: Example on how the values get updated after writing the real-world size and clicking on Calibrate.

It is true that in this simple example, each segment just carried with the calculus of just one of the parameters as they just move in one of the cardinal edges at same time, not being oblique but in really, the writing of both segments help on the calculus of both of the parameters.

Next as it could be seem slightly in the 5.13 image, another tab was added to the Cali-

bration tool in Toxtrac. The original Preview tab suffered a rename to Auto, as seemed more accurate and a new an actual Preview tab was added.

As it was thought that with just numeric changes in the Simple or the Advanced (in a more direct way) tabs could still not be intuitive enough, with this new **Preview** panel 5.16 whatever change done in the just mentioned options, could be reflected graphically.

Now, after selecting a frame from the input video as Reference Image, the user has an option to show an hypothetical scaled Grid. This Grid, that firstly can be adjusted its own base side, will react to any change written in the Camera Matrix. This is done in order to let the user have another help input, in this case graphical to understand better what is the real effect of one or another parameter on the camera intrinsic settings (Camera Matrix).

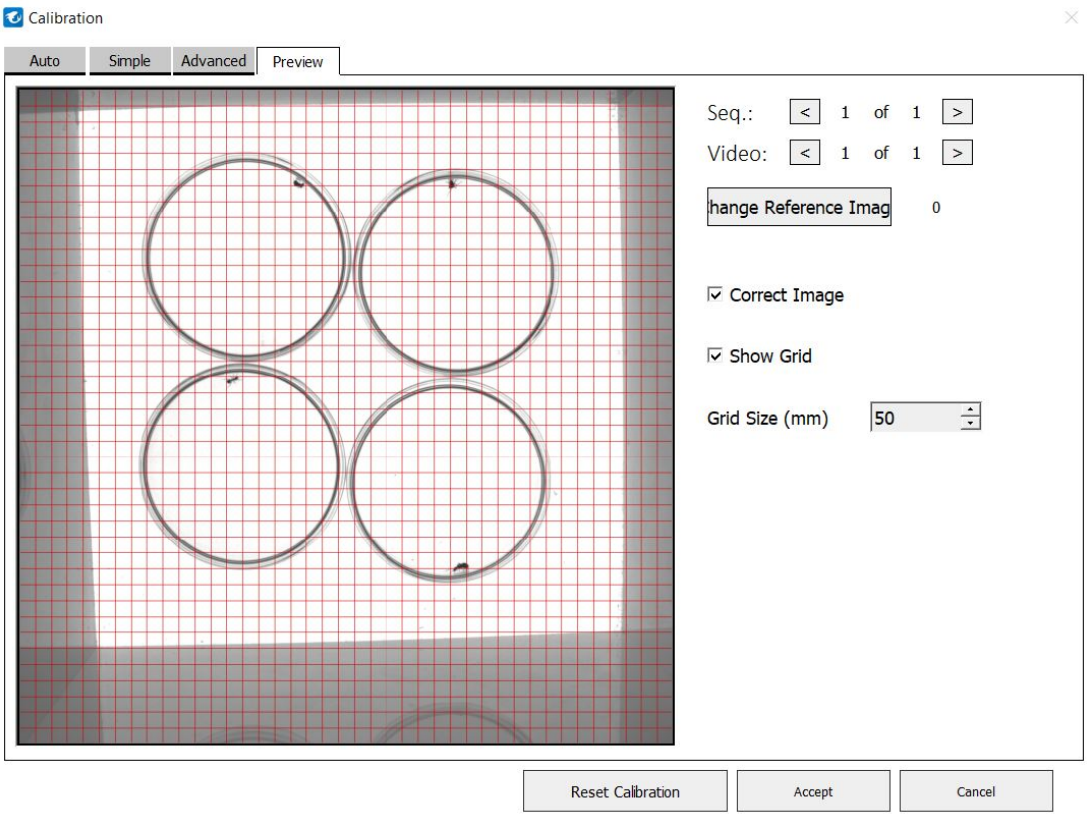


Figure 5.16: Example of the new Preview panel with the Show Grid mode ON.

Additionally, one of the main points on this new Visual tool is that the user is let now to see what would happen in **real-time** when he changes a Extrinsic Parameter of the Camera as the Rotation Matrix. The Rotation Matrix, is a 3x3 matrix R that, as said briefly before, is a big part on the camera setting and will define partly (along with the Translation vector) the absolute position of the Focal Plane in the coordinate world, and basically the orientation of the camera.

So, as in Toxtrac is available to set as the user wants the Rotation Matrix due to the Advanced panel, this down here 5.17 will be an example on how this rotation change will be seen in the Grid. Toxtrac will also let the user to change both Extrinsic and Intrinsic values and experiment with them at the same time.

