UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

VIDEO OBJECT TRACKING USING FOREGROUND MODELS

A Master's Thesis submitted to

the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Stefano Maccarana

Advisor: Montse Pardàs

Abstract

This Master Thesis present an approach to Video Object Tracking segmentation using foreground models. For the video sequences analysed, the foreground and the background have been modelled using Spatial Colour Gaussian Mixture Models (SCGMMs).

SCGMMs are Gaussian Models which describes the foreground and the background using five components in colour and spatial domains.

In order to have a better result in the segmentation process, the Gaussian Models computed for each frame are passed to the next frame using a tacking technique that helps in the individuation of the object in foreground alone the sequence.

Using the location provided by the tracking, the Gaussian Mixture Model for the background is computed only in the close region around the object in foreground allowing in this way a better modelling of the region.

The Thesis is structure as follows: after a presentation of the study of the State of the Art where the techniques for tracking and segmentation are presented, there is the presentation of the method proposed. At the end there is a Chapter that describes the results obtained and some conclusions and a Chapter which presents some future developments.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my advisor Professor Montse Pardàs for the continuous support of research, for her motivation, enthusiasm and patience.

Her guidance helped me in all the time of research and writing of this Thesis. I could not have imagined having a better advisor for my Thesis.

Besides my advisor, I would like to thank all the UPC staff for allowing me to be here in Barcelona and for giving me all the tools necessary to develop my Thesis

I would also thank Fatima and my beloved family for supporting me daily and for encouraging me to face new challenging experience every day.

Table of contents

List of Figures

List of Tables

1 INTRODUCTION

1.1 SEGMENTATION INTRODUCTION

Image segmentation, according to the definition given by Haralick and Shapiro [1] is the partition of an image into a set of non-overlapping regions whose union is the entire image. The purpose of segmentation is to decompose the image into parts that are meaningful with respect to a particular application.

Video segmentation has been defined [2] as the problem of partitioning a video sequence into coherent regions with regard to motion and appearance properties.

The main application of these techniques is to identify objects or other relevant information in digital images and video sequences.

One of the classical field of application for image segmentation is the bio-medical one, where, for example, it is necessary detect tissues and bones in medical images.

Concerning the segmentation for video applications, as analysed in this Thesis, the principal employments for the technique are, for instance, video surveillance high level computer vision applications like human behaviour analysis or video sequence indexation, where a specific segmentation of the object, previously determined by the user, is needed.

Another example of video segmentation is the video conferencing application, where it is necessary to extract the foreground, the speaker, and replace the background with another image, like in the weather forecasting news.

1.2 MASTER THESIS STRUCTURE

The current Master Thesis is structured as follows.

In Chapter 2 the State of the Art is presented. We describe the techniques studied before to start the project developed for this Thesis.

In particular, GrabCut algorithm, used as initial base for the development of the foreground and background segmentation, MEANSHIFT algorithm used to develop the tracking part of the system and three different approaches that make use of Spatial Colour Gaussian Mixture Models to segment video sequences.

In Chapter 3 the method proposed for the project is described. Here we also describe the developing environment used to work on this Thesis.

In Chapter 4 the results produced from the proposed algorithm are presented.

We also provide in this Chapter a comparison between the results provided by the method proposed and results provided by other approaches.

In Chapter 5 and Chapter 6 we present the conclusions about the results provided and specifically in Chapter 6 we propose some future developments which can be done on the proposed method.

2 STATE OF THE ART

In image object tracking the common difficulties that a system must handle are problems like changes of the object patterns or the scene, non-rigid object structures, occlusions and camera motion.

In this framework, the object trajectory is calculated by an estimation of the object location in each video frame. Such an estimation has an essential task in environment like video editing, post-processing and interactive applications where the shape of the object should be considered.

Objects are usually represented by a geometrical shape like an ellipse or a rectangular; nevertheless, such a fixed shape is a too simple representation of real objects. This could be a problem for applications, like gesture recognition, using object's shape in order to extract information about the scene since they cannot make use of these trackers and they require video object segmentation.

Several techniques, such as the ones described in the following pages, solve that problem.

The techniques presented are methods adopted for tracking and segmentation, and have been used as support for my Thesis

A fundamental distinction between tracking, which can be defined as the problem of estimating the trajectory of an object in the image plane as it moves around a scene [3], and segmentation techniques is that tracking systems are usually designed for real-time purposes, whereas segmentation algorithm may work off-line in order to have accurate results.

The methods presented are:

- GrabCut - Interactive Foreground Extraction using Iterated Graph Cuts;
- CAMSHIFT;
- Monocular Video Foreground/Background Segmentation by Tracking Spatial-Color Gaussian Mixture Models;
- Foreground Object Segmentation for Moving Camera Scenarios Based on SCGMM;
- Video SnapCut: robust Video Object Cutout Using Localized Classifiers.

2.1 GRABCUT-INTERACTIVE FOREGROUND EXTRACTION USING ITERATED GRAPH CUTS

GrabCut [4] is an extension of the graph-cut algorithm [5] [6], and with this purpose it enhances the previous algorithm using new features.

2.1.1 Image segmentation by graph cut

Graph cut algorithm [5] is devoted to the segmentation of monochrome image given an initial trimap T. The trimap can be described as the partitioning of an image in three regions, a foreground region, a background region and an unknown region.

The image is described as an array $\boldsymbol{z} = (z_1, \cdots, z_N)$ of grey values, and the segmentation of this image is expressed as an array $\underline{\alpha} = (\alpha_1, \cdots, \alpha_N)$ of opacity values, where $\alpha_n \in \{0,1\}$, with 0 for background and 1 for foreground.

The grey-level distributions for foreground and background are described by the parameters $\underline{\theta}$ which consist of histograms of grey values:

$$\underline{\theta} = \{h(z;\alpha), \alpha = 0,1\} \quad (1)$$

The segmentation process goal is to deduce the unknown values $\underline{\alpha}$ from the given image $\boldsymbol{z}$ and the model $\underline{\theta}$. This task is performed by energy minimization.

An energy function $\boldsymbol{E}$ is defined:

$$\boldsymbol{E}(\underline{\alpha}, \underline{\theta}, \boldsymbol{z}) = U(\underline{\alpha}, \underline{\theta}, \boldsymbol{z}) + V(\underline{\alpha}, \boldsymbol{z}) \quad (2)$$

Where:

- $U(\underline{\alpha}, \underline{\theta}, \boldsymbol{z})$ evaluates the fit of the opacity distribution $\underline{\alpha}$ to the data $\boldsymbol{z}$;
- $V(\underline{\alpha}, \boldsymbol{z})$ is the smoothness term.

Once the energy model is fully defined, with the terms above, the segmentation can be estimated as a global minimum:

$$\underline{\hat{\alpha}} = \arg\min_{\underline{\alpha}} \boldsymbol{E}(\underline{\alpha}, \underline{\theta}) \quad (3)$$

This minimization is done using a minimum cut algorithm like the one described in [6].

2.1.2 GrabCut

The first enhancement of GrabCut is to allows an incomplete trimap, another improvement is to use coloured images instead of grey-scale images replacing the monochrome image model by a Gaussian Mixture Model. Additionally, it replaces the min cut estimation algorithm with a powerful iterative procedure that alternates between estimation and parameter learning and it allows also an incomplete labelling to relax the demands on the interactive user.

2.1.3 Incomplete Trimap

Incomplete labelling is useful when the user, instead of defining the full complete trimap T, specify only the foreground region T_F by drawing a rectangle around the subject that is the foreground. GrabCut uses the pixels inside of it as provisional T_F to generate the foreground GMM. These pixels can change to belong to the background after some iterations of the GrabCut algorithm, whereas those pixels outside of the bounding box are assigned to T_B at the beginning and never change label. This is related only to a single frame, because, obviously, foreground and background regions can change along the video sequence.

$$T = T_F \cup T_B \cup T_U \quad (4)$$

Where T_F is the foreground, T_B the background, and T_U the unknown.

2.1.4 Colour data modelling

In GrabCut, the image is taken as a vector $z = (z_1, \cdots, z_n, \cdots, z_N)$ of pixel in the RGB colour space. In order to do this, two GMMs[1] are used, one for the foreground and another one for the background. This introduces an additional vector $k = (k_1, \cdots, k_n, \cdots, k_N)$ in the optimization framework, assigning to each pixel a unique GMM component, either from the foreground or the background. The segmented image is represented as an array $\alpha = (\alpha_1, \cdots, \alpha_n, \cdots, \alpha_N)$ which gives to every pixel a value that can be 0 or 1, where 0 means background, and 1 means foreground.

With these changes, the energy function for segmentation becomes:

$$\boldsymbol{E}(\underline{\alpha}, \boldsymbol{k}, \underline{\theta}, \boldsymbol{z}) = U(\underline{\alpha}, \boldsymbol{k}, \underline{\theta}, \boldsymbol{z}) + V(\underline{\alpha}, \boldsymbol{z}) \quad (5)$$

U evaluates the fit of the opacity distribution $\underline{\alpha}$ to the data $\boldsymbol{z}$ given the histogram model $\underline{\theta}$, and is defined, taking into account the colour GMM models, as:

$$U(\underline{\alpha}, \boldsymbol{k}, \underline{\theta}, \boldsymbol{z}) = \sum_n D(\alpha_n, k_n, \underline{\theta}, z_n) \quad (6)$$

Where:

$$D(\alpha_n, k_n, \underline{\theta}, z_n) = -\log p(z_n | \alpha_n, k_n, \underline{\theta}) - \log \pi(\alpha_n, k_n) \quad (7)$$

With:

- $p(\cdot)$ that is a Gaussian probability distribution,
- $\pi(\cdot)$ are mixture weighting coefficients.

The parameters of the model are:

$$\underline{\theta} = \{\pi(\alpha, k), \mu(\alpha, k), \Sigma(\alpha, k), \alpha = 0,1, k = 1 \cdots K\} \quad (8)$$

[1] GMMs, Gaussian Mixture Models.

Where:

- π are the weights,
- μ the means
- Σ the covariances of the Gaussian components for the foreground and the background distributions.

$V(\underline{\alpha}, \boldsymbol{z})$ is the smoothness term and is computed using the Euclidean distance in the RGB colour space.

2.1.5 Iterative energy minimization

The energy minimization works iteratively, allowing an automatic refinement of the opacities $\underline{\alpha}$, which define the pixels belonging to foreground or to background, as newly labelled pixels from the T_U region of the initial trimap are used to refine the colour GMM parameters $\underline{\theta}$.

The energy minimization workflow of is shown below:

1. Assign GMM components to pixel: for each *n* in T_U

$$k_n := arg \min_{k_n} D_n(\alpha_n, k_n, \theta, z_n) \quad (9)$$

2. Learn GMM parameters from data *z*:

$$\underline{\theta} := arg \min_{\underline{\theta}} U(\underline{\alpha}, \boldsymbol{k}, \underline{\theta}, \boldsymbol{z}) \quad (10)$$

3. Estimate segmentation: use min cut to solve:

$$\min_{\{\alpha_n : n \in T_U\}} \min_{\boldsymbol{k}} \boldsymbol{E}(\underline{\alpha}, \boldsymbol{k}, \underline{\theta}, \boldsymbol{z}) \quad (11)$$

4. Repeat from step 1, until convergence.
5. Apply border matting

2.2 CAMSHIFT

CAMSHIFT [7] is a tracking algorithm proposed by Gary R. Bradski in order to improve the Mean Shift algorithm [8] which was never intended to be used as a tracking algorithm, but it is quite effective in this role.

The mean shift algorithm operates on probability distributions. To track coloured objects in video frame sequences, the colour image data has to be represented as a probability distribution; hence CAMSHIFT algorithm uses colour histograms to accomplish this.

Colour distributions extracted from video sequences change over time, so the Mean Shift algorithm has been modified to adapt dynamically to the probability distribution it is tracking.

Dynamically changing distributions occur when objects in video sequences are being tracked and the object moves so that the size and location of the probability distribution changes in time.

The CAMSHIFT algorithm adjusts the search window size during its operation, relying on the zeroth moment information.

2.2.1 Mean shift

To understand how the CAMSHIFT algorithm works, it is necessary to explain how the mean shift algorithm does. The mean shift algorithm is a non-parametric technique that climbs the gradient of a probability distribution to find the nearest dominant mode (peak).

For an image, a searching window is chosen; then the mean location is computed and the searching window is centred at this mean location iteratively until convergence.

For discrete 2D image probability distributions, the mean location (the centroid) within the search window is found as follows.

Compute the zeroth moment:

$$M_{00} = \sum_x \sum_y I(x,y) \quad (12)$$

Compute the first moment for x and y:

$$M_{10} = \sum_x \sum_y xI(x,y) \; ; \; M_{01} = \sum_x \sum_y yI(x,y) \quad (13)$$

Where $I(x,y)$ is the pixel probability value at coordinates (x,y).

Then the mean search window location, the centroid, is computed in its coordinates $x_c$and y_c,

$$x_c = \frac{M_{10}}{M_{00}} \; ; \; y_c = \frac{M_{01}}{M_{00}} \quad (14)$$

The zeroth moment, described above, can be seen as the distribution “area” found under the search window.

2.2.2 CAMSHIFT for video sequence

The CAMSHIFT algorithm is computed following these steps:

1. Choose the initial location of the search window;
2. Computes the Mean Shift as described above and stores the zeroth moment M_{00}.
3. Sets the search window size equal to a function of the zeroth moment found in Step 2.
4. Repeat Steps 2 and 3 until convergence.

The steps above are repeated for each frame starting from the mean location, that is the centre of the window, computed for the previous frame.

In addition to this, CAMSHIFT can also compute the orientation of the object in each frame.

To compute the orientation, the second moment for x and y is required:

$$M_{20} = \sum_x \sum_y x^2 I(x,y) \; ; \; M_{02} = \sum_x \sum_y y^2 I(x,y) \qquad (15)$$

Where, as described before, $I(x,y)$ is the pixel probability value at coordinates (x,y).

Hence, the orientation θ will be computed as:

$$\theta = \frac{arctan\left(\frac{2\left(\frac{M_{11}}{M_{00}} - x_c y_c\right)}{\left(\frac{M_{20}}{M_{00}} - {x_c}^2\right) - \left(\frac{M_{02}}{M_{00}} - {y_c}^2\right)}\right)}{2} \qquad (16)$$

2.3 MONOCULAR VIDEO FOREGROUND/BACKGROUND SEGMENTATION BY TRACKING SPATIAL-COLOR GAUSSIAN MIXTURE MODELS

In this paper [9] has been studied the foreground and background segmentation problem for monocular videos. Particularly the authors were interested in video sequence recorded by static or hand-cameras filming large moving foreground objects.

