UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

REGION BASED PARTICLE FILTER

A Degree's Thesis

Submitted to the Faculty of the

Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona

Universitat Politècnica de Catalunya

by

Andreu Girbau Xalabarder

In partial fulfilment

of the requirements for the degree in

Sistemes Audiovisuals **ENGINEERING**

Advisor: Ferran Marqués Acosta

David Varas González

Barcelona, February 2015

Abstract

In this project the implementation of a video object tracking technique based on a particle filter that uses the partitions of the various frames in the video has been tackled. This is an extension of the standard particle filter tracker in which unions of regions of the image are used to generate particles. By doing so, the tracking of the object of interest through the video sequence is expected to be done in a more accurate and robust way.

One of the main parts of this video object tracker is a *co-clustering* technique that allows having an initial estimation of the object in the current frame, relying on the instance of the same object in a previous frame. While developing the object tracker, we realized the importance of this co-clustering technique, not only in the context of the current video tracker but as a basic tool for several of the research projects in the image group. Therefore, we decided to concentrate on the implementation of a generic, versatile co-clustering technique instead of the simple version that was necessary for the tracking problem. This way, the main goal of this project consists on implementing the *co-clustering* method presented by [4] in an accurate way while presenting a low computation time. Moreover, the complete Region-based particle filter for tracking purposes is presented.

Therefore, the aim of this Final Degree Project is, mainly, to give a guideline to future researchers who will use this algorithm; to help understand and apply the mentioned *co-clustering* for any project in need of this method.

Resum

En aquest projecte es presenta la implementació d'una tècnica de seguiment en vídeo basada en un filtre de partícules que empra les particions de diversos fotogrames. Això és una extensió del filtre de partícules estàndard per seguiment, on les unions de les regions de la imatge són utilitzades per generar les partícules. Mitjançant això, s'espera que el seguiment de l'objecte d'interès a través de la seqüència de vídeo sigui realitzat d'una manera més acurada i robusta.

Una de les parts principals d'aquest seguidor d'objectes en vídeo és la tècnica de *co-clustering*, que permet tenir una estimació inicial de l'objecte en el fotograma actual basant-se en l'objecte del fotograma anterior. Mentre desenvolupàvem el seguidor d'objectes ens vam adonar de la importància d'aquesta tècnica de *co-clustering*, no només en el context del seguidor per vídeo actual sinó, també, com a eina bàsica per nombrosos projectes de recerca en el grup d'imatge. D'aquesta manera, vam decidir concentrar-nos en la implementació d'una tècnica de co-clustering genèrica i versàtil en comptes d'una versió simplificada necessària pel problema del seguiment. L'objectiu principal d'aquest projecte, per tant, consisteix en implementar el mètode de *co-clustering* presentat per [4] d'una manera acurada, tot presentant un temps de computació reduït. A més a més el propi "Region-based particle filter" pel propòsit de seguiment és presentat també.

L'objectiu d'aquest Projecte Final de Carrera és, principalment, donar una pauta a futurs investigadors que faran servir aquest algorisme; per ajudar a entendre i aplicar el mencionat *co-clustering* per qualsevol projecte que necessiti d'aquest mètode.

Resumen

En éste proyecto se presenta la implementación de una técnica de seguimiento en video basada en un filtro de partículas que utiliza las particiones de varios fotogramas. Esto es una extensión del filtro de partículas estándar para seguimiento, dónde las uniones de la regiones de la imagen son utilizadas para generar las partículas. Mediante esto se espera que el seguimiento del objeto de interés a través de la secuencia de video sea realizado de una manera más precisa y robusta.

Una de las partes principales de éste seguidor de objetos en video es la técnica de *co-clustering*, que permite tener una estimación inicial del objeto en el fotograma actual basándose en el objeto del fotograma anterior. Durante el desarrollo del seguidor de objetos nos percatamos de la importancia de esta técnica de *co-clustering*, no solo en el contexto del seguidor de video actual, sino como herramienta básica para numerosos proyectos de investigación en el grupo de imagen. De esta forma, decidimos concentrarnos en la implementación de una técnica de co-clustering genérica y versátil en vez de una versión simplificada necesaria para el problema de seguimiento. El objetivo principal de éste proyecto, por lo tanto, consiste en implementar el método de *co-clustering* presentado por [4] de una manera precisa y presentando un tiempo de computación reducido. Además, el propio "Region-based particle filter" para el propósito de seguimiento es presentado también.