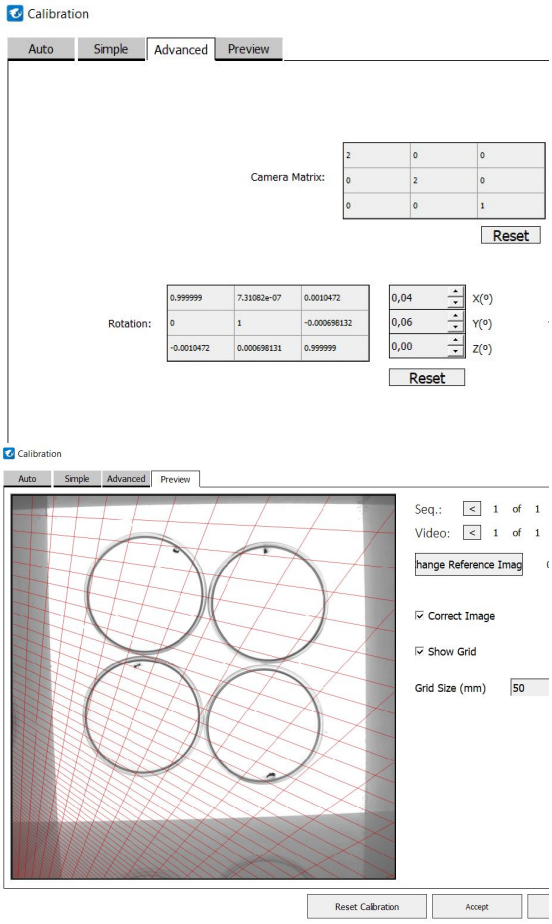


Figure 5.17: Experimenting changing both Camera Matrix and Rotation Matrix along with the real-time result shown in the grid.

Finally, in this section were also made some minor changes on the old interface itself as just leaving the buttons needed in the now-called Auto tab in that panel. Also, now the panels

have all their functionalities embedded in boxes and in each tab out of those boxes, just three buttons remain isolated and fixed for all tabs, **Reset Calibration**, **Accept**, and **Cancel**.

5.5 Image Segmentation

5.5.1 Context and Planification

Toxtrac's development team realized that, based on a new set of videos, the tool was not prepared for them, not being able to detect the animal on the samples, in this new scenario and with the already coded techniques.

These new samples were completely different to the old scenarios like the ones used in the previous chapters of this dissertation, not letting the program to be completely functional making it possible to track the animal in these cases.

In the previous cases, we had big similarities on how the samples were recorded or on how the scenario was built and filled. As some examples, the camera angle or the homogeneity on the background of the arenas.

On the other side, the problem resided now on the irregularities in the illumination of the arena through the samples, that was not longer an arena but a kind of cage. This cage due to the camera angle of the recordings, had its walls visible, causing varieties on how the light reflected on these metallic walls, being this a problem in terms of Image Segmentation.

The second problem found here was the relation between the animal now recorded and the background underneath it. In this case the animal is a mouse of a kind-of white colour. The problem resides in that this mouse is now moving over a metallic gate that under it had something like sawdust, of a colour really similar to the mouse one. This is seen perfectly in the example down below 5.18.

Figure 5.18: Example of the new scenario

The objective now was clear, knowing the new scenario: Elaborate a full segmentation module that is able to fulfill entirely the segmentation needs to these new and more demanding videos far from the criteria and characteristics of the old ones.

5.5.2 Implementation

Approach with Assisted Segmentation

The first approach thought to reach the wanted solution was to investigate on the assisted segmentation methods, using them exclusively, thanks to the algorithms that are being used in them.

In this case, I thought about the **Watershed** proposition, because it was one of the most popular options inside this assisted category, and it is usually useful in situations similar to the one we are facing now, with elements with big similarities between them to segment.

Now I proceed to explain step by step what has been done to reach the solution in this approach.

First of all, and operating frame by frame, a **Distance Transformation** should be performed. With the obtained result and applying it a normalization, we pass it through a threshold so then we can calculate the different peaks or markers needed by the Watershed algorithm to work properly.

We should take in account what does assisted algorithm mean, needing the help of some kind of input to do the job, in this case, markers from where the different parts of the segmentation can start spreading.

Now, it is just a job of executing the algorithm with those parameters obtained and show and analyze the final result. To make it clearer, it is usual to select a set of different colours to use as in the set of segments created in the resultant image.

Checking the result we quickly get that what we have gotten is not enough for the requisites we were asked, lacking hardly from precision , as shown below in figure 5.20. That is why we should continue searching for other technique that fulfills the original demands.

Figure 5.19: Original video frame to segment

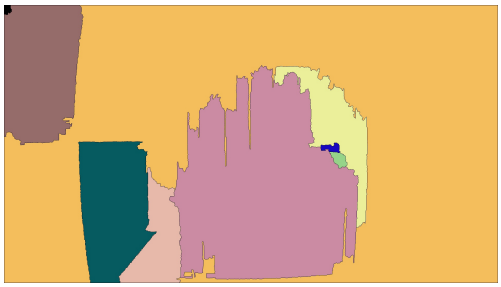


Figure 5.20: Watershed segmentation on that same frame

As we see, we can deduce that, although there is some kind of figure in the center of the image in the resultant segmentation, the generated figure is far from desirable, no matter that it detects parts from the animal.