They consider, for instance, that being a hand-held camera, it can be shaking producing other moving object in the background. As it is written in this paper, this kind of sequence are difficult to analyse and the main challenges are:

- The fact that there are other moving background objects may cause confusions to the segmentation if they are not correctly modelled;
- the objects in foreground can have relatively large dimensions, so superposition between them and background objects may occur frequently;
- the object in foreground can have complex and rapid movements;
- foreground and background objects may have portions with similar colour patterns;
- foreground and background objects have different motion patterns.

The approach used in this method is to model both the foreground and the background using SCGMM models. These models are built using a MRF[2] energy function that can be minimized using the graph cut algorithm.

Considering sequences with rapid and complex foreground and background motions, the SCGMM learned from previous frames cannot be used to segment the current frame, because of large variations in the spatial domain.

Hence, an assumption they made is that the motions between subsequent frames are small, so the mixture models in the previous frame can be directly applied to the segmentation of the current frame.

[2]MRF = Markov Random Field

2.3.1 Energy Minimization Formulation

With this approach they propose to solve the background-foreground segmentation using energy minimization.

Calling $\boldsymbol{z}_i$ the feature vector extracted from the pixel i, there is an unknown binary label f_i which can be 0 or 1 depending if the pixel belongs to the background or the foreground.

Over the unknown labelling variables of every pixel is defined an energy-based objective function in the form of a fist order Markov random field energy function:

$$E(f) = E_{data}(f) + \lambda E_{smooth}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) + \lambda \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) \quad (17)$$

Where:

- $\mathcal{N}$ is the set of 8 connected neighbouring pixels;
- $\mathcal{P}$ is the set of the pixels of the frame.

The energy function described above can be minimized by a graph cut algorithm [6].

The smoothness term $E_{smooth}(f)$ is described as:

$$E_{smooth}(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) = \sum_{\{p,q\} \in \mathcal{N}} \frac{1}{d(p,q)} e^{\frac{-(I_p - I_q)^2}{2\sigma^2}} \quad (18)$$

Where:

- I_p is the intensity of the pixel p;
- σ is the average intensity difference between neighbouring pixels
- $d(p,q)$ is the distance between pixel p and q.

The data energy term $E_{data}(f)$ evaluates the probability of each pixel to belong to the foreground or the background according to the models estimated.

As said above, $\boldsymbol{z}_i$ is the feature vector extracted from the pixel i, and in this case is:

$$\boldsymbol{z}_i = (x, y, r, g, b) \quad (19)$$

Where:

- (x, y) are the pixel coordinates;
- (r, g, b) are the pixel colour values.

Using a five dimensional feature vector, a five dimensional GMM is obtained and the probability of a pixel to belong to foreground or background is described as:

$$p(z|l) = \sum_{k=1} p_{l.k} G(\boldsymbol{z}; \mu_{l,k}, \boldsymbol{\Sigma}_{l,k}) \quad (20)$$

Where:

- l represent the foreground or the background;
- $p_{l.k}$ is the prior of the k^{th} Gaussian component in the mixture model;
- $\mu_{l,k}$ is the mean of each Gaussian component;
- $\boldsymbol{\Sigma}_{l,k}$ is the covariance matrix of each Gaussian component.

In Formula (20) $G(\boldsymbol{z}; \mu_{l,k}, \boldsymbol{\Sigma}_{l,k})$ is the k^{th} Gaussian component and can be described as:

$$G(\boldsymbol{z}; \mu_{l,k}, \boldsymbol{\Sigma}_{l,k}) = \frac{1}{(2\pi)^{5/2} \left|\Sigma_{l,k}\right|^{1/2}} e^{-\frac{1}{2}(z_i - \mu_{l,k})^T \Sigma_{l,k}{}^{-1} (z_i - \mu_{l,k})} \quad (21)$$

Where:

- $\mu_{l,k}$ is the mean of the k^{th} Gaussian distribution;
- $\Sigma_{l,k}$ is the covariance matrix of the k^{th} Gaussian distribution;
- l, as before, indicates if the pixel belongs to the foreground or to the background

The covariance matrix is considered in a block diagonal form, separating the spatial and colour features.

In this way each Gaussian component can be factorized as:

$$G(\boldsymbol{z}; \mu_{l,k}, \boldsymbol{\Sigma}_{l,k}) = G(\boldsymbol{z}_s; \mu_{l,k}, \boldsymbol{\Sigma}_{l,k}) G(\boldsymbol{z}_c; \mu_{l,k}, \boldsymbol{\Sigma}_{l,k}) \quad (22)$$

Where the subscripts c and s stand for colour and spatial.

The data cost $E_{data}(f)$ is described as:

$$E_{data}(f) = \sum_{p \in \mathcal{P}} D_p(f_p) = \sum_{p \in \mathcal{P}} -\log p(\boldsymbol{z}_p | f_p) \quad (23)$$

Where $p(\boldsymbol{z}_p | f_p)$ is computed using Formula (20).

2.3.2 SCGMM Joint Tracking and Segmentation

The problem considered with this approach is how to propagate the two SCGMM computed for foreground and background along the entire sequence, since the both the background and foreground objects can continuously move.

They assume that for two subsequent frames the colour of the foreground and background object does not change, so the colour model part of the SCGMM remains identical for both frames. Instead the spatial part of the SCGMM has to be updated.

In order to compute the updated model for the new frame, they combine the foreground and background into a global SCGMM for the whole image.

In this stage there is an update using an EM algorithm[3] of the spatial coordinates of the whole models, and the colour components remain the same of the previous frame.

After the updating, the SCGMM joint model is split back, as originally, into two distinct models for the current frame, one for the foreground and one for the

[3] EM algorithm = Expectation – Maximization algorithm

background. The two SCGMM models are then used to perform graph cut segmentation.

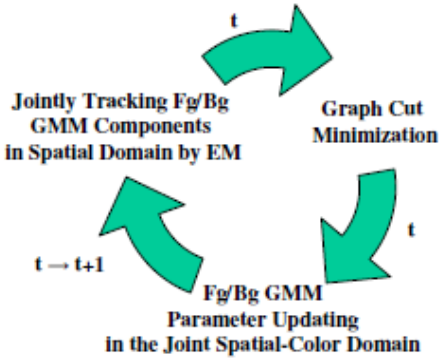

Figure 1 Iterative circle of foreground and background segmentation [9]

The scheme above summarize the method used by [9].

2.4 FOREGROUND OBJECT SEGMENTATION FOR MOVING CAMERA SCENARIOS BASED ON SCGMM

In this paper [10] the approach is to use a region-based parametric probabilistic model, the Spatial Colour Gaussian Mixture Model (SCGMM) to model the foreground object to segment and also the close background regions around the object. Doing this, there is a robust classification of the pixels that could belong to the foreground or to the background.

2.4.1 Foreground and Background Model

To model the foreground and background areas, two parametric models are used, one per each zone.

Each model is a Spatial Colour Gaussian Mixture Model, so, as in other approaches, the pixel is described with a five dimensional vector $z_i, i \in \mathbb{R}^5$.

The likelihood $p(z_i|l)$ of a pixel to belonging to the foreground or the background is described as in Section 2.3.1 for [9].

Again, as in [9], the colour components and the spatial components are decoupled, so can be factorized as:

$$G_l(z_i, \mu_k, \Sigma_k) = G(x_i, \mu_{k,s}, \Sigma_{k,s})G(v_i, \mu_{k,c}, \Sigma_{k,c}) \quad (24)$$

Where:

- x_i is the pixel spatial information;
- v_i is the pixel colour information.

The parameter estimation is done with an EM algorithm.

2.4.2 Initialization, updating and classification

The initialization is done for each model, foreground and background, defining a bounding box. Then there is a computation of the RGB[4] histogram for both the models and the calculation of the correct number of Gaussians needed to represent the foreground model and the close background regions.

After this, take place a fast two-steps initialization process: on the first step the algorithm places the Gaussian distribution of the two models over the spatial region belongings to each model, foreground or background.

Then, on the second step, for each class an EM algorithm is used to computes the parameter of each GMM, obtaining iteratively a maximization of the likelihood.

With regard to the updating there is the assumption that the complete update of the classes (fg or bg) in both the domains, colour and spatial, could lead to propagation error, so there is a two-step updating for each model; one step is for the spatial domain and the other is done for the colour domain.

- Spatial domain updating: only the spatial components of the GMM are updated.
  Each pixel is assigned to a Gaussian k according to:

$$P(k|z_i, l) = \frac{w_k G_l(z_i, \mu_k, \sigma_k)}{\sum_k w_k G_l(z_i, \mu_k, \sigma_k)} \quad (25)$$

  Once the pixel has been assigned to a Gaussian, takes place a calculation of the mean and covariance matrix of each Gaussian.

- Colour domain updating: after the spatial domain update, it takes place the update for the foreground model in the colour domain.

[4] RGB = Red, Green, Blue.

As for the classification, the segmentation determination is taken using a *Maximum A Posteriori* decision. This is achieved considering an energy function that is solved using a standard graph cut algorithm [6].

One the classification is complete for the frame $t-1$, the labelling can be done for the frame t.

2.4.3 Comparison with [9]

Since this paper [10] is based on the one [9] described in Section 2.3, it is opportune to make a comparison between the two approaches.

Both the one from Yu [9] and the one from Gallego [10] make use of the Spatial Colour Gaussian Mixture Model as parametric probabilistic model to describe the foreground object and the background.

A difference between the two approaches regarding the SCGMMs is that for Gallego [10], the modelling for the background region refers only to the close regions that appear surrounding the object in foreground instead of modelling all the background in the frame.

Another important difference the two methods in how they approach the updating of the models. Yu [9], after the computation of the two SCGMMs for foreground and background, creates a joint model where he updates only the spatial components of the joint model, and then separate it into the two foreground and background models again.

Gallego [10], instead, for each frame of the sequences uses a three steps process: classification of each pixel inside the bounding box that includes the foreground and the close background according to the foreground and background defined from the previous frame; updating of each model first in spatial and then in colour domains; redefinition of the ROI (bounding box) according to the resultant foreground segmentation.

2.5 VIDEO SNAP CUT: ROBUST VIDEO OBJECT CUTOUT USING LOCALIZED CLASSIFIERS

The goal of this paper [11] is to develop a video cutout system which can work on complicated video sequences.

They propose a video segmentation model with overlapping localized classifier, which are a group of overlapping window that are localized around the contour of the object in foreground.

2.5.1 Local classifiers

Each one of these windows is associated with a local classifier that segments a local portion of the foreground contour.

Each of these classifiers describes local feature of the frame such as shape, colour and motion.

Given a frame I^t, the user provides an accurate mask $L^t(x)$ for the object to be segmented. A set of overlapping windows $W_1^t, \cdots, W_n^t$ along the contour C^t is then created.

Each classifier inside a window W_k^t has:

- a local colour model M_c;
- a colour model confidence f_c;
- a local shape model M_s.

Two GMMs, one for the foreground $(\mathcal{F})$ and one for the background $(\mathcal{B})$, are then created considering that for a pixel x:

$$p_c(x) = \frac{p(x|\mathcal{F})}{p(x|\mathcal{F}) + p(x|\mathcal{B})} \quad (26)$$

Where $p(x|\mathcal{F})$ and $p(x|\mathcal{B})$ are the probabilities computed from the two Gaussian Mixture Models.

Then the local colour model confidence f_c is used to describe how much the foreground is separable from the background in the local region.

The model M_s which describes the local shape, contains the user provided segmentation mask $L^t(x)$, and a shape confidence f_s is computed.

2.5.2 Frame propagation and updating

Once the classifiers are initialized, they are propagated on the next frame employing a motion estimation technique.

This technique is based on a computation of the SIFT[5] feature points between the two consecutive frames and then the optical flow between these two sets of points is calculated.

The motion vectors obtained from the above method are only used to shift the local windows in order to cover the contour of the moving object.

After the computation of the propagation of the local features, an update of them is required. Hence the shape model M_s is updated warping the initial foreground mask $L^t(x)$ accordingly to the motion vectors computed before.

To update the colour models two new GMM are built, one for the foreground $G_{\mathcal{F}}^{t+1}$ and the other for the background $G_{\mathcal{B}}^{t+1}$, sampling the pixels in the new windows. Now there are two possible sets of colour, one from the previous frame, called history model $M_c^h = \{G_{\mathcal{F}}^t, G_{\mathcal{B}}^t\}$, and one updated $M_c^h = \{G_{\mathcal{F}}^t, G_{\mathcal{F}}^{t+1}, G_{\mathcal{B}}^t, G_{\mathcal{B}}^{t+1}\}$.

At this point a decision is made to select which one of the two models has to be used for segmenting the current image window.

[5] SIFT = Scale-invariant feature transform

The local classifier adaptively integrates the new updated models M_s and M_c in order to produce the segmentation, following the rule that when the sets of colour from the foreground and the background are separable the colour model M_c should be used, otherwise M_s should have a bigger importance.

The foreground probability in the window W_k in the current frame becomes:

$$p_{\mathcal{F}}^{k}(x) = f_s(x)L^{t+1}(x) + \frac{1 - f_s(x)}{p_c(x)} \quad (27)$$

The probability above is referred to a single window W_k. Considering instead the whole frame, the combination of the foreground probabilities becomes:

$$p_{\mathcal{F}}(x) = \frac{\sum_k p_{\mathcal{F}}^{k}(x)(|x - c_k| + \varepsilon)^{-1}}{\sum_k (|x - c_k| + \varepsilon)^{-1}} \quad (28)$$

For large motions, the process can be iterated in order to have a more precise segmentation.

3 THEORETICAL DESCRIPTION OF THE METHOD

In this Chapter it is presented a theoretical description of the method, the explanation of how the system works.

The system is essentially composed by two sections: Tracking and Segmentation, that are sequentially applied on the input video sequence.

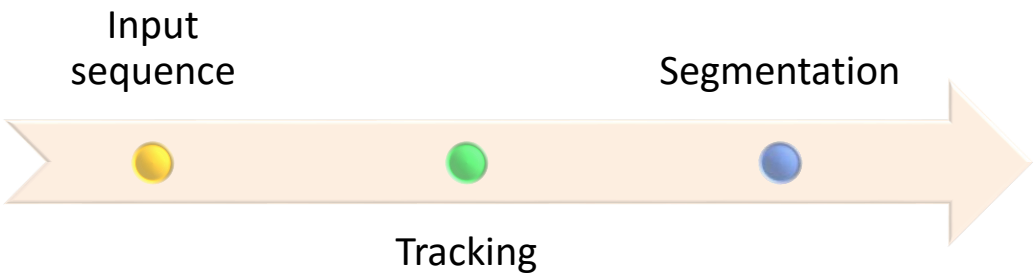

Figure 2 Flow chart of the system

The input sequence is split into single frames and each frame is treated with tracking and segmentation.