El objetivo de éste Proyecto Final de Carrera es, principalmente, dar una pauta a futuros investigadores que utilizarán éste algoritmo; para ayudar a entender y aplicar el mencionado *co-clustering* en qualquier proyecto que necesite de éste método.

I quan vam acabar d'escalar la muntanya en veiérem una altra més enllà.
I era més gran. I era més maca. Ostres.

Acknowledgements

First things first. There are so many people I would like to thank something. The education, the companionship, the motivation to keep forward… as I can't express it one by one, I hope this page will do the work for me.

To my advisors Ferran Marques and David Varas. Both knowing me for my insistency, I would like to thank the support, time, patience, knowledge, more patience and the opportunity and trust to do this project.

To the UPC staff and to the professors, to make someone wanting to overcome again and again; thank you for the knowledge and the feeling of impotence in some cases, which lead us to what we are now. Here, I would like to mention professor José Gómez, who passed this year, thank you for making my first in this School a great year.

To my colleagues and to my friends. A list of all the people who is important to me in some way would be too extensive, so I would like to thank you all for these years. Also a special mention to *Els competitius* from Barcelona, who made this 4 years a pleasure and to the *Leones team* in Girona, who always gave their support when I needed it and their understanding when I was away from home.

To that special person. That special person who has been at my side all this time from the first day of arrival, who had the perseverance to make me grow intellectually and as a person. Without your help I most likely wouldn't be here today. Thank you.

To my family. To Lluís, Eulàlia, Maria, Joan, Mercè, Joaquim, Anna, Pere, Xavier, Núria and Tomàs. Thank you. For everything.

Andreu Girbau Xalabarder

Revision history and approval record

Revision	Date	Purpose
0	26/12/2014	Document creation
1	25/01/2015	Document revision
2	31/01/2015	Document revision
3	03/02/2015	Document revision
4	05/02/2015	Document revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Andreu Girbau Xalabarder	Andreu.girbau@alu-etsetb.upc.edu
Ferran Marqués Acosta	Ferran.marques@upc.edu
David Varas González	David.varas@upc.edu

Written by: Andreu Girbau Xalabarder		Reviewed and approved by: Ferran Marqués Acosta David Varas González	
Date	26/12/2014	Date	31/01/2015
Name	Andreu Girbau Xalabarder	Name	Ferran Marqués Acosta David Varas González
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables:

List of Algorithms:

1. Introduction

1.1. Motivation

Object tracking is required in many applications. Surveillance, human-machine interaction and visual effects generation are just a few examples of its usages. The main drawback of tracking methods is that the tracked object is often estimated as a geometrical shape (such as an ellipse or rectangle), so only the position and the bounding box of the object are the output of the algorithm. Although these algorithms can work in real time, their accuracy may not be enough in certain applications.

Systems that segment objects from images are used to obtain more accurate information from partitions. These systems provide the silhouette of the objects but the computation time is high. This restriction makes these algorithms unpractical for almost any application that needs quick results.

The aim of this project, initially, was to combine the two previous techniques in order to obtain both types of information, the position of the object of interest and its shape accurately computed, with the less possible computation time.

As the project was being developed, the decision of making the *co-clustering* method the main objective of the thesis was taken. The tracking problem would need a simplified version of *co-clustering*, but a generic *co-clustering* is a tool that would give service to the Image Processing Group (GPI in Catalan) in many areas, such as the proper tracking, 3D reconstruction and coding among others.

This thesis presents the Region-based particle filter method for tracking with special attention to the *co-clustering* method, as it is the final objective of the project.

1.2. Previous work

This project is an iteration and implementation to the David Varas and Ferran Marques work in the field of Region-based particle filters. Also a lot of previous work from the scientific community in particle filters and image segmentation was used to fulfill the project.

1.3. Objectives

The main objective of this project is the programming and optimization of a generic *co-clustering* method formulated by [4] in order to give service to the Image Processing Group at UPC. The state of art of *co-clustering* is about 60 seconds for a partition to partition computation.

A secondary objective is to generate a tracking algorithm using the Region-based particle filter [3]. To achieve this, the methodology of the Region-based particle filter will be studied in order to design and implement an efficient structure for it.

Also, for a specific object of interest, the *particle support partition* will have special attention too, as it will provide information of the regions that are eligible to be part of the object via particles.

1.4. Project development

An overall learning was performed in a first stage. This learning was based on the study of different tracking methods with special focus on particle filters. The region-based particle filter was faced after the first learning step.