Here, we have to point out that, even it can seen as a big problem that there exist segmented zones in parts for example that correspond to one wall of the gate, it is not that big issue at all, due to the fact that those unnecessary parts can be easily discarded with other techniques.

The point here that really makes this result to be discarded it is exclusively the lack of precision in the segmented part that should correspond to the mouse, mixing in the same colored area parts that truly fit to the mouse with some that are just part of the surface of the gate (including those bars and sawdust we mentioned before when introducing the scenario, that were one of the priority points that promoted this investigation). As the solution is not enough, we should start thinking of discarding this path.

Obviously, this frame was an example of how Watershed worked, and the rest of the frames of the video over this approach are not included in this report, but of course, this problem persisted in greater or lesser extent in the hole video.

Before completely moving on and discarding this idea, there had to be taken in account and try other alternatives over this same experiment to see if with any kind of variation the solution could be reached.

For example, the function used in the calculation of the **Distance Transformation**, that helps in generating the reference points for Watershed to work, had a point of maneuver, trying different ways of calculating the Distance itself, changing the formula.

The **original method** works using the **Euclidean Distance**, represented in OpenCV arguments with the **DISTL2** option.

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}$$

We have two extra options, with the options **DISTL1** that performs this equation:

$$d(p,q) = |q_1 - p_1| + |q_2 - p_2|$$

and **DISTC** which works as:

$$d(p,q) = max(|q_1 - p_1|, |q_2 - p_2|)$$

Where **p** and **q** are the the two points in the real plane from which we want to obtain the distance from each other. In different tries calculating the Distance Transformation with all

these different equations, no significant improve was perceived.

Other option, instead of changing how the operation is itself, was trying to modify a the operating. This required on modifying the frame image from which we were working either with different types of thresholding or either **Morphological Operations**, for example.

Even taking all this alternatives in account, or mixing them, the final result was still far from the desired precision, so we decided to change drastically the path to follow.

Approach using Texture Segmentation

After this approach with assisted techniques, it will now be followed a new method based on the idea of Texture Segmentation. This is the process in which an image is segmented into regions with different textures containing similar group of pixels.

First of all is that we have to find a way to modify/filter the image input based on its entropy values, helping by this way to show easily the differences between all the various elements that compose the scenario. As shown in figure 5.21, always referring to the frame used before as a example in figure 5.19.

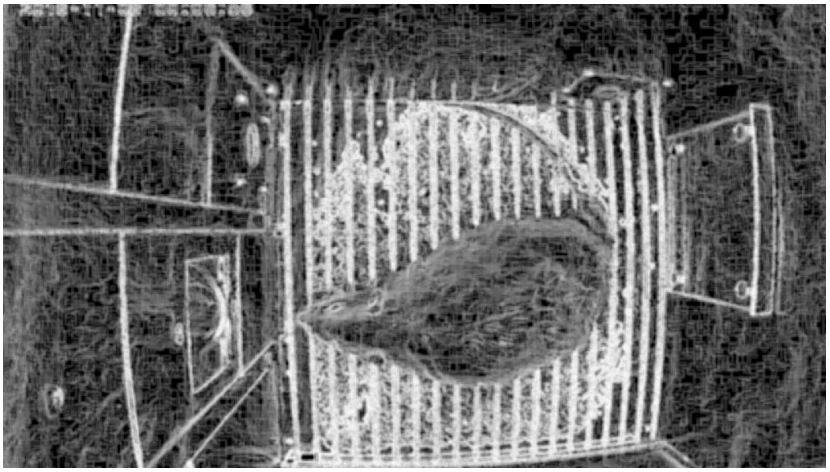


Figure 5.21: Example after treatment with the function of the entropy calculation.

This result, will be pretty useful for, with a thresholding operation, (figure 5.22), generate a mask with which elaborate the segmentation.

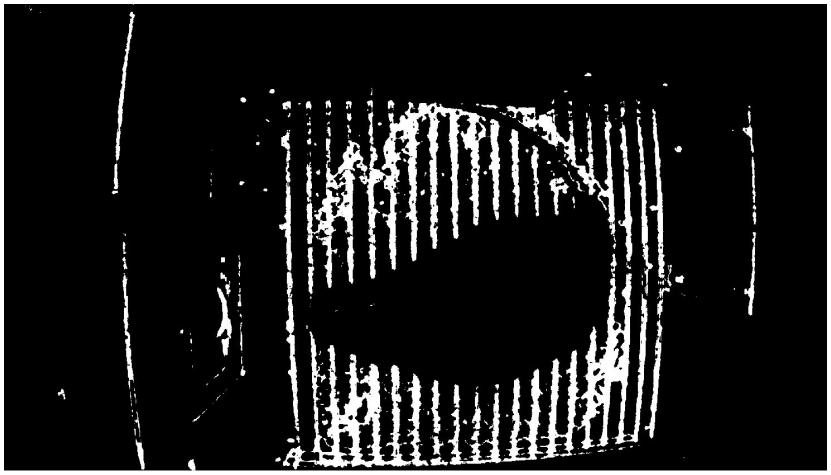


Figure 5.22: Example after thresholding last result.