At this point it is necessary to distinguish between two modes of work, which will be described in the next Sections:

- Work Mode: the user selects a bounding box, then GrabCut is used for initialization on the first frame;
- Test Mode: in order to obtain quantitative results and evaluate exclusively the tracking and segmentation developed, the initialization on the first frame is done using a ground truth mask.

3.1 TRACKING

Tracking part is a fundamental part for this project since the location provided by the tracking is used to initialize the segmentation tool.

Tracking is based on a CAMSHIFT [7] algorithm and provides a rectangular bounding box which then will be used for the next step, the segmentation.

Tracking method is developed in different parts described below.

3.1.1 Initialization

As said in the previous page, there are two working modes, Work Mode and Test Mode.

Since the Camshift algorithm is based on histogram tracking, to better compute this histogram can be applied two solutions:

- for Work Mode: on the first frame the user provides a rectangular selection around the object that he wants to be tracked, and then segmented, along the entire sequence. A segmentation with the original GrabCut algorithm is then performed using the rectangular selection provided by the user;
- for Test Mode: a masking using the ground truth mask is performed on the first frame in order to select only the object and to have the most precise possible computation of the histogram.

In this way only the actual subject is taken into account and the small portion of background around the object has been disregard in the computation of the histogram.

3.1.2 Conversion to HSV

The result, foreground area, of the GrabCut algorithm or of the masking using the ground truth depending on the method adopted is than used by the tracking algorithm to compute a histogram.

It is important to remark that, before the computation of the histogram, a conversion of the colour model is performed, in fact the frame is converted from RGB to HSV model. HSV is the acronym of Hue, Saturation, Value which are described below. The conversion is done because the RGB components of an object's colour in an image are all correlated with the amount of light hitting the object, and for that reason with each other, image descriptions in terms of those components make object discrimination difficult. Instead, descriptions in terms of Hue, Saturation and Value are often more relevant [12]. The hue, H, of a colour indicates which pure colour it describes. Hue values are expressed as a number that indicates the position of the corresponding pure colour on the colour wheel (look at Figure 3), as a fraction between 0 and 1.

Value 0 refers to red; $1/6$ is yellow; $1/3$ is green, $1/2$ is cyan, $2/3$ is blue and $5/6$ is magenta.

The saturation, S, of a colour defines how white the colour is. A pure white has a saturation value of 0, a pure red, that is fully saturated, has a saturation value of 1.

The value, V, of a colour indicates how much lightness the colour has. Black, that is the darker colour has a value of 0, and this value increase moving away from black toward the white.

To better understand the HSV model it is useful to take a look at the following diagram.

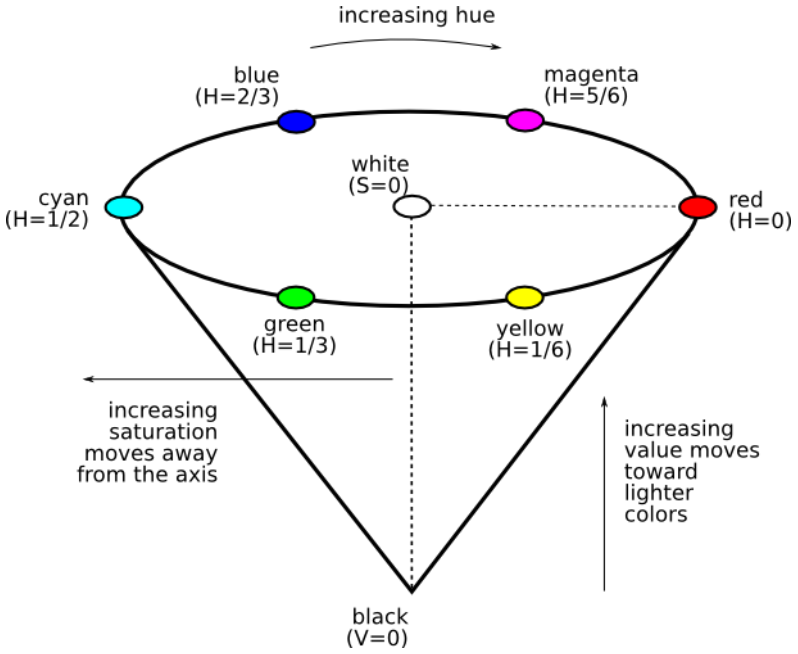

Figure 3 HSV model diagram [13]

The conversion from RGB to HSV is performed using the algorithm below.

$$V \leftarrow \max(R, G, B) \tag{29}$$

$$S \leftarrow \begin{cases} \dfrac{V - \min(R, G, B)}{V}, & V \neq 0 \\ 0, & otherwise \end{cases} \tag{30}$$

$$H \leftarrow \begin{cases} \dfrac{60(G - B)}{V - \min(R, G, B)}, & if\ V = R \\ 120 + \dfrac{60(B - R)}{V - \min(R, G, B)}, & if\ V = G \\ 240 + \dfrac{60(B - R)}{V - \min(R, G, B)}, & if\ V = B \end{cases} \tag{31}$$

If $H < 0\ then\ H \leftarrow H + 360$

3.1.3 Histogram

After the conversion, as said before, the algorithm computes the histogram of the object taking into account only the hue channel.

Histogram is a representation of the intensity distribution of an image, in this case the intensity distribution of the hue channel. Each bin, element of the histogram, represents a hue intensity and the value contained in each element corresponds to the number of pixels which have that hue intensity.

This histogram is stored and taken as object model for the entire video sequence.

On the next frames, also converted to HSV, the probability density function of the colour model is estimated.

3.1.4 Back projection

Back projection is a method used for computing how well the pixels in a given image fit the distribution of pixels in a histogram model.

After the conversion to HSV, only the Hue channel is taken into account and, as said before, the histogram of this channel is computed and used as histogram model.

The function gathers the value for each pixel from the H channel in the input images and finds the corresponding histogram bin. Then reads the bin value, scales and stores it in the back projection image, in the same coordinates of the pixel. In terms of statistics, the function computes the probability of each element value with respect to the empirical probability distribution represented by the histogram.

3.1.5 Location computation

The computation of the location of the centre of the object is performed by using the mean shift algorithm.

The function implements the iterative object search algorithm. It takes as input the back projection, computed before, and the initial position of an object.

The mass centre in window of the back projection image is computed and the search window centre shifts to the mass centre. The procedure is repeated for a number of iterations or until the window centre shifts by less than a small epsilon.

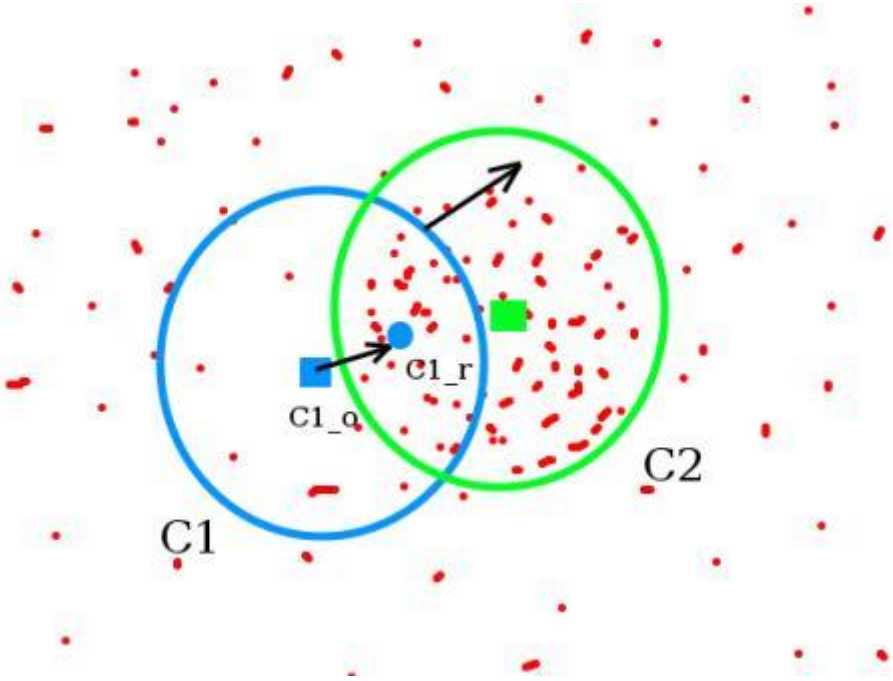

Figure 4 Mean shift algorithm [14]

In the figure above we show how the mean shift algorithm works. The blue circle, named *C1*, and its original centre is marked as the blue rectangle, named *C1_o*. Computing the centroid of the points inside that window, we will obtain a new point, depicted as a small blue circle called *C1_r* which is the real centroid of window. As we can easily see they don't match. So we move the window in order to have the new centroid as its centre.

Again a new centroid is computed and, as before, probably, it won't match. So we move the window again, and we continue the iterations such that centre of window and its centroid fall on the same location. In this way we finally obtain a window with maximum pixel distribution. In the figure is named *C2* and it is marked with a green circle.

3.1.6 Output window

When the convergence is reached the CAMSHIFT algorithm updates the size of the window and also calculates the orientation of best fitting ellipse.

Then a bounding box is provided to the segmentation section, and the tracking algorithm skips to the next frame and starting from the coordinates of the mass centre re-computes the new location of the object.

Figure 5 Tracking

In Figure 5 above is possible to see an example of the result of the tracking part.

3.2 OUR APPROACH FOR THE SEGMENTATION

Segmentation method used for this project is based on an implementation of GrabCut [4] algorithm, which has been extended to perform tracking in video sequences following the main ideas of [9].

GrabCut, that is an extension of the graph-cut algorithm [6], was designed to segment colour images as explained in Section 2.1.

Since our approach is devoted to video sequences, the GrabCut algorithm is taken as a starting point, where some extensions have been added.

First of all, GrabCut uses a three dimensional model, where each dimension represents a channel of the image to segment. The three channel are Red, Green and Blue.

In our case we expand this model in five dimensions, we keep the three original dimensions and we add two additional dimension which are the spatial coordinates.

Thus each pixel can be described as a five-dimensional vector $\boldsymbol{z}$.

$$\boldsymbol{z} = (r, g, b, x, y) \qquad (32)$$

Hence, instead of Colour Gaussian Mixture Models, Spatial Colour Gaussian Mixture Models, SCGMMs, are used as proposed in [9].

In order to do this, in the code, at the original three channel image are added two additional channels containing the x and y dimensions.

So, instead of using an RGB image we use an $RGBxy$ one.

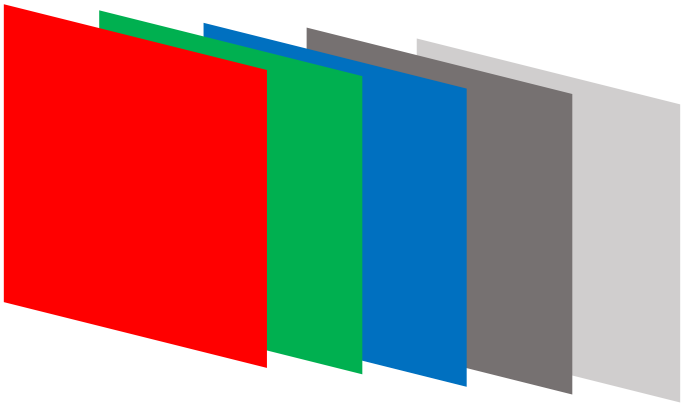

Figure 6 Image with 5 channels: RGBxy

The spatial information provided by the tracking algorithm and added to the models is very important since it will be used to compute the foreground and background models of the frame.

The original GrabCut algorithm was designed to operate for images and not for video sequences, hence an extension has been added.

Instead of compute the models of foreground and background for every frame thinking them as independent from the other frames, as GrabCut does for still images, we store the models computed for the frame t and we update it in the frame t+1, repeating this until the end of the sequence.

This allows us to produce better results and less errors in the segmentation of the entire sequence.

3.2.1 First Frame

We have to distinguish the procedure for the first frame and for the next ones, on the first frame we will pursue the following method.

Since the video sequence in split into frames, each frame can be considered as a still image. On every still image the segmentation algorithm is applied in order to produce a video segmentation.

On the first frame, *frame0*, we perform a segmentation using the basic GrabCut algorithm with the addition of the spatial channels. In order to have a good segmentation, we consider the first frame as an initialization one.

According to the working mode used the segmentation could be done with two different approaches:

- Work Mode: the segmentation of the first frame is done using GrabCut and the result is corrected by the user interaction in order to produce an accurate segmentation.
- Test Mode: the segmentation of the first frame is done using the ground truth mask which produce a correct classification of the pixel in two classes, foreground or background. In this case no user interaction is required.

This allows us to have very good models of the first frame and so we can store it in order to re-use it in the next frames.

Every k^{th} Gaussian component obtained for each region, foreground and background, can be described as:

$$G_l(z_i, w_k, \mu_k, \Sigma_k) \quad (33)$$

Where:

- $l = \{fg, bg\}$ indicates if the model approximates the foreground or the background;
- z_i is the feature vector of each pixel i;
- w_k is the weight of the k^{th} Gaussian;
- $\mu_k$is the mean of the k^{th} Gaussian;
- $\Sigma_k$is the covariance matrix of the k^{th} Gaussian.

Is important to say that actually the model is decoupled into two parts, one for the colour feature and the other for the spatial dimension, so the Formula (33) becomes:

$$G_l(z_i, w_k, \mu_k, \Sigma_k) = G_{ls}(z_{ic}, w_k, \mu_{kc}, \Sigma_{kc}) G_{ls}(z_{is}, w_k, \mu_{ks}, \Sigma_{ks}) \quad (34)$$

Where the subscripts c and s refer to the colour or the spatial domain. In particular, is useful to notice that the covariance matrix becomes:

$$\Sigma_k = \begin{bmatrix} \Sigma_{k_c} & 0 \\ 0 & \Sigma_{k_s} \end{bmatrix} \quad (35)$$

In order to have a batter description of the frame, the background models is computed only in the region close to the object.

This is possible thanks to the tracking algorithm that gives a location of the object tracked.