For a proper development of the algorithm it was divided in stages. Each stage has been developed in 3 phases: planning, programming and evaluation of the state.

1.4.1. Planning

This phase consists on reviewing the functionality of each fragment of the algorithm in an isolated way. For almost all the states a function (or part of the final program) and its associated input and output were defined. This phase was also where the programming strategy for each algorithm fragment was planned.

1.4.2. Programming

The programming phase was performed based on the planning defined in the previous section. Almost all the functions were implemented in 2 steps: basic programming and optimization.

- **Basic programming**

The basic programming, in the majority of cases, was handled in Matlab code. This decision was made due to a better display of the characteristics of each stage and to the fact that some functions were already implemented in Matlab. For these functions, this step consists on reconsidering their implementation and determine more efficient strategies for their calculation.

- **Optimization**

Once a function was programmed, an optimization step was performed. This optimization step takes in account the computational time for each function. The most time consuming functions were implemented in C/C++ using the "mex-files" Matlab technology. Using this technology, the functions have been compiled and can be used in a Matlab environment, making the computation time decrease.

1.4.3. Evaluation

An aim of this project was to get as close as possible to an on-line object tracking, so the duration of each function was taken in account in order to determine if they were as good as possible in a time-related evaluation. If the function did not match the expectations, a reformulation and a reimplementation were made for that fragment of the algorithm.

1.5. Thesis Outline

The structure of this thesis is organized as follows.

Chapter 1: An introduction to the project. Motivation, objectives and the development structure of the project is explained in this section.

Chapter 2: Color-based particle filter analysis. This chapter will explain the functionality of a standard particle filter based on color distributions used for tracking. The particle definition for this concrete tracker and the steps of particle filtering based on SIR algorithm (resampling, propagation, weighting and estimation) will be described in this section. Also each step will be illustrated with an example, to give an overall view of the object tracking based on color distributions.

Chapter 3: Definition of region-based particle filter. The way the particles are defined, the 4 modified steps of the SIR algorithm and some results are in this chapter. *Particle support partition* and *co-clustering* will have special attention for the reason that they are the core of this project. The most important variables will be explained in a table and pseudo code will be written for each step.

Chapter 4: The results of the algorithm will be displayed in this section. Taking in account that the perturbation part of the algorithm is not yet finished, this chapter will display co-clustering results between different partitions and their computation time.

Chapter 5: The budget that this project would require. The time spent on this project and some other costs will be included in this chapter.

Chapter 6: The conclusions and future development this project brings. Objectives achieved, things that could change for a better development and next steps for the Region-based particle filters will be explained in this chapter.

2. Color-based particle filter

The color-based particle filter is a tracking method that relies on matching the color histogram of the pixels inside a geometrical form that represents the object and the color histogram of the object model. In this section, the tracking method based on color filtering is explained. For this purpose, a brief explanation of each step of the algorithm based on [2] and [3] is presented.

This method is widely used because it is simple and computationally efficient. Moreover, the use of color distributions makes the algorithm robust to partial occlusions, rotation and scale variance; the target model of the particle filter is defined by the color information of the tracked object.

A visual example of an iteration of a generic particle filter is presented in Figure 2.1.

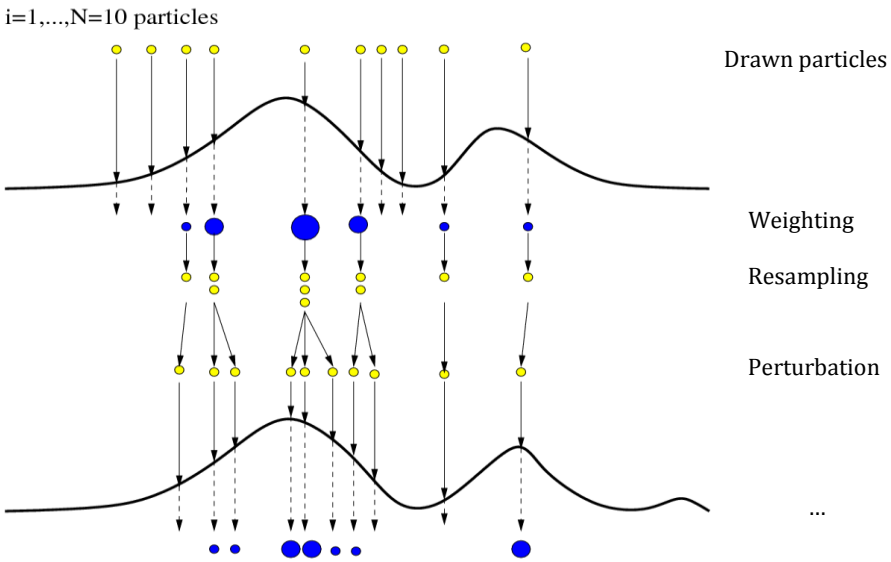

Figure 2.1.Generic particle filter scheme

2.1. Particle and model definition

In order to analyze a dynamic system, at least two models are required: First, a model describing the evolution of the state in time (the system model) and, second, a model relating the noisy measurements to the state (the measurement model).