Watching the thresholded result, we can see a big number of impurities that will difficult our procedure of using last result as a mask. That is why we should use a function, that, originally, is only available in **MatLab**), called **bwareopen**.

With a not-so-complex implementation in C++, bwareopen will set a value **P**, and with **findContours** function, will find all blobs or independent group of pixels for proceeding to only draw those areas with a bigger area than value P, helping, by this way to make those impurities that we were talking about that were complicating the process. Also, if any of those blobs found is big enough and has any kind of hole inside, this function will be work as a filling tool and will make our task then, easier.

Here in the next figure 5.23, the result seeing both kind of assumptions solved.

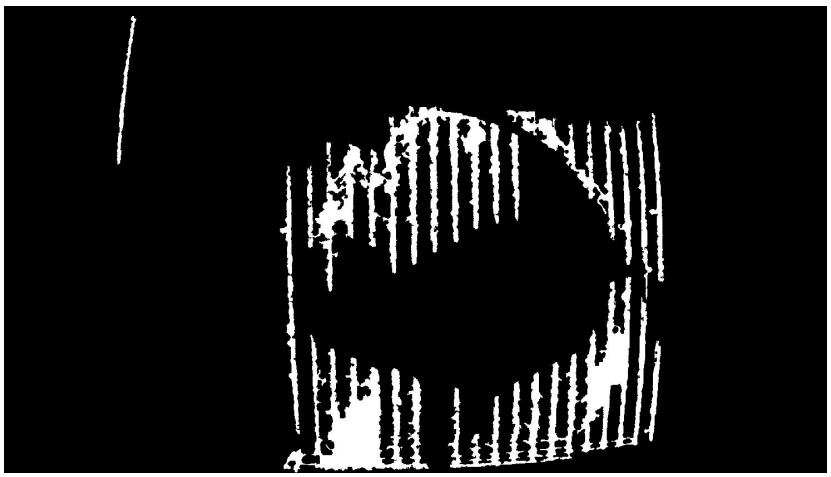


Figure 5.23: Example after filtering by size.

After obtaining this last result, it can be seen that in big part of the image and mostly near the bars of the surface of the gate there are loads of irregularities without any defined shape, as shown zooming in figure 5.24.

An approach to solve this issue is helping us from the idea of morphological operations, in this particular case, we will work with the **CLOSE** operation, that is similar to summing up two other operations, first executing an **ERODE** and over it a **DILATE** operation.

Clearly, seeing the result in figure 5.25, this issue was perfectly solved.

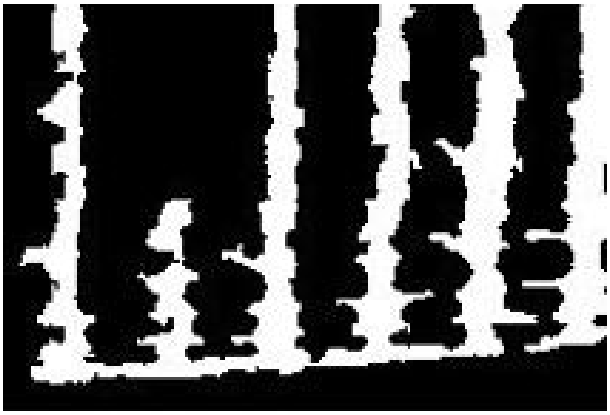


Figure 5.24: Zoom to one of the image sections where the commented irregularities are most evident

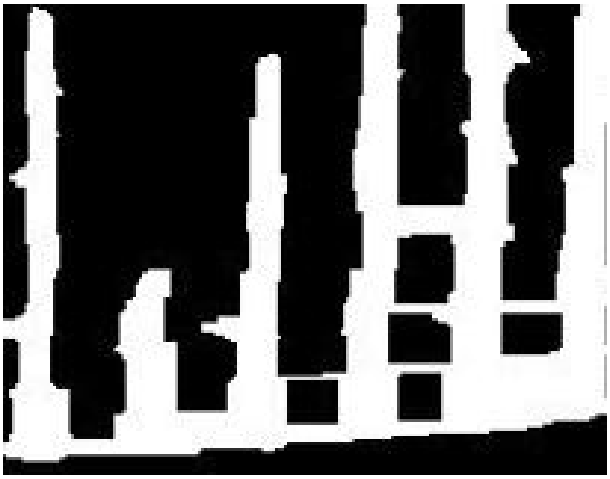


Figure 5.25: Same section zoomed up after the closing operation

After this last treatment, shown at full scale in figure 5.24, the next step now should be to fill all the holes that appeared. With this transformation making the edges more straight, will be much easier.

For this task we will again make use of a function that exists originally in MATLAB, **imfill**, without directly translation in C++, but which idea will result much useful to solve this kind of situations.

Basically, imfill 's concept is that, starting from a pixel (0,0) of an image, the first step is to execute the function **floodfill**, that it is gonna be used to change the value of all those pixels that are found from black to white until finding with one that it is not black, working like water filling a surface and leaving then the holes that are fully surrounded by white elements, so at the end of the execution of this function, the only parts that are still the same, are those holes.