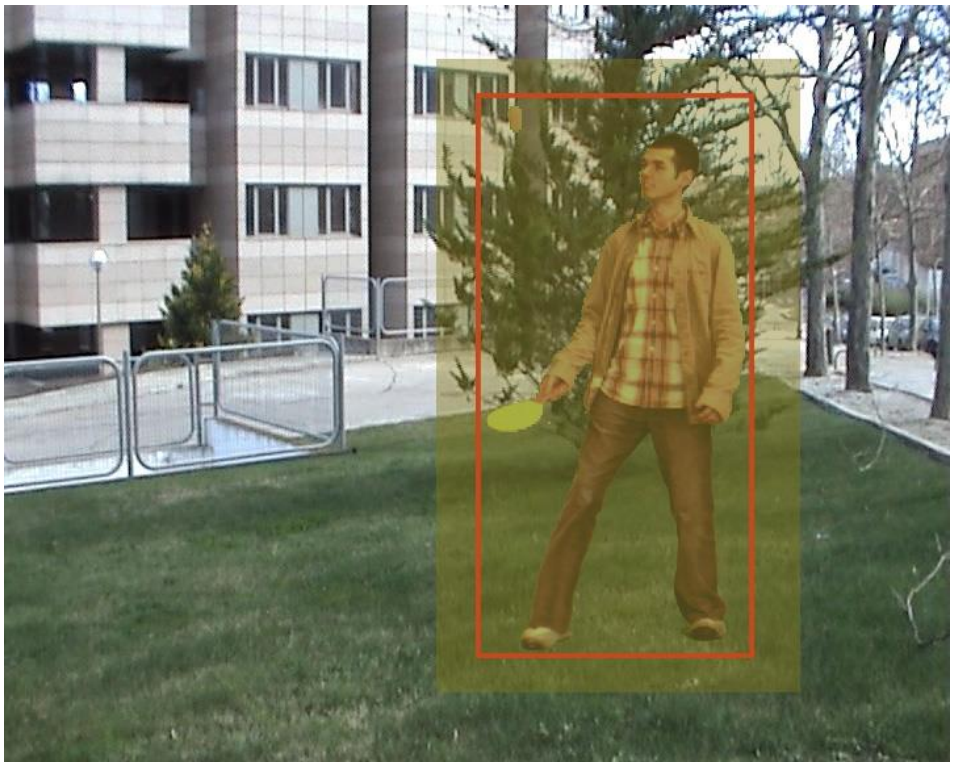

Figure 7 Example of the Close Background

As it can be seen in Figure 7, if the region provided the by the tracking algorithm is the one which is within the red rectangle, the area where the background model is computed is the yellow one.

The models of foreground and background are then saved separately using the structure described below.

Taking as example a model computed with k Gaussian the stored model will be the following:

$$(k \times weight\ elements) + (k \times mean\ elemens) + (k \times covariance\ matrices)$$

Where:

- *Weight elements* are the weight of each Gaussian,
- *Mean elements* are the means of each channel of each Gaussian, so in this case 5 values,
- *Covariance matrices* are $5x5$ matrices that contains the covariance of each Gaussian.

So, using for example $k = 30$ Gaussians, 930 values per each model will be obtained.

3.2.2 From the second frame until the end

From the next frame, *frame1*, until the end of the sequence, the algorithm is quite different.

After loading the previous saved models, a unique merged model for foreground and background is created.

So, instead of having two separate models with a k size, from now only one of double size $2k$ is present.

A very important observation at this step is that is necessary to scale the Gaussian weights of the two models in order to maintain $\sum_{2k} weight_k = 1$, hence a downscale by a factor of 2 per each weight is performed.

In addition to this, it is important to take into account how the models are structured, so it is fundamental to think about the structure described above.

The merged model will have a structure equal to the single models, but with a bigger dimension, the very first elements will be the ones from the foreground models and then the ones from the background model.

$$(2k \times weight\ elements) + (2k \times mean\ elemens) + (2k \times variance\ matrices)$$

Once the models have been merged, the assignation of each pixel to one of the $2k$ Gaussians is needed.

The assignation of each pixel of the frame to one of the $2k^{th}$ Gaussians of the GMM2k model is done using the likelihood function $D(m)$ for each pixel m to belong to a Gaussian. The pixel is assigned to the Gaussian for which it has the greater value of $D(m)$.

$$D(m) = -\log \sum_{i}^{2k} w_k \frac{1}{\sqrt{det\Sigma_i}} e^{\left(-\frac{1}{2}[z_m-\mu_i]^T \Sigma_i^{-1} [z_m-\mu_i]\right)} \quad (36)$$

Where:

- w_k is the weight of a k^{th} Gaussian;
- Σ_i is the covariance matrix of a k^{th} Gaussian and $det\Sigma_i$ is its determinant;
- z_m is the feature vector of a pixel m;
- μ_i is the mean of each component of a k^{th} Gaussian.

Since the model is decoupled into colour and spatial components, the likelihood function assumes this form:

$$D(m) \qquad (37)$$
$$= -\log \sum_{i}^{2k} w_k \left[\frac{1}{\sqrt{det\Sigma_{ic}}} e^{\left(-\frac{1}{2}[z_{mc}-\mu_{ic}]^T \Sigma_{ic}^{-1} [z_{mc}-\mu_{ic}] \right)} \frac{1}{\sqrt{det\Sigma_{is}}} e^{\left(-\frac{1}{2}[z_{ms}-\mu_{is}]^T \Sigma_{is}^{-1} [z_{ms}-\mu_{is}] \right)} \right]$$

Where the subscripts c and s indicate respectively the colour and spatial domains.

After the assignation of each pixel to one of the $2k^{th}$ Gaussian has taken place, is required a phase of learning of the Gaussian Mixture Model.

An important fact to notice is that here takes place the update of the models computed in the previous frame, and then merged into one, but only in the spatial domain.

Hence the learning of the GMM is performed updating only the spatial components and taking into account the previous values for the colour domain.

Thus for each of the k^{th} Gaussian, the covariance matrix Σ_{kc} and the means of the colour domain is copied from the previous frame and only Σ_{ks} and the means of the spatial domain is actually computed.

So:

$$\Sigma_k(t) = \begin{bmatrix} \Sigma_{k_c}(t-1) & 0 \\ 0 & \Sigma_{k_s}(t) \end{bmatrix} \qquad (38)$$

Where t is the actual frame, and $t-1$ is the previous one.

After the $2k$ model update, a separation between the two models, foreground and background, is performed.

At this step it is necessary to remind that, when the two models have been merged, the weight values have been downscaled by a factor of 2, hence now an upscale by a factor of 2 for these values is required.

After the creation of the two models, one for the foreground and the other for the background, similarly, as done for the merged model, each pixel m has to be assigned to one of the models, foreground or background, according to the probability:

$$D(m) = -\log \sum_{i}^{k} w_k \frac{1}{\sqrt{det\Sigma_i}} e^{\left(-\frac{1}{2}[z_m-\mu_i]^T \Sigma_i^{-1}[z_m-\mu_i]\right)} \quad (39)$$

$$= -\log \sum_{i}^{k} w_k \left[\frac{1}{\sqrt{det\Sigma_{ic}}} e^{\left(-\frac{1}{2}[z_{mc}-\mu_{ic}]^T \Sigma_{ic}^{-1}[z_{mc}-\mu_{ic}]\right)} \frac{1}{\sqrt{det\Sigma_{is}}} e^{\left(-\frac{1}{2}[z_{ms}-\mu_{is}]^T \Sigma_{is}^{-1}[z_{ms}-\mu_{is}]\right)}\right]$$

Where the subscripts c and s indicate respectively the colour and spatial domains.

The pixel m is assigned to the Gaussian, that can belongs to the foreground or background model, for which it has the greater value of $D(m)$.

Once each pixel has been assigned to one of the Gaussians and so to one of the models, foreground or background, the construction of the graph takes place.

According to [15] a graph is an ordered pair $G = (V, E)$ comprising a set V of vertices or nodes together with a set E of edges, which are 2-element subsets of V.

For every pixel m in the image there is a node and two special nodes, one for the foreground and the other for the background. These nodes are connected by edges, the links of the graph, which can be of two types:

- *N-links*, where *N* stays for neighbourhood, which connect each pair of neighbouring pixels;

- *T-links*, where *T* stays for terminal, which connect pixels to the terminal, foreground or background nodes.

Per each pixel m there are 8 neighbour n, as can be seen in Figure 8, and the $weightN$ describes the penalty of setting the boundary of the segmentation between each pair of neighbouring pixels.

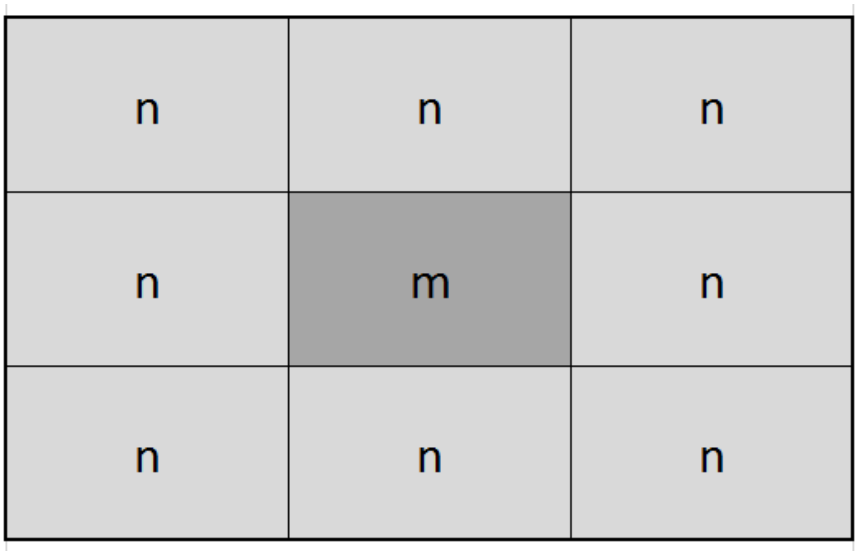

Figure 8 Pixel m and its 8 neighbours

Considering the pixel m and its neighbour n the edge weight will be computed as:

$$weightN(m,n) = \frac{\gamma}{dist(m,n)} e^{-\beta \|\boldsymbol{z}_m - \boldsymbol{z}_n\|^2} \quad (40)$$

Where:

$$\beta = \frac{1}{2\langle \|\boldsymbol{z}_m - \boldsymbol{z}_n\|^2 \rangle} \quad (41)$$

and γ is set equal to 50 accordingly to [16].

$weightT$, since it is the weight of the edge that connects the pixel to one of the two nodes, foreground or background, is instead computed using Formula (39) where the summation is over the foreground Gaussian or over the background Gaussian depending on which models the pixel has been assigned previously.

Looking at the Figure 9 is possible to understand how the construction of the graph works.

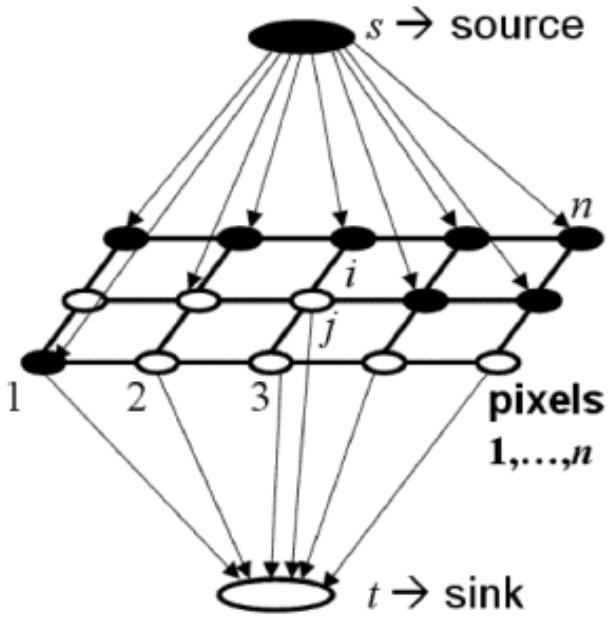

Figure 9 Graph [17]

Indicating with Sink and Source respectively the background and the foreground regions, a pixel is assigned to the foreground or the background performing a cut on the graph [6].

This cut will disconnect two types of links:

- *N-links*: a cut removes the links between pairs of pixels associated to different terminals.
- *T-links*: a cut removes one of the two edges that connect a pixel with a terminal Sink or Source node, jointing it this way to the foreground or to the background class.

This means that the cut must be interpreted not only links between neighbouring pixels (*N-links*) but as well as on a set of links between terminal nodes and pixels (*T-links*).

Once the cut is complete for the whole graph, a binary mask will be produced according to the remaining edges. If the pixel is connected to the node foreground, the value of the mask in its position will be 1, conversely if it is connected to the background node, the value of the mask in its position will be 0.

This binary mask will be used to evaluate the performances of the algorithm comparing it with the ground truth mask (see Section 3.3), and also to produce the output images, that are the foreground and the background of the frame.