The state $\boldsymbol{x_k}$ defines the evolution of a target (object of interest) at time $\boldsymbol{k}$ given a set of measurements $\boldsymbol{z_{1:k}}$ up to the same time instant. It is defined by the true parameters (position, velocity…) that characterize the object of interest; in other words, the state vector contains relevant information required to describe the system under investigation.

The observations $\boldsymbol{z_k}$ are what can be measured in order to do a correct estimation of the state. The measurement vector represents (noisy) observations that are related to the state vector.

The state, as it is represented as a geometrical shape, could be parameterized by a four-dimensional vector. At each time instant, the algorithm receives an image (measurement $\boldsymbol{z_k}$) and the object is tracked using a parameterization of a geometrical shape and motion cues (state $\boldsymbol{x_k}$) [3]:

Measurement: $$\boldsymbol{z_k = I_k} \quad (2.1)$$

State: $$\boldsymbol{x_k = \{x, y, w, h\}} \quad (2.2)$$

Where:

- $\boldsymbol{I_k}$ is the image or frame in time $\boldsymbol{k}$.
- ***(x,y)*** represent the geometrical form location.
- ***(w,h)*** the width and the height of the geometrical form.

2.2. Resampling

Consider $\boldsymbol{S_k} = \left\{\boldsymbol{x_k^{(1)}}, \boldsymbol{x_k^{(2)}}, \dots, \boldsymbol{x_k^{(N_s)}}\right\}$ as a set of $\boldsymbol{N_s}$ particles at time $\boldsymbol{k}$ another set of particles $\boldsymbol{S'_k} = \left\{\boldsymbol{x'^{(1)}_k}, \boldsymbol{x'^{(2)}_k}, \dots, \boldsymbol{x'^{(N_s)}_k}\right\}$ is created doing a random sampling of $\boldsymbol{S_k}$ with replacement. In this process, articles with high weights are more likely to be chosen many times.

The basic idea of resampling is to eliminate particles that have small weights and to concentrate on particles with large weights. The idea is to draw $\boldsymbol{N_s}$ samples from the current

weighted approximation; generate N_s new particles $x'^{(i)}_n$ from the set $x^{(i)}_n$ according to the importance weights $w^{(i)}_n$.

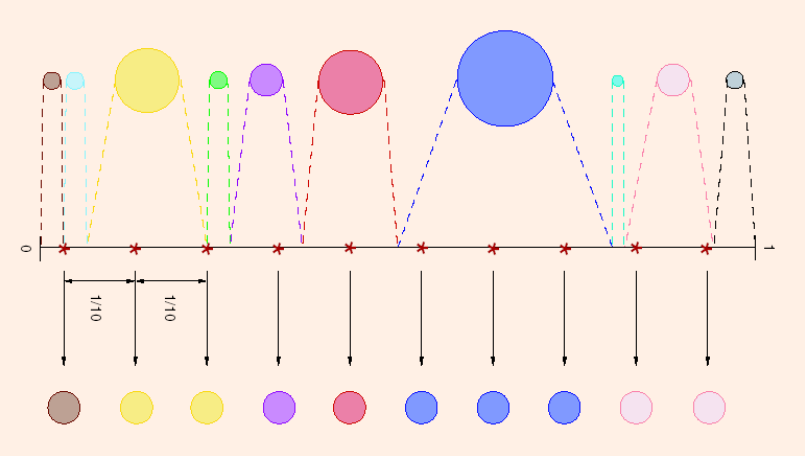

Figure 2.2.resampling step

In the figure above the resampling step can be observed. The more weight a particle has, the more probable is to be chosen many times for generating the new set of particles. This is used to avoid the degeneracy phenomenon (a particle has large weight and the others are negligible). Particles with small weights can survive the resample step, but they will be eliminated statistically after a few iterations.