After this execution, the values of the pixels get inverted, from white to black and vice-versa.

Finally, we just have to combine this last inverted color result with the original one from which we executed the **floodfill** function.

The change between the original image, shown in figure 5.26 and the result after this filling process, shown in figure 5.27 is more than evident, and can be interpreted as a success in this path.

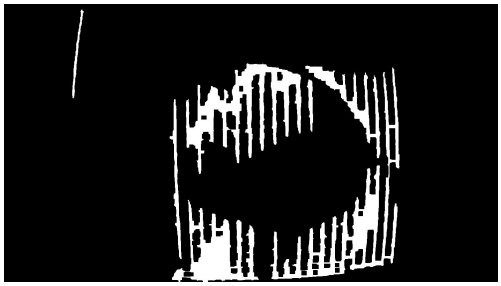


Figure 5.26: Full Image after the recent Closing Operation

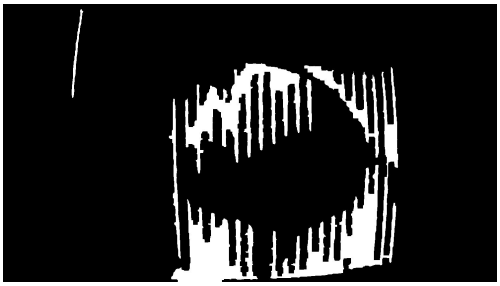


Figure 5.27: Image after the treatment with the filling function

Once finished this treatment we continue on other side, going back again to the original frame and applying a threshold on it (after some testing, the exact threshold was found). The previously obtained result in figure 5.27 will be used as a mask, subtracting it to the recently thresholded frame. (figure 5.28)

Figure 5.28: Result after the mentioned subtraction.

After this operation in order to complete the cleaning of the image imperfections caused by the color similarities between the animal and the material in the surface underneath it was decided to return to one of the original ideas in this project, the background subtraction using an image as reference to operate. So, first of all, with the background image already obtained, lets proceed to change it color-space to gray scale and now threshold it. After it, subtract with the previous result, getting as result the consequent image in figure 5.29.

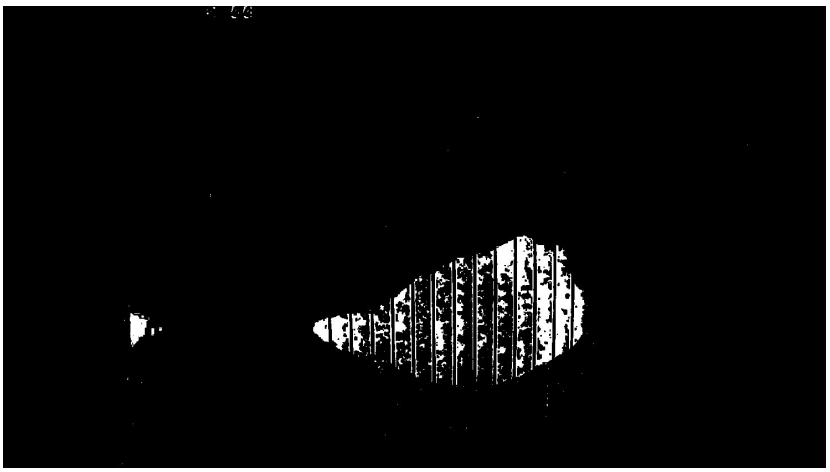


Figure 5.29: Result after background subtraction.

By seeing the resultant image, we get that the subtraction just done gets rid of the parts we were wishing to erase perfectly but, as it is a subtraction, it also made some parts of the animal disappear. This it is not a problem thanks to the using of **Morphological Operations** we can ”re-draw” completely the animal getting a totally reliable result as shown in figure 5.30.

The operations used where a combination of **dilate** and **erode**. Dilate executes a morphological operation (dilation) which its result is to gradually fatten the boundaries of an object.

On the other side, Erode executes an operation (erosion) which erases pixels of the boundary of the object, shrinking it.

It is dilate, as seems clear by the explanation of the operations, the operation that helps in completing the animal figure while on the other side erode will help erase the small group of pixels that are still left in the frame and are not desired in the result.

Figure 5.30: Result after morphological operations

Now, finally, after all this operations, we have gotten a result that let us identify clearly where the animal is with its full figure filled, so, there are no more subtractions or any kind of editing in the resulted frame.

The only task left is to identify which one of all the figures that might appear in this stage of the process is the desired one to get shown.

In this frame we were using as example, luckily there are still two blobs left we can use to express what we are going to do graphically. The idea to solve these possible scenarios of reaching the final stage in the processing with any kind of blob or undesired spot is to identify all the objects that appear in the frame and just get the biggest of them all, supposing that it will always be the animal, other valid approach would be by setting a value that represents an area, from which we can assure all the unwanted figures in the frame will get filtered except from the animal.

This was done with the **findContours** function. This function by finding the contours of the different objects in the frame, will identify them with an ID so we only have to loop all the ID's found and save, in every loop, the number ID of the biggest object, known thanks to doing the measurement of the area in each object found.