Starting from the original frame:

Figure 10 Example of a frame

And applying the binary mask produced by the segmentation:

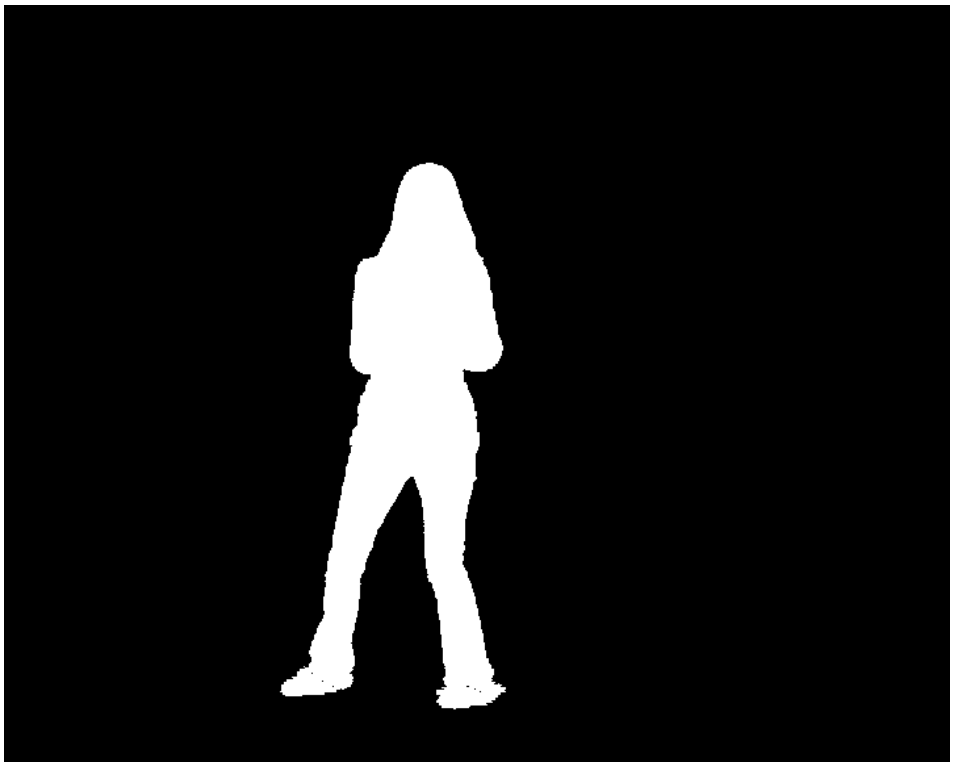

Figure 11 Binary mask produced by the segmentation

The image of the foreground and the background are produced for each frame:

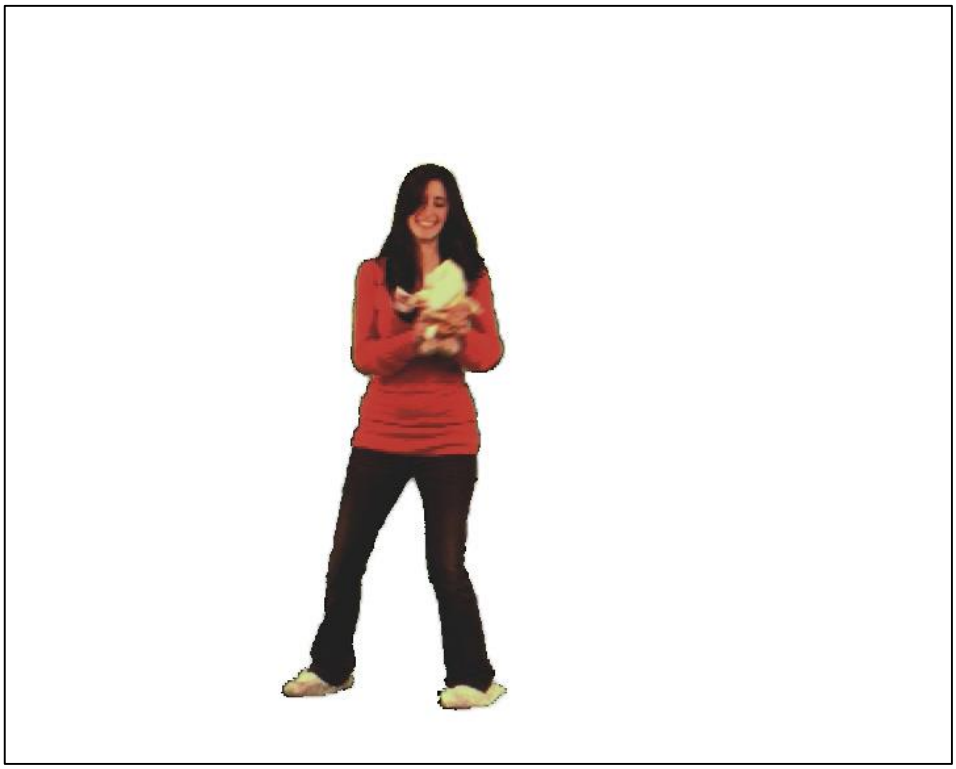

Figure 12 Foreground extracted from a frame

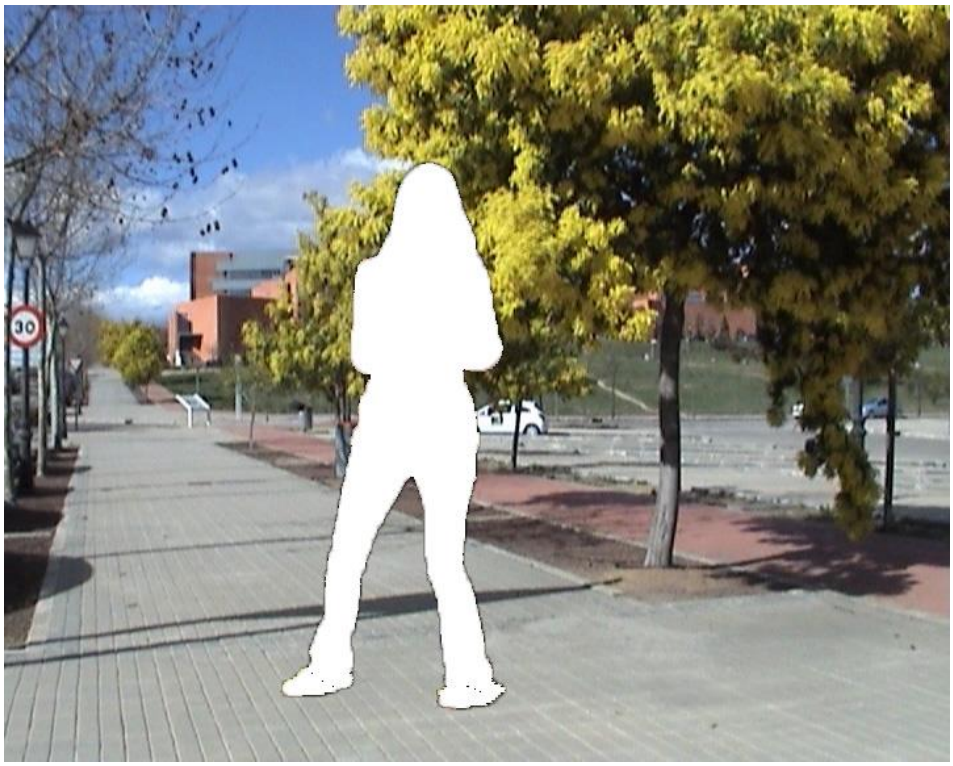

Figure 13 Background extracted from a frame

The figures above are only an example for a frame of a sequence, to better understand the results produced by the algorithm it would be useful to look at Chapter 4.

After the segmentation, another update of the models is required. This is done in order to avoid error propagation, as suggested in [9].

Also in this case the update takes place only for the spatial coordinates, so the means of the colour components for every k^{th} Gaussian, and the covariance matrix for the colour domain for every k^{th} Gaussian will remain the same as before.

So:

$$\Sigma_k(t) = \begin{bmatrix} \Sigma_{k_c}(t-1) & 0 \\ 0 & \Sigma_{k_s}(t) \end{bmatrix} \quad (42)$$

And

$$\mu_c(t) = \mu_c(t-1) \quad (43)$$

Where t is the actual frame, and $t-1$ is the previous one.

Once the segmentation and the subsequently spatial update are completed, a backup of the two Spatial Colour Gaussian Mixture Models is required in order to employ them for the next frame, as described at the beginning of Section 3.2.2.

Is possible to summarize the entire algorithm using the following flow charts in Figure 14 and in Figure 15, where the first one represent the global algorithm, meanwhile the second one describes in detail how the segmentation works for each frame starting from the second one.

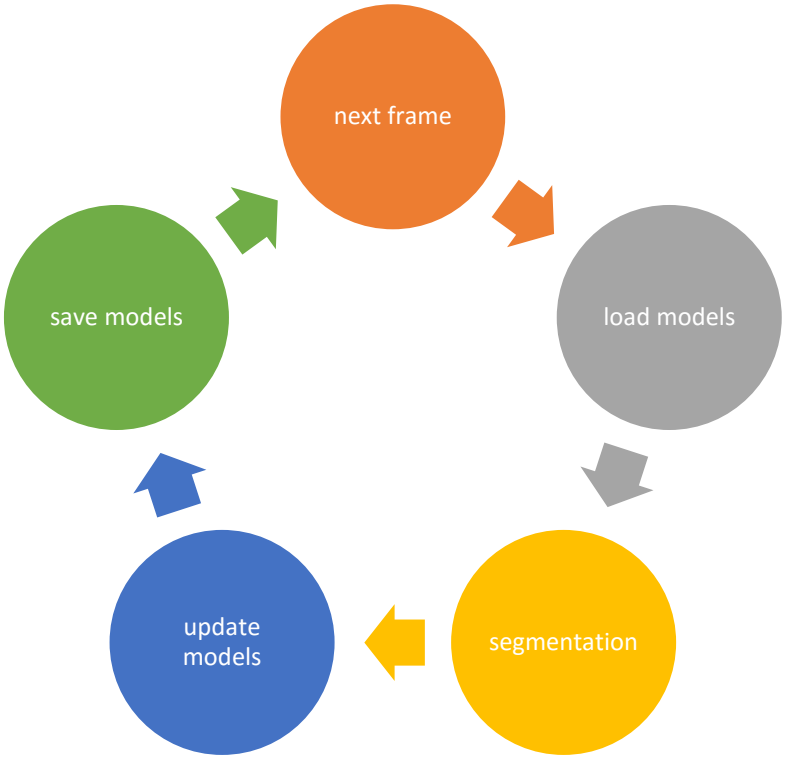

Figure 14 Flow chart of the entire segmentation process

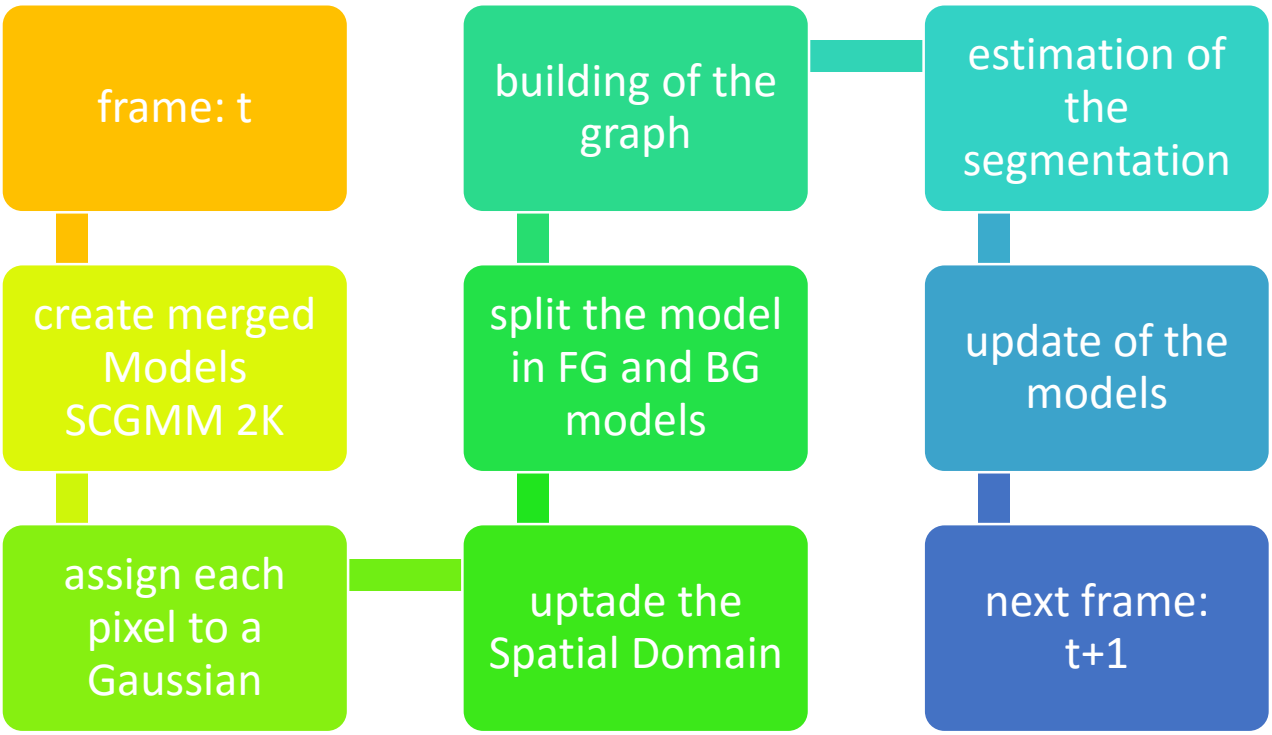

Figure 15 Flow chart of the segmentation

3.3 EVALUATION

Evaluation is a fundamental part in each project since the quality and the robustness of the system proposed can be measured.

In order to do this, qualitative and quantitative evaluations have been performed.

The metrics used in the evaluation, as it is done in [18], are:

- $Average\ pixel\ error\ rate\ per\ frame$,
- $Average\ pixel\ error\ rate\ per\ frame\ \%$,
- $Precision$ (mean and variance),
- $Recall$ (mean and variance),
- $f_{measure}$.

Which are formulated as follows:

- $Average\ Pixel\ Error\ Rate\ per\ Frame = \frac{XOR(S,GT)}{F}$,
- $Average\ Pixel\ Error\ Rate\ per\ Frame\ \% = \frac{Average\ pixel\ error\ rate\ per\ frame}{number\ of\ pixels\ per\ frame}$,
- $Precision = \frac{TP}{TP+FP}$,
- $Recall = \frac{TP}{TP+FN}$,
- $f_{measure} = 2 \times \frac{Precision \cdot Recall}{Precision+Recall}$.

where:

- S = segmented image,
- GT = ground truth of the segmented image,
- F = total frame number,
- TP = True Positives, pixels marked as Foreground which are foreground in the ground truth,

- FP = False Positives, pixels marked as Foreground which are background in the ground truth,
- FN = False Negatives, pixels marked as background which are foreground in the ground truth.

The $f_{measure}$ is a measure of a test's accuracy. It considers both the precision and the recall to compute the score. This measure can be interpreted as a weighted average of the precision and recall, where its best value is 1 and the worst is 0.

In order to compute the measures presented above two sets of images have been compared, one produced from the algorithm and one that is the ground truth.

The following figures are an example of the ground truth mask and the mask provided by the algorithm.

On the left the ground truth mask is presented, and on the right there is the output mask provided by the algorithm.

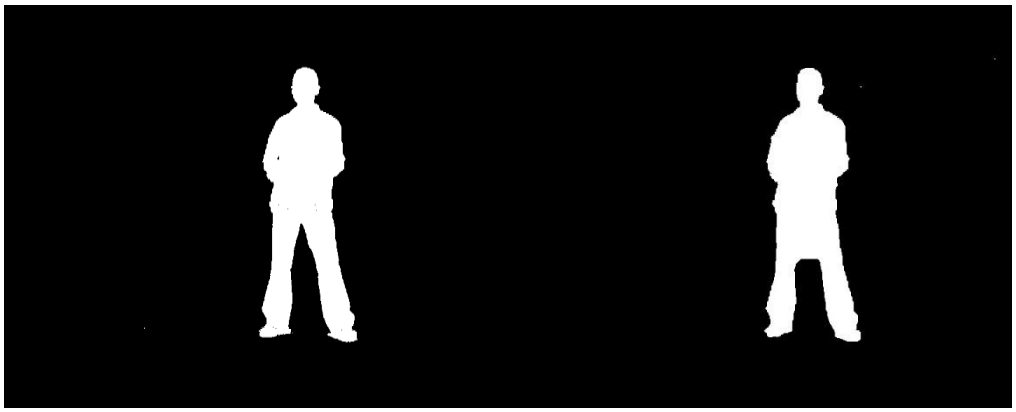

Figure 16 Ground truth mask and output mask

Another method to understand how the algorithm is performant is to look at the representation of the models, foreground and background, computed for each frame.