2.3. Propagation and perturbation

The next section explains the propagation and perturbation of the particles. The model to follow is showed at the next figure:

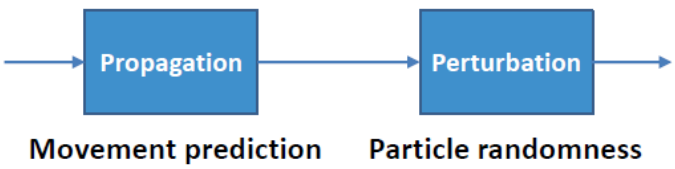

Figure 2.3. Propagation and perturbation steps

The particles from the new set of particles $\boldsymbol{S'_k}$ are propagated following a function that describes the movement of the object between consecutive instants. The sample set is propagated through the application of a dynamic model:

$$\boldsymbol{x_{k+1}^{(i)} = A x'^{(i)}_k + B v_k^{(i)}} \tag{2.3}$$

Where:

- $\boldsymbol{A}$ is a deterministic matrix. Those particle parameters contained in $\boldsymbol{x_k^{(i)}}$ that are supposed to change between consecutive images are first estimated using $\boldsymbol{A}$ in the *prediction step* for the object movement prediction.

- $\boldsymbol{B}$ is a certain variance.

- $\boldsymbol{v_k^{(i)}}$ is a random variable to estimate the trajectory. The parameters obtained multiplying $\boldsymbol{x_k^{(i)}}$ and $\boldsymbol{A}$ are slightly modified using the random variable $\boldsymbol{v_k^{(i)}}$ multiplied by a variance $\boldsymbol{B}$. This random component is added in the *perturbation step* to the samples of the set, creating $\boldsymbol{N_s}$ hypothetical states of the system [3].

2.4. Weighting

A weight $\boldsymbol{w_k^{(i)}}$ is assigned to each particle $\boldsymbol{x_k^{(i)}}$. This weight depends on the measure $\mathbf{z_k}$. In the color-based particle filter, $\boldsymbol{w_k^{(i)}}$ is based on the comparison of the color histograms of the model and each particle. The weight $\boldsymbol{w_k^{(i)}}$ for each particle can be computed through the *Bhattacharya coefficient*.

The Bhattacharya coefficient for each color distribution is calculated via:

$$\boldsymbol{\rho_k^{(i)} = \sum_{i=1}^{3 \cdot Nb} \frac{1}{3} \sqrt{\frac{h_{model}^{(i)} \cdot h_{particle_k}^{(i)}}{norm_k^{(i)}}}} \tag{2.4}$$

where $\boldsymbol{norm_k^{(i)}}$ is a normalization coefficient.

$$\boldsymbol{norm_k^{(i)} = \#pixels_{model} \cdot \#pixels^{(i)}_{particle_k}} \tag{2.5}$$

Once the Bhattacharya coefficient is calculated, the weight of each particle is computed as:

$$\boldsymbol{w_k^{(i)} = e^{-\frac{(1-\rho_k^{(i)})}{\sigma}}} \tag{2.6}$$

2.5. Estimation

Once the weights of the samples are calculated a discrete posterior density function is computed as the weight normalization; then, the mean state of the system at each iteration can be computed as:

$$\boldsymbol{\pi_k^{(i)} = \frac{w_k^{(i)}}{\sum_{i=1}^{N_s} w_k^{(i)}}} \tag{2.7}$$

$$\boldsymbol{E[S_k] = \sum_{i=1}^{N_s} \pi_k^{(i)} x_k^{(i)}} \tag{2.8}$$

Since all the samples represent the same geometrical shape, mean state is a new particle with the same shape and whose parameters are defined by the weighted mean of the parameters of the $\boldsymbol{N_s}$ particles.

- **Applied example**

For a real color-based particle example a color tracker already is used. In this case the number of particles is fixed to $N_s = 30$. Each particle is painted in yellow and the mean state is painted in green.

Figure 2.4. Tracking of an object of interest using a tracker already developed

3. Region-Based Particle filters

In contrast with a tracking algorithm, which estimates the position of a certain object of interest, a region-based particle filter is able to track and segment the tracked object at each time instant. In this section, the Region-based particle filters will be explained, with an emphasis to *co-clustering*.

As this project has a large programming part and it is pretended to be used in a near future, a summary of important variables used in *co-clustering* and their description is handled. Pseudo code will also be written in the different sections.

3.1. Particle and model definition

The first change from the color-based to the Region-based particle filter is that its particles are generated using regions from a partititon. Each particle is a union of regions from a partition of the image.