Finally, with the function **drawContours**, it gets drawn the biggest object found, by area, and it just leaves visible the biggest blob found that, if everything goes well, should be the animal.

Down here, in figure 5.31, the final result just showing, as said, the mouse. Right to it, in figure 5.32, a comparison between the original frame before all this work and the final result.

Figure 5.31: Final Result, just showing the animal

Figure 5.32: Comparison between the frame and the final result.

At the end of the process it is time to show up in a more schematic way this whole algorithm developed with this flow diagram down here.

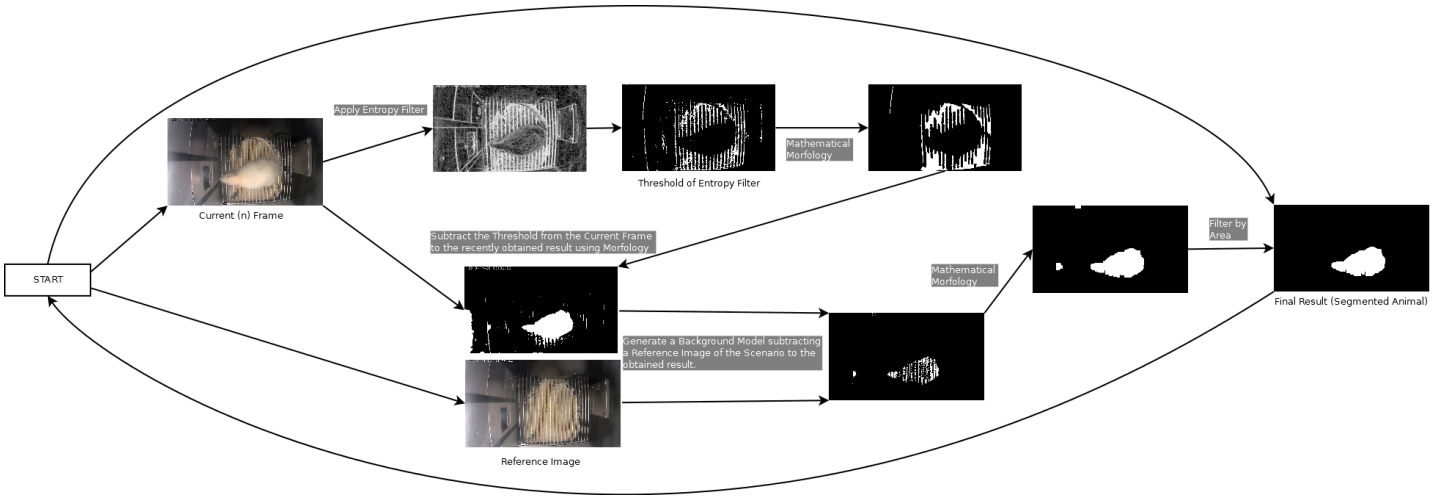


Figure 5.33: Diagram on the work of the Segmentation tool developed

5.5.3 Measuring Results

Taking into account that the obtained result reached at the end was the desired when the problem was proposed at the beginning, now it is time to see if it is not only question of a determined frame and if it works with a full video identifying in most of the resultant frames the animal (only the animal).

The test video used is a set that will give us enough quantity of frames to analyze the results precisely. Talking purely about the content of the video in order to help the understanding of the circumstances, we will found a mouse in a kind of cage in the same conditions explained above that were causing us problems to segment it from the rest of the image, with different light condition, due to reflection in the three visible walls of the cage. The cage has a surface formed by bars as seen in the images, separated all by the same distance each other. Under this bars, there will be some kind of sawdust of a yellowish tone, similar up to some point to the color of the mouse itself. The mouse, in these seconds of video, will be moving in different ways from simple movements of ”walking” all over the cage, to a few tries of climbing some of the walls of the cage, in these tries, the mouse will keep itself for a few frames as if it appear to be standing, changing drastically the figure of the animal, just talking in terms of getting the idea of what is in the video that we are segmenting.

For ease the analysis and get a pure idea of what is the meaning of the figure we get, and how fully and faithful is it related to the animal shape, I am going to save the frames to analyze with the original frame mixed to the result itself, like in figure 5.32.

After letting the treatment work for the full video, we get a full result of **236 frames**. Out of all these examples, we encounter with some cases of a False Negative situation.

We talk about a False Negative situation when in a diagnosis of any kind, there is no recognition at all of something that, in reality, is there. So in this case, False Negatives appear when, in the result frame the biggest detected element in the image is not the animal, for whatever reason this is caused. Below, in figure 5.34 I show a Frame where the blob detected comes from treatment received by the numbers from above the frame, because somehow, the animal was omitted.

Figure 5.34: Example of a False Negative situation

A continuous appearance of False Negative could be one of the worse signals in this kind of job so is the situation to avoid, more than other issues like the quality of the figure segmented, for example. Doing a calculation of the results obtained, the number of False Negatives is pretty reduced, appearing in just **15** of the 236 resultant frames, which represents that the **93.65%** of the set has been a satisfactory result for our purposes.