The subsequent figure is only an example of the representation of the computed models. The depicted ellipses on the image are the representation of the Gaussians that form the foreground and background models.

Red ellipses are for the foreground, and green ellipses are for the background. The thickness of the line that draws the ellipses is proportional to the weight of each single Gaussian.

Each ellipse is centred at a point $P(Px, Py)$ where the coordinates represent the mean value of the x and y value of the Gaussian.

The axis of the ellipse are the eigenvectors of the covariance matrix of the Gaussian distribution in the x and y dimensions. The size of the axis is proportional to the corresponding eigenvalues.

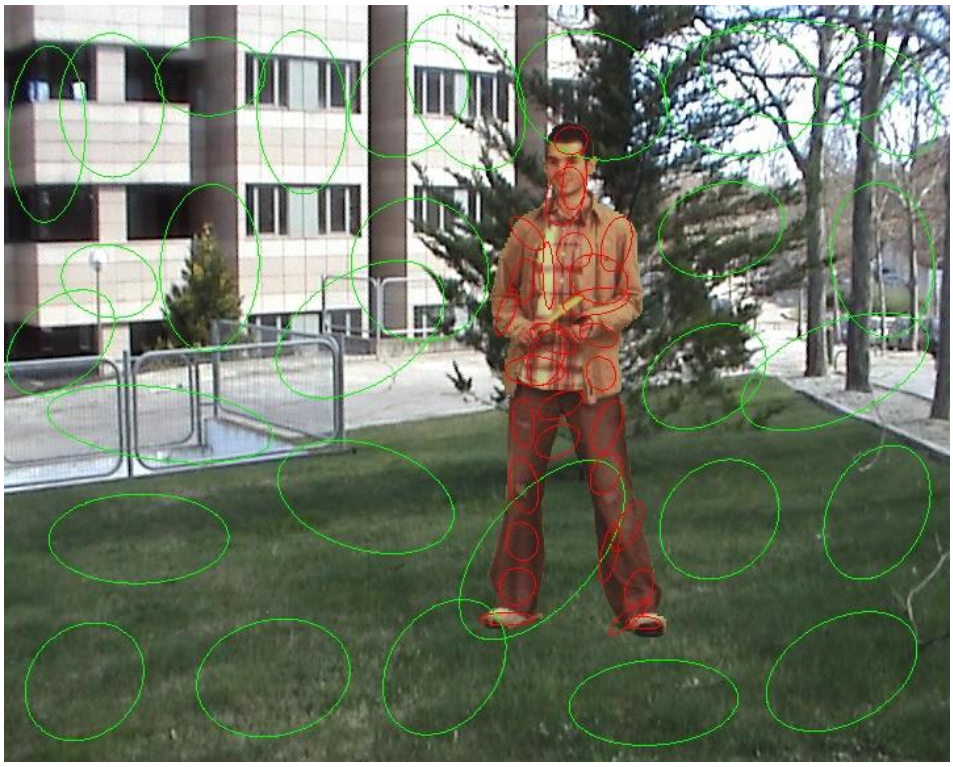

Figure 17 Representation of the FG and BG models

3.4 DEVELOPMENT ENVIRONMENT

The entire project has been developed using the C++ programming language exploiting the OpenCV library in its version 2.4.

Figure 18 OpenCV logo [13]

OpenCV [19], acronym of Open Source Computer Vision, is a library which contains programming functions for real time computer vision application.

The code has been written using the IDE[6] QtCreator [20] installed on a computer with ram memory of 6BG and an Intel® Core™ i5-3337U @ 1.80Ghz processor unit, running Ubuntu 14.04 LTS.

[6] IDE = Integrated Development Environment

4 RESULTS

As it is presented in Section 3.3 qualitative and quantitative evaluations have been performed.

The sequences used for this project were obtained from the cVSG Public Database [21] and from the SegTrack Database [22].

In the following pages are presented, in detail the results obtained for each sequence using the Test Mode.

Per each sequence is presented the drawings of the model computed for a frame. The models are represented as explained in Section 3.3 with red ellipses for the foreground and green ellipses for the background. The output mask obtained with the segmentation for the same frame is also presented. At the end, for each sequence, there is a table which presents the qualitative measure for the sequence.

A comparison between the results obtained with the proposed method and other State of Art algorithm is proposed in Section 4.2, at the end of this Chapter.

4.1 RESULTS

The sequences, form the cVSG Public Database [21], used to test the algorithm proposed are:

- Exhausted runner,
- Hot day,
- Playing alone,
- Teddy bear,
- Thirsty man.

And from the SegTrack Database [22], the sequences used are:

- Bird fall,
- Cheetah,
- Girl,
- Monkey dog,
- Parachute,
- Penguin.

4.1.1 Exhausted runner

Figure 19 Example of the model computed for a frame: Exhausted runner

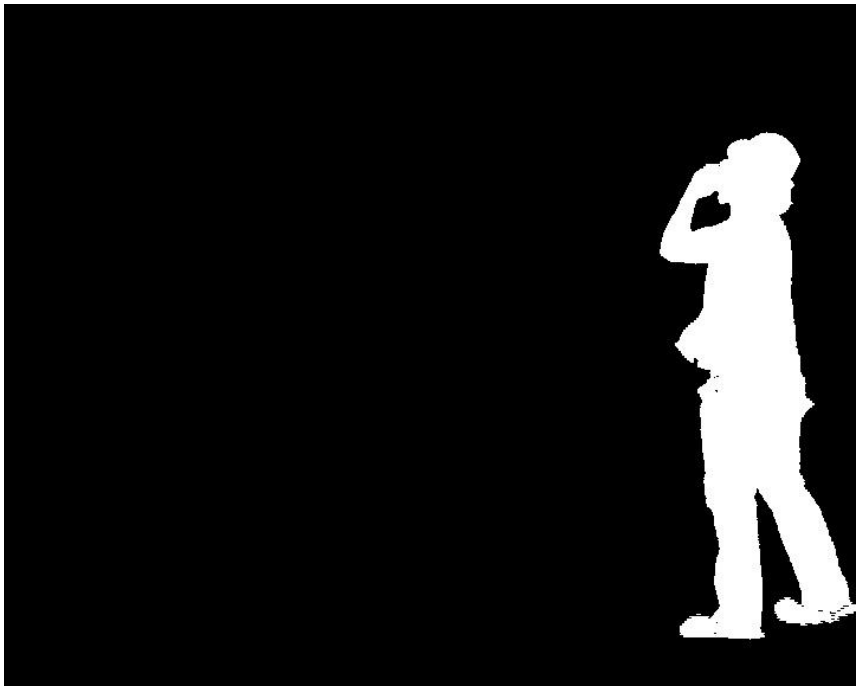

Figure 20 Mask obtained from the segmentation: Exhausted runner

Measure	Value
Number of frames	29
Average Pixel Error Rate per Frame	5034
Average Pixel Error Rate per Frame %	1.214%
Precision [mean]	0.994
Precision [variance]	0.000
Recall [mean]	0.994
Recall [variance]	0.000
f measure	0.994

Table 1 Quality measures of: Exhausted runner

For this sequence the quality results are pretty good. There are 5034 wrong pixels per each frame on average, but having each frame a big dimension ($720x576$, in total $414\ 720$ pixels) the percentage of wrong pixels is only 1.214% per each frame.

Having a high score in both the $precision$ and $recall$ also the $f_{measure}$ has a quite high score.

4.1.2 Hot day

Figure 21 Example of the model computed for a frame: Hot day

Figure 22 Mask obtained from the segmentation: Hot day

Measure	Value
Number of frames	51
Average Pixel Error Rate per Frame	4898
Average Pixel Error Rate per Frame %	1.181%
Precision [mean]	0.982
Precision [variance]	0.00
Recall [mean]	0.994
Recall [variance]	0.00
f measure	0.988

Table 2 Quality measures of: Hot day

Also for this sequence the quality results are pretty good. There are on average only 4898 wrong pixels per each frame, and with the frame's dimension of $720x576$, in total $414\ 720$ pixels, the percentage of wrong pixels is only 1.181% per each frame.

Having a high score in both the $precision$ and $recall$ also the $f_{measure}$ has a quite high score.

4.1.3 Playing alone

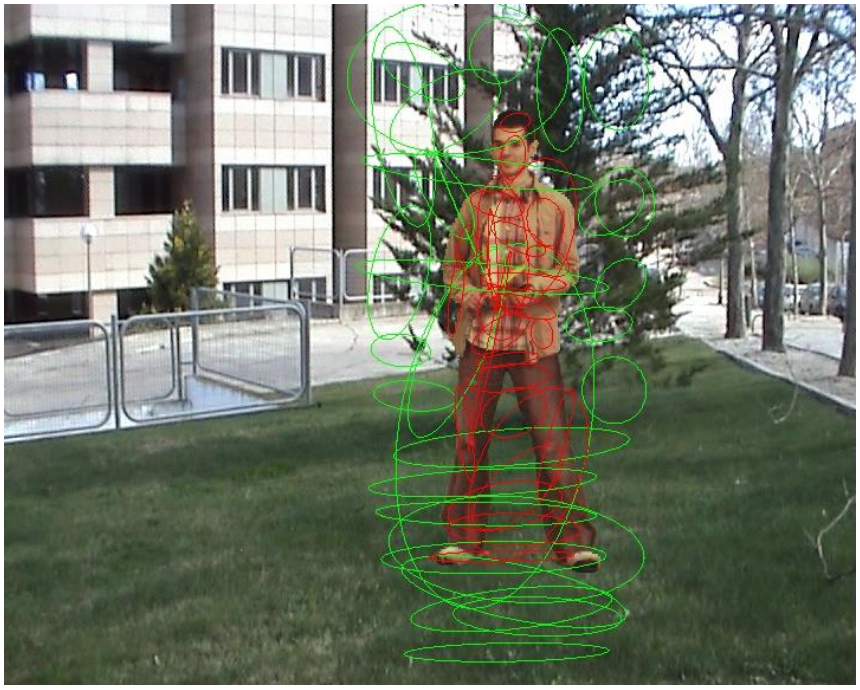

Figure 23 Example of the model computed for a frame: Playing alone

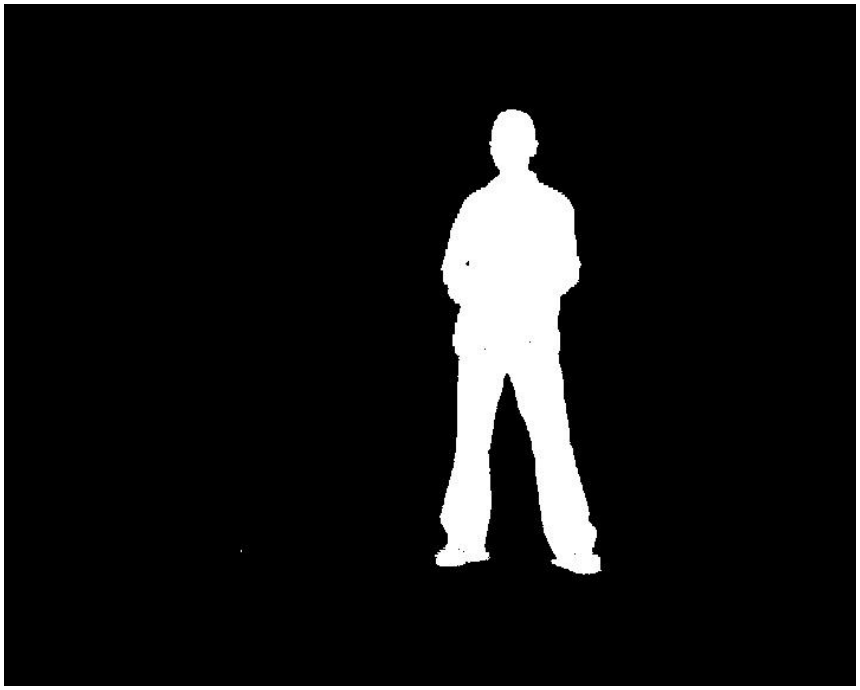

Figure 24 Mask obtained from the segmentation: Playing alone

Measure	Value
Number of frames	81
Average Pixel Error Rate per Frame	5055
Average Pixel Error Rate per Frame %	1.219%
Precision [mean]	0.983
Precision [variance]	0.000
Recall [mean]	0.982
Recall [variance]	0.000
f measure	0.982

Table 3 Quality measures of: Playing alone

Same for the previous sequences, also for this one the quality results are quite good. The percentage of wrong pixels remains low, only 1.219% per each frame.

With a $precision$ score of 0.983 and $recall$ of 0.982, also the $f_{measure}$ results 0.982.

4.1.4 Teddy bear

Figure 25 Example of the model computed for a frame: Teddy bear

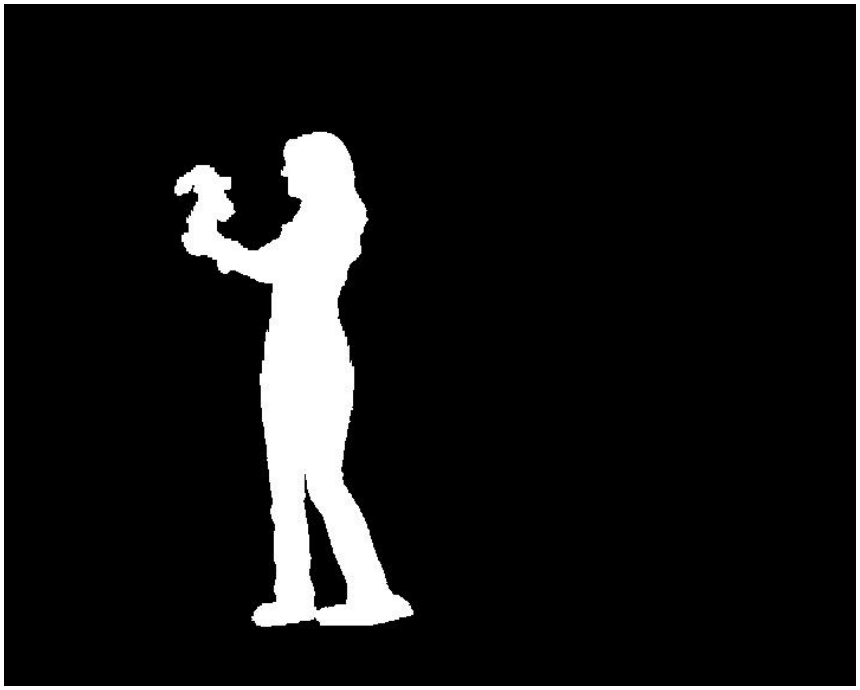

Figure 26 Mask obtained from the segmentation: Teddy bear

Measure	Value
Number of frames	117
Average Pixel Error Rate per Frame	5271
Average Pixel Error Rate per Frame %	1.271%
Precision [mean]	0.986
Precision [variance]	0.000
Recall [mean]	0.992
Recall [variance]	0.000
f measure	0.989

Table 4 Quality measures of: Teddy bear

Also in this case the quality measures are decent. Only 1.271% of pixels per each frame is wrong and with a $precision$ score of 0.986 and $recall$ of 0.992, the $f_{measure}$ score is 0.989.