Leaving away these problems found in the result diagnosis, I have to say that it works perfectly almost around the video getting the segmented image of the mouse in a pretty high percentage of its actual shape in most of the frames, leaving just discarded small boundaries on just one side of the animal that depends on the frame and never changes the right detection of the position of the animal and always discarding the tail of the mouse.

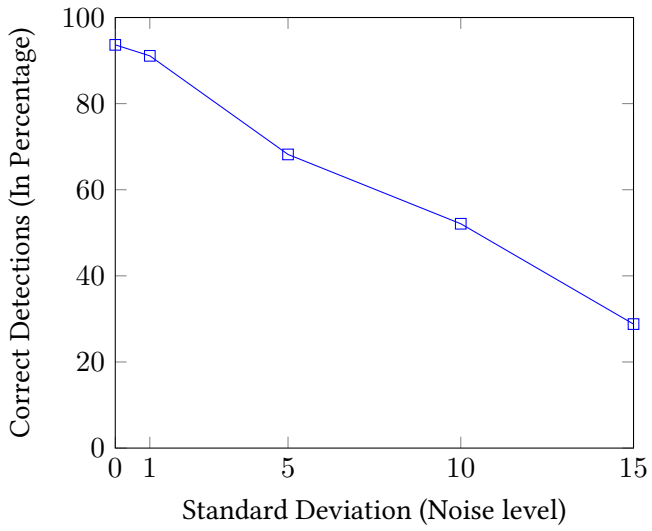
This issue of the tail was ignored since the beginning of the work because, in practical terms, it was not seen as a priority, taking always first the idea of getting the right position of the mouse and only getting its figure, discarding the rest of the elements in the picture.

In order to test the reliability of the algorithm, the next step is to probe it over the video but this time, applying noise over it, in order to make it more difficult to the algorithm to work and see, using different noise samples how it gets less effective in each try.

The noise applied to the video was created using a function that fills a Mat object with random numbers but following a normal distribution, this is the $randn()$ function, where there have to be set specified values for the mean vector and for the standard deviation matrix. After generating this noise matrix, it will be added to the Mat object that represents each video-frame, and securing that the hole the video gets spoiled equally with the same noise for each frame in the experiments.

In this case the experiments where carried out using 0 as the mean vector in the function and increasing in each try the standard deviation matrix value, so that it gets spoiled in

crescendo.



	Standard Deviation	Correct Detections	Accuracy(% of Correct Detection)
Original	-	221	93,65
Experiment 1	1	215	91,1
Experiment 2	5	161	68,2
Experiment 3	10	123	52,1
Experiment 4	15	68	28,8

Table 5.1: Results Overview of the Experiments done by applying noise.

Again, all considered False Negatives frames, detecting other parts seen in the cage but not the animal (while the animal is there), were classified as ”non-correct”.

Chapter 6

Conclusion and Future Work

This chapter will be used as the end of this dissertation, pointing out firstly the learning obtained by myself in the realization of this project and finally talking about how could this work be continued in the future.

6.1 Conclusion

The project originally was planned as a way to introduce and work into the OpenCV library and Computer Vision in general, planned to do it while creating or improving functionalities in the context of the tool **Toxtrac**. As seen, in this project, concepts and functionalities have been touched from different fields of Computer Vision, as Background Subtraction techniques, Camera Calibration and Image Segmentation and going from learning the most basic points of each one, to being able, at the end, to even create usable options for a real tool like Toxtrac.

Also working by phases in a real application environment, that is available online for anybody to download and use it, has been useful for me to get my first notions on how ”real” projects are made, the size and complexity of a program of this dimension and even check what documentation (and how is it) is needed to create a software application along with the software itself.

Finally, while the realization of this job, not only have I learned new concepts or experiences in a real project as just pointed out, also it has been useful to remember and re-study some ideas touched in subjects I had during the degree as the project management terms, analysis, and even coding.

6.2 Future Work

First of all, it has to be pointed out that, as I was working in the environment of the real application Toxtrac, any kind of possibility of including new options or improving what I have done, resides on the decision of the real developers of the app.

In my opinion the most basic way to continue in a going-further process, would be to continue trying different methods in the Background subtraction and Image Segmentation, and implement the ones that end seeming useful so that they can stay as new options to be chosen by the user in future stable versions of the app.

In a personal point of view a good way of using the knowledge obtained in this project, could be trying this computer vision elements in other context, far from the Toxtrac app and check if I would be able to develop similar technologies in other typical fields of Computer Vision as Human recognition or Traffic video analysis.

Bibliography

[1] T. S. Huang, "Computer vision: Evolution and promise." [Online]. Available: http://cds.cern.ch/record/400313/files/p21.pdf

[2] M. Piccardi, "Background subtraction techniques: a review," 2004. [Online]. Available: http://profs.sci.univr.it/~cristanm/teaching/sar_files/lezione4/Piccardi.pdf

[3] T. D. Christopher Richard Wren, Ali Azarbayejani and A. P. Pentland, "Pfinder: Real-time tracking of the human body," 1997. [Online]. Available: https://pdfs.semanticscholar.org/e182/225eb0c1e90f09cc3a0f69abb7ac0e9b3dba.pdf

[4] M.-H. Hung, J.-S. Pan, and C.-H. Hsieh, "A fast algorithm of temporal median filter for background subtraction." *J. Inf. Hiding Multim. Signal Process.*, vol. 5, no. 1, pp. 33–40, 2014.

[5] B. Lo and S. Velastin, "Automatic congestion detection system for underground platforms," 2001. [Online]. Available: https://www.academia.edu/14262757/Automatic_congestion_detection_system_for_underground_platforms

[6] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *IEEE transactions on pattern analysis and machine intelligence*, vol. 25, no. 10, pp. 1337–1342, 2003.

[7] W. E. L. Grimson, C. Stauffer, R. Romano, and L. Lee, "Using adaptive tracking to classify and monitor activities in a site," in *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231)*. IEEE, 1998, pp. 22–29.

[8] Z. Zivkovic, "Improved adaptive gaussian mixture model for background subtraction," in *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, vol. 2. IEEE, 2004, pp. 28–31.

[9] Z. Zivkovic and F. Van Der Heijden, "Efficient adaptive density estimation per image pixel for the task of background subtraction," *Pattern recognition letters*, vol. 27, no. 7, pp. 773–780, 2006.

[10] C. Tomasi, "A simple camera model," 2016. [Online]. Available: https://courses.cs.duke.edu//fall16/compsci527/notes/camera-model.pdf

[11] "Survey over image thresholding techniques and quantitative performance evaluation," 2004. [Online]. Available: https://www.researchgate.net/publication/202972407_Survey_over_image_thresholding_techniques_and_quantitative_performance_evaluation

[12] "Image segmentation." [Online]. Available: https://en.wikipedia.org/wiki/Image_segmentation

[13] "Watershed theory explanation by the centre de morphologie mathématique in paris," 2010. [Online]. Available: https://people.cmm.minesparis.psl.eu/users/beucher/wtshed.html#princip

[14] "Mathematical morphology and road scenes analysis." [Online]. Available: https://people.cmm.minesparis.psl.eu/users/beucher/prometheus.html

[15] "Grabcut -interactive foreground extraction using iterated graph cuts," 2004. [Online]. Available: https://www.microsoft.com/en-us/research/wp-content/uploads/2004/08/siggraph04-grabcut.pdf

[16] "Siox: Simple interactive object extraction." [Online]. Available: https://imagej.net/plugins/siox

[17] "What's the difference between supervised, unsupervised, semi-supervised and reinforcement learning?" [Online]. Available: https://blogs.nvidia.com/blog/2018/08/02/supervised-unsupervised-learning/

[18] Y. Baştanlar and M. Özuysal, "Introduction to machine learning," *miRNomics: MicroRNA Biology and Computational Analysis*, pp. 105–128, 2014.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[20] "State-of-the-art survey on color segmentation methods." [Online]. Available: https://upcommons.upc.edu/bitstream/handle/2117/93501/04Jvl04de11.pdf?sequence=4&isAllowed=y

[21] "Procesamiento morfológico de imágenes en color. aplicación a la reconstrucción geodésica. chapter 3." [Online]. Available: https://rua.ua.es/dspace/bitstream/10045/10053/5/Ortiz-Zamora-Francisco-Gabriel_4.pdf

[22] "Morphological transformations, opencv tutorials." [Online]. Available: https://docs.opencv.org/3.4/d4/d76/tutorial_js_morphological_ops.html

[23] "Translation-symmetry." [Online]. Available: https://en.wikipedia.org/wiki/Translational_symmetry

[24] "Types of morphological operations, matlab." [Online]. Available: https://www.mathworks.com/help/images/morphological-dilation-and-erosion.html

[25] "Image processing learning resources, hipr2." [Online]. Available: https://homepages.inf.ed.ac.uk/rbf/HIPR2/morops.htm

[26] "Morphological transformations, opencv." [Online]. Available: https://docs.opencv.org/4.5.2/d9/d61/tutorial_py_morphological_ops.html

[27] B. Stroustrup, *The C++ programming language*, 3rd ed., 1998. [Online]. Available: http://www.staroceans.org/e-book/The_C__Programming_Language__Stroustrup_.pdf

[28] G. B. García, *Learning Image Processing with OpenCV*, 1st ed., 2015.

[29] A. K. Gary Bradski, *Learning OpenCV: Computer Vision with the OpenCV Library*, 2008.

[30] "Microsoft visual studio." [Online]. Available: https://en.wikipedia.org/wiki/Microsoft_Visual_Studio

[31] "Qt designer manual." [Online]. Available: https://doc.qt.io/qt-5/qtdesigner-manual.html

[32] "Virtualdub2." [Online]. Available: http://virtualdub2.com/

[33] "Gimp." [Online]. Available: https://www.gimp.org/

[34] "About latex." [Online]. Available: https://www.latex-project.org/about/

[35] "Functional requirements." [Online]. Available: https://en.wikipedia.org/wiki/Functional_requirement

[36] "Non-functional requirements." [Online]. Available: https://en.wikipedia.org/wiki/Non-functional_requirement