4.1.5 Thirsty man

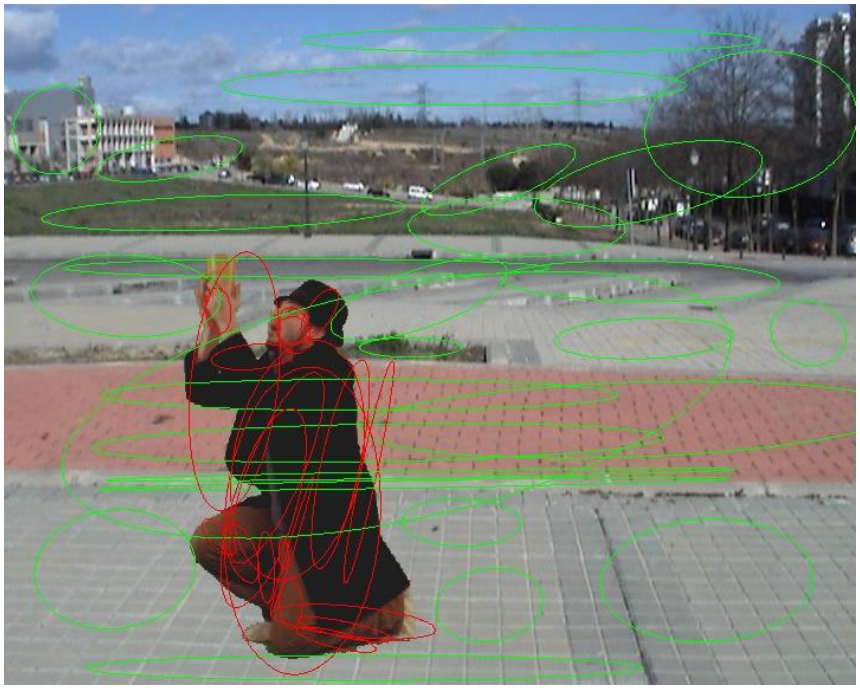

Figure 27 Example of the model computed for a frame: Thirsty man

Figure 28 Mask obtained from the segmentation: Thirsty man

Measure	Value
Number of frames	22
Average Pixel Error Rate per Frame	3894
Average Pixel Error Rate per Frame %	0.939%
Precision [mean]	0.998
Precision [variance]	0.000
Recall [mean]	0.998
Recall [variance]	0.000
f measure	0.998

Table 5 Quality measures of: Thirsty man

This sequence is the one that has the less number of wrong pixels and so also the percentage is the lower with only 1.181% wrong pixels per each frame.

Due to this also the $precision$ score of 0.998 and $recall$ of 0.998, giving a $f_{measure}$ of 0.998 is the best score for the tested sequences.

A thing that can be noticed in Figure 27 is that the background model is not restricted to the close area around the foreground. This problem is due to errors produced by the tracking algorithm which in some cases produce a bounding box too large with respect to the real dimension of the foreground.

4.1.6 Bird fall

Figure 29 Example of the model computed for a frame: Bird fall

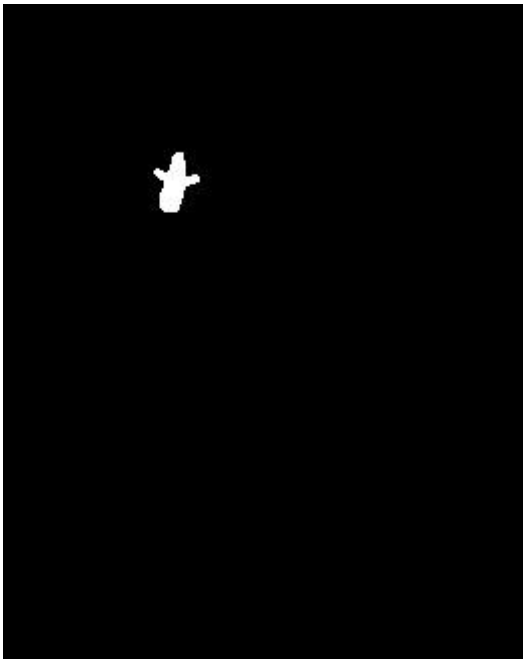

Figure 30 Mask obtained from the segmentation: Bird fall

Measure	Value
Number of frames	30
Average Pixel Error Rate per Frame	5967
Average Pixel Error Rate per Frame %	7.045%
Precision [mean]	0.373
Precision [variance]	0.128
Recall [mean]	0.855
Recall [variance]	0.007
f measure	0.519

Table 6 Quality measures of: Bird fall

The segmentation of the sequence "Bird fall" produces a quite high $Average\ Pixel\ Error\ Rate\ per\ Frame$, and due to the small dimensions of each frame, $259x327$ pixels, the percentage of error on average per each frame is 7.045%.

This produce a low score on the $precision$, only 0.373. With this value of the $precision$ alse the $f_{measure}$ score is quite low, only 0.519.

Analysing in detail the $precision$ score, it is possible to say that with such a low value is achieved because of a high number of $False\ Positives$. This can be explained considering that for the last 14 frames of the sequence, the tracking method is not very accurate and the area include in the Bounding box obtained from CAMSHIFT is too large and it includes also parts of the tree. In this way the segmentation method sees the tree as a possible area where the foreground can be found, and due to the large similarity between the colours of the bird and the tree, the result is not correct as can be seen in Figure 31 and Figure 32.

Figure 31 Model computed for frame 23 of Bird fall

Figure 32 Segmentation mask obtained for frame 23 of Bird fall

A way to improve the outcomes with this sequence is to change the tracking method. For example, in this sequence, is possible to track the object using in

the frame $t + 1$ the position of the object in the frame t adding at this coordinates a shift in the x and y direction, avoiding the CHAMSHIFT algorithm.

Using this method better results are obtained than using the CAMSHIFT algorithm.

Measure	Value
Number of frames	30
Average Pixel Error Rate per Frame	746
Average Pixel Error Rate per Frame %	0.881%
Precision [mean]	0.601
Precision [variance]	0.022
Recall [mean]	0.928
Recall [variance]	0.004
f measure	0.730

Table 7 Quality measures of: Bird fall with different tracking method

Figure 33 Model computed for frame 23 of Bird fall with different tracking method

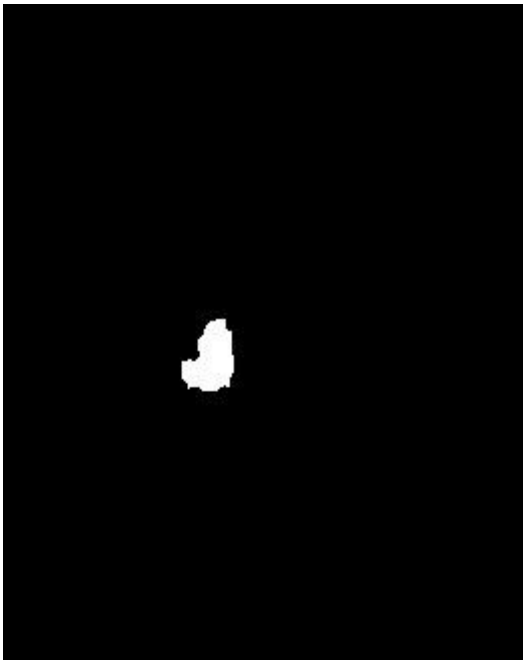

Figure 34 Segmentation mask obtained for frame 23 of Bird fall with different tracking method

4.1.7 Cheetah

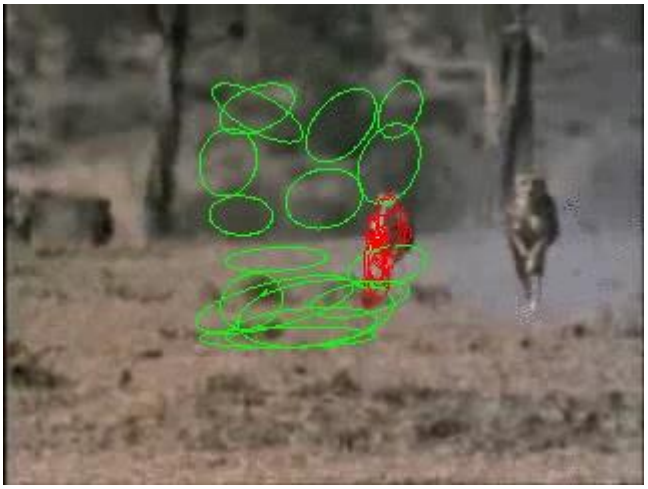

Figure 35 Example of the model computed for a frame: Cheetah

Figure 36 Mask obtained from the segmentation: Cheetah

Measure	Value
Number of frames	29
Average Pixel Error Rate per Frame	2163
Average Pixel Error Rate per Frame %	2.816%
Precision [mean]	0.844
Precision [variance]	0.070
Recall [mean]	0.320
Recall [variance]	0.043
f measure	0.465

Table 8 Quality measures of: Cheetah

In this case the $Average\ Pixel\ Error\ Rate\ per\ Frame\ \%$ per each frame is 2.816%, less than in "Bird fall". The $precision$, with a score of 0.844 is quite good, the $recall$ has a low score, only 0.320. This means that there are a lot of $False\ Negatives$. $False\ Negative$ means that a pixel is assigned to the background instead to the foreground as it should be. This is verified for example in frame 28 as can be seen in Figure 37.

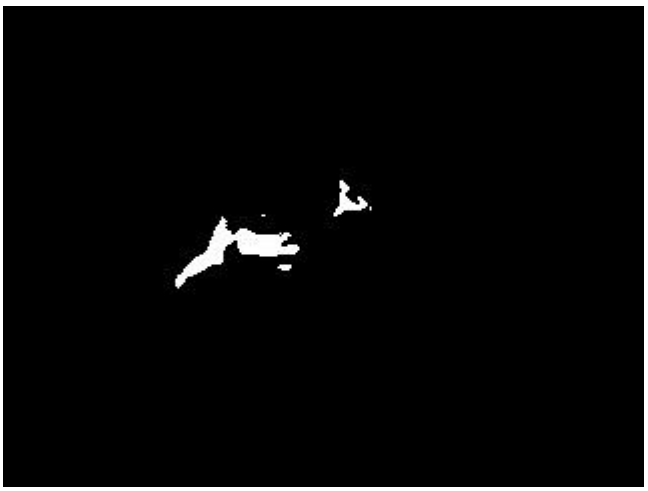

Figure 37 Mask obtained from the segmentation of frame 28 of Cheetah

The $f_{measure}$, due to the low score of the $recall$ has a value of 0.465.

4.1.8 Girl

Figure 38 Example of the model computed for a frame: Girl

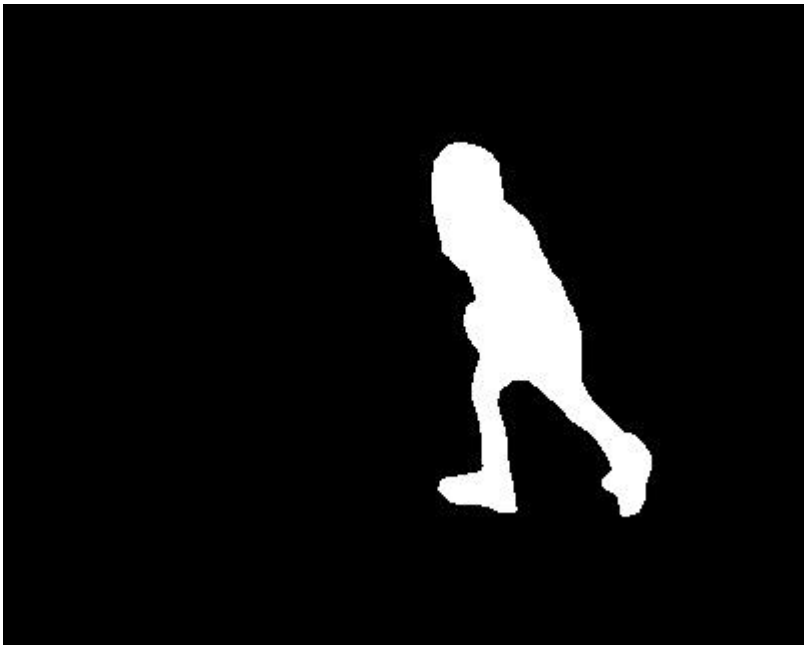

Figure 39 Mask obtained from the segmentation: Girl

Measure	Value
Number of frames	21
Average Pixel Error Rate per Frame	9396
Average Pixel Error Rate per Frame %	7.341%
Precision [mean]	0.657
Precision [variance]	0.031
Recall [mean]	0.687
Recall [variance]	0.011
f measure	0.672

Table 9 Quality measures of: Girl

This sequence has a quite high percentage of wrong pixels per frame, 7.341%.

The bigger problem for this sequence are the arms of the girl, that are continuously moving for all the sequence. The arms are not correctly segmented for all the sequence.

4.1.9 Monkey dog

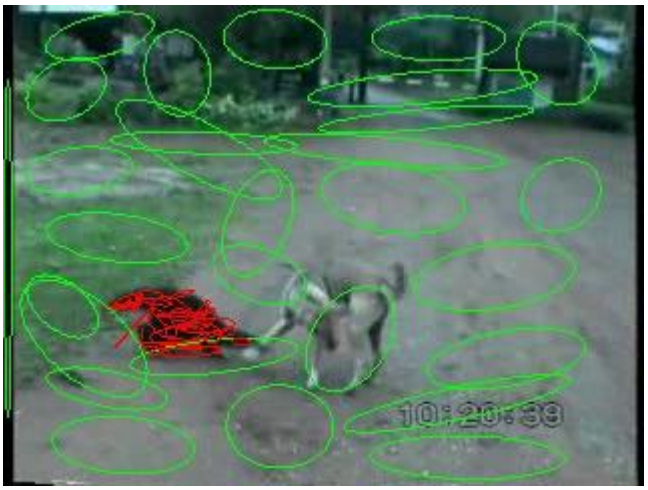

Figure 40 Example of the model computed for a frame: Monkey dog

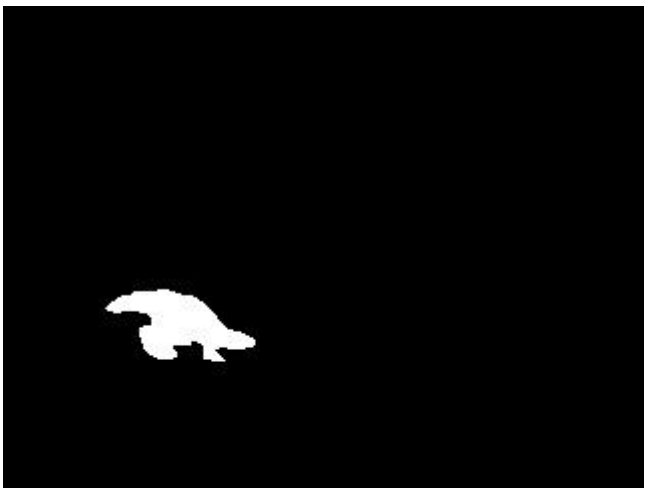

Figure 41 Mask obtained from the segmentation: Monkey dog

Measure	Value
Number of frames	27
Average Pixel Error Rate per Frame	1067
Average Pixel Error Rate per Frame %	1.389%
Precision [mean]	0.939
Precision [variance]	0.000
Recall [mean]	0.833
Recall [variance]	0.006
f measure	0.883

Table 10 Quality measures of: Monkey dog

In this case the percentage of wrong pixels per each frame is only 1.389%. This fact leads to a quite high $precision$ score of 0.939 and a $recall$ score of 0.833, that produce a $f_{measure}$ of 0.883.

4.1.10 Parachute

Figure 42 Example of the model computed for a frame: Parachute

Figure 43 Mask obtained from the segmentation: Parachute

Measure	Value
Number of frames	24
Average Pixel Error Rate per Frame	969
Average Pixel Error Rate per Frame %	0.665%
Precision [mean]	0.987
Precision [variance]	0.002
Recall [mean]	0.959
Recall [variance]	0.001
f measure	0.973

Table 11 Quality measures of: Parachute

The segmentation of Parachute sequence produces the best result in comparison with the other sequences of the same data base.

With only 0.665% of wrong pixels per frame on average, this sequence has also a very good score in the $precision$ measure with 0.987 and in the $recall$ measure with 0.959, this leads to a $f_{measure}$ score of 0.973.

4.1.11 Penguin

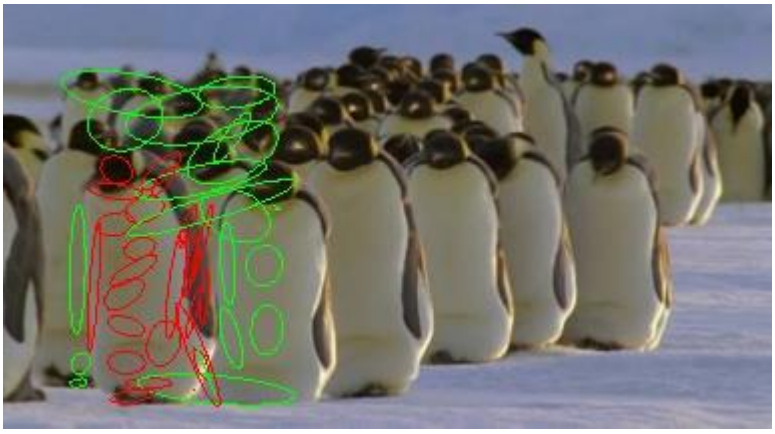

Figure 44 Example of the model computed for a frame: Penguin

For this sequence, after the correct computation of the foreground and background models on the first frame, due to the good initialization, a problem occurs.

The penguin to track is the first on the left, since the close background is too similar to the desired object in foreground, the tracking method enlarge too much the Bounding box which contains the foreground and it starts to track the entire group of penguins.

For the same reason, the colour model computed for the background in the first frame can be applied to all the group of penguins, and also the segmentation fails.

Hence no result about this sequence is presented here.

The above Figure, is the result of the computation of the models for foreground and background on the first frame, the unique frame where the computation takes place correctly due to the good initialization.

4.2 COMPARISON WITH OTHER METHODS

4.2.1 cVSG Public Database [21]

	Proposed method			**Gallego** [10]		
Sequence	Precision	Recall	f measure	Precision	Recall	f measure
Exhausted runner	0.994	0.994	0.994	0.986	0.985	0.986
Hot day	0.982	0.994	0.988	0.980	0.985	0.983
Playing alone	0.983	0.982	0.982	0.997	0.984	0.990
Teddy bear	0.986	0.992	0.989	0.916	0.981	0.948
Thirsty man	0.998	0.998	0.998	-	-	-

Table 12 Comparison sequence of cVSG Public Database [21]

As it can be observed from Table 12 our method achieves a quite high $f_{measure}$ score in all the sequences analysed. With respect to the method proposed by Gallego [10], which is described in Section 2.4, we have obtained in better result in sequences “Exhausted runner”, “Hot day” and “Teddy bear”. Instead, with the sequence “Playing-alone”, Gallego [10] has obtained a better $f_{measure}$ score than us.

4.2.2 SegTrack Database [22]

	Proposed method			**Varas** [18]		
Sequence	Precision	Recall	f measure	Precision	Recall	f measure
Bird fall	0.37	0.86	0.52	0.86	0.70	0.77
Bird fall [no CAMSHIFT]	0.60	0.93	0.73	0.86	0.70	0.77
Cheetah	0.84	0.32	0.46	0.85	0.77	0.81
Girl	0.66	0.69	0.67	0.76	0.72	0.74
Monkey dog	0.94	0.83	0.88	0.78	0.73	0.75
Parachute	0.99	0.96	0.97	0.99	0.89	0.94

Table 13 Comparison sequence of SegTrack Database [22]

For the sequences from SegTrack Database [22], Varas method [18] has been used to compare the results.

Varas presents a video object segmentation approach that extends the particle filter to a region-based image representation. Particles are defined as unions of regions in the current image partition and their propagation is computed through a single co-clustering. Image partition is considered part of the particle filter measurement, which enriches the available information. The prediction step uses a co-clustering between the previous image object partition and a partition of the current one, which allows to tackle the evolution of non-rigid structures.

Further details about Varas method can be found in [18].

As it can be seen in Table 13, with this database, Varas method has obtained better $f_{measure}$ scores with sequences “Bird fall”, “Cheetah” and “Girl”.

Instead, with sequences “Monkey dog” and “Parachute” our method has obtained a better $f_{measure}$ score than Varas.

Analysing in detail the sequence “Bird fall” it is possible to affirm that we have achieved a better $recall$ score that Varas, it means that we have obtained a less number of $False\ Negatives$. Instead our $precison$ score is worse than the one

achieved by Varas, it means that we have obtained a high number of $False\ Positives$.

As can be seen for “Bird fall [no CAMSHIFT]”, the version analysed with a different tracking method ad described in Section 4.1.6, our $f_{measure}$ score is quite close to the score obtained by Varas, this is due to the improvement obtained in the $precision$ score.

Instead, for the sequence “Cheetah” our $recall$ score is half the one achieved by Varas, and as explained above, this means that we have obtained a high number of $False\ Negatives$.

5 CONCLUSIONS

In the development of this Thesis entitled Video Object Tracking using foreground models, have been studied State of the Art approaches like "*Monocular Video Foreground/Background Segmentation by Tracking Spatial-Color Gaussian Mixture Models*", "*Foreground object segmentation for moving camera scenarios based on SCGMM*", and "*Video Snap Cut: robust Video Object Cutout Using Localized Classifiers".*

We have also studied algorithms like *CAMSHIFT* and *GrabCut* which have been used as a starting point for development of the project.

After the study of the above approaches and algorithm has been developed the method used for the project which makes use of the Spatial Colour Gaussian Mixture Models to describe the foreground and the background from each frame of a video sequence.

Afterwards, the algorithm proposed for the segmentation has been tested using sequences for two different data bases, SegTrack Data Base and cVSG Public Data Base.

As presented and explained in the Chapter 4 the best qualitative results are obtained segmenting the sequences from cVSG Public Data Base, with scores that are above or close to the ones from Gallego's approach [10].

With the sequences from SegTrack Data base, the results are not so good for the sequences “Bird fall” and “Cheetah”, but for “Bird fall [no CAMSHIFT]”, “Girl” and “Money dog” they are close to the Varas’s method [18] scores.

For the sequence “Parachute”, also it from SegTrack Data Base, the quality results are better than the one from the approach used to compare.

As said in Section 4.1.6, a different tracking method has been used for the sequence “Bird fall [no CAMSHIFT]”. This method improves the quality results for this sequence, but it cannot be used in different sequences like “Cheetah”.

It cannot be used since the object changes its shape and dimension and the tracking doesn’t cope with this problem. For the sequence “Bird fall [no CAMSHIFT]” it is not a problem since, the bird has always the same shape and dimension.

A consideration about the time performance is due: for the method proposed the time efficiency is not a strong point. The execution of the code is slow and it is not exploitable for real time applications like video conferences.

Since the purposes of this method are not real time applications, the problem of the slowness of the execution could be not a considerable issue, but some improvements can be done as described in Chapter 6.

6 FUTURE DEVELOPMENTS

To continue the work, the main points on which an attention is need is the tracking part.

As described in Section 4.1.11, for the sequence “Penguin”, the tracking fails after one frame due to the large similarity between the object chosen to follow and its close background.

Similar problems occur for the sequence “Bird fall”.

So, an improvement could be done in this context, for example changing the tracking method instead of using CAMSHIFT.

As described in Section 4.1.6 for the sequence “Bird fall [no CAMSHIFT]”, CAMSHIFT has been avoided, and to track the object in frame $t + 1$ has been used its position in the previous frame t with a shift in the x and y coordinates. This approach is not suitable for all the sequences, since in this case the dimension of the object doesn’t change, but in other sequences does.

Another improvement that could be done is the enhancement of the time efficiency of the code, since as mentioned in Chapter 5 the code is not suitable for real time applications.

In order to obtain a faster execution of the code a compromise between the quality results and the velocity should be accepted.

Decreasing for example the number of components which are describing the SCGMMs for foreground and background, it will reduce the execution time. Another thing that can be done in order to speed up the execution of the code, is to decrease the number of iteration which are done to compute and update the SCGMMs.

7 BIBLIOGRAPHY

[1] R. M. Haralick and L. G. Shapiro, “Image Segmentation Techiniques,” 1982.

[2] A. Levinshtein, C. Sminchisescu and S. Dickinson, “Spatiotemporal Closure,” in *Computer Vision – ACCV 2010*, 2010, pp. 369-382.

[3] A. Yilmaz, O. Javed and M. Shah, “Object Tracking: A Survey,” *ACM Computing Surveys,* vol. 38, no. 4, 2006.

[4] C. Rother, V. Kolmogorov and A. Blake, “GrabCut: Interactive foreground extraction using iterated graph cuts,” 2004.

[5] Y. Y. Boykov and . M.-P. Jolly, “Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images,” in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference*, 2001.

[6] Y. Boykov, O. Veksler and R. Zabih, “Fast Approximate Energy Minimization via Graph Cuts,” vol. IEEE Trans. Pattern Analysis and Machine Intelligence, 2001.

[7] G. Bradski, Real time face and object tracking as a component of a perceptual user interface, 1998.

[8] D. Comaniciu and P. Meer, “Mean shift: A robust approach toward feature space,” in *IEEE Transaction on Pattern Analysis and Machine Intellgence*, 2002.

[9] T. Yu, C. Zhang, M. Cohen, Y. Rui and Y. Wu, “Monocular Video Foreground/Background Segmentation by Tracking Spatial-Color Gaussian Mixture Models,” in *IEEE Workshop on Motion and Video Computing*, 2007.

[10] J. Gallego, M. Pardas and M. Solano, “Foreground Objects Segmentation for Moving Camera Scenarios Based on SCGMM,” *Computational Intelligence for Multimedia Understanding,* 2012.

[11] X. Bai, J. Wang, D. Simons and G. Sapiro, “Video SnapCut: Robust Video Object Cutout Using Localized Classifiers,” *ACM Trans. Graph,* 2009.

[12] G. H. Joblove and D. Greenberg, “Color spaces for computer graphics,” in *SIGGRAPH '78 Proceedings of the 5th annual conference on Computer graphics and interactive techniques*, New York, 1978.

[13] J. W. Shipman, “Introduction to color theory,” [Online]. Available: http://infohost.nmt.edu/tcc/help/pubs/colortheory/web/index.html.

[14] O. Documentation, “http://docs.opencv.org/,” [Online]. Available: http://docs.opencv.org/.

[15] S. Iyanaga and Y. Kawada, Encyclopedic Dictionary of Mathematics, MIT Press, 1977.

[16] J. F. Talbot and X. Xu, “Implementing GrabCut,” 2006.

[17] “https://www.cs.auckland.ac.nz,” [Online].

[18] D. Varas and F. Marques, “Region-based particle filter for video object segmentation,” in *CVPR - Computer Vision and Pattern Recognition*, Ohio, 2014.

[19] “OpenCV,” [Online]. Available: www.opencv.org.

[20] “QtCreator,” [Online]. Available: http://www.qt.io/.

[21] F. Tiburzi, M. Escudero, J. Bescós and J. M. Martínez, “A ground truth for motion-based video-object segmentation,” in *15th IEEE International Conference*, 2008.

[22] D. Tsai, M. Flagg, A. Nakazawa and J. M. Rehg, “Motion Coherent Tracking Using Multi-label MRF Optimization,” *International Journal of Computer Vision,* pp. 190-202, November 2012.

[23] Y.-Y. Chuang, B. Curless, D. H. Salesin and R. Szeliski, A Bayesian approach to digital matting, Proc. IEEE Conf. Computer Vision and Pattern Recognition, 2001.