A new representation of the state and measurements for the tracking problem in terms of regions is defined:

$$\boldsymbol{x_k} = \bigcup_{r}^{n_k^o} \boldsymbol{R_k^r} \tag{3.1}$$

$$\boldsymbol{z_k} = [\boldsymbol{I_k}, \boldsymbol{P_k}] \tag{3.2}$$

Where **(3.1)** is the new state and **(3.2)** the new measurement.

- $\boldsymbol{n_k}$ is the number of regions that form the partition.
- $\boldsymbol{n_k^o}$ is the number of regions that characterize the object. $\boldsymbol{n_k^o} \leq \boldsymbol{n_k}$.
- $\boldsymbol{P_k} = \{\boldsymbol{R_k^{(1)}}, \boldsymbol{R_k^{(2)}}, \dots, \boldsymbol{R_k^{(n_k)}}\}$ is a partition of the image $\boldsymbol{I_k}$.

First of all, the over-segmentation of the frame is computed. This will be referred as a fine partition.

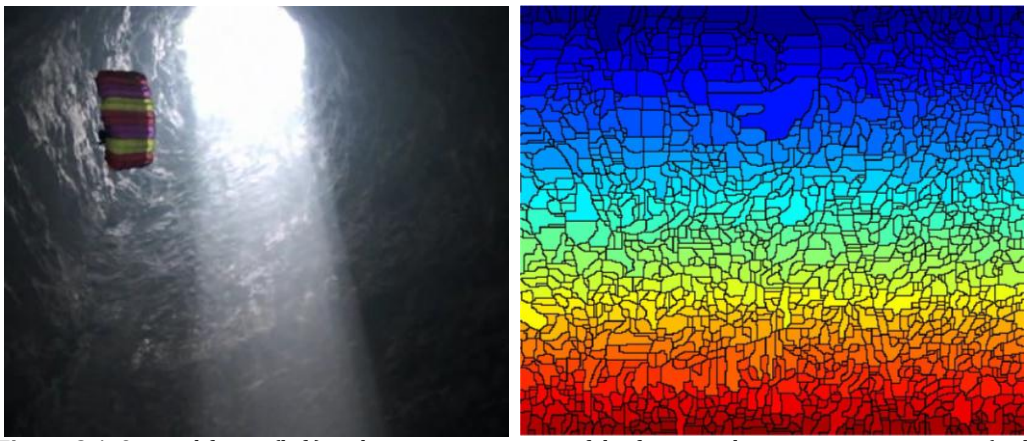

Figure 3.1. Original frame (left) and over segmentation of the frame with contour superposition (right).

For a better overview the partition and the image superposition are shown in the next figure.

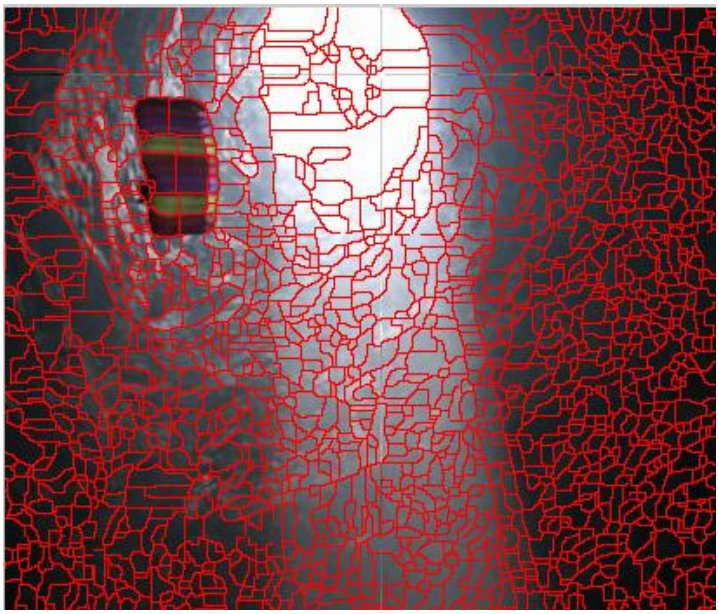

Figure 3.2. Over Segmentation of the frame.

The next step is to define the object model. This model will be used to weight the particles and to estimate the object at the next time instant ***k***.

Figure 3.3. Model of the object of interest generated via union of regions of the fine partition.

Finally, the particles will be generated. At time t_0 all the particles match the model (as the initialization of the particles are the proper union of regions from the model), the particles below are taken from time t_{k-1}, where the particles have been propagated and perturbed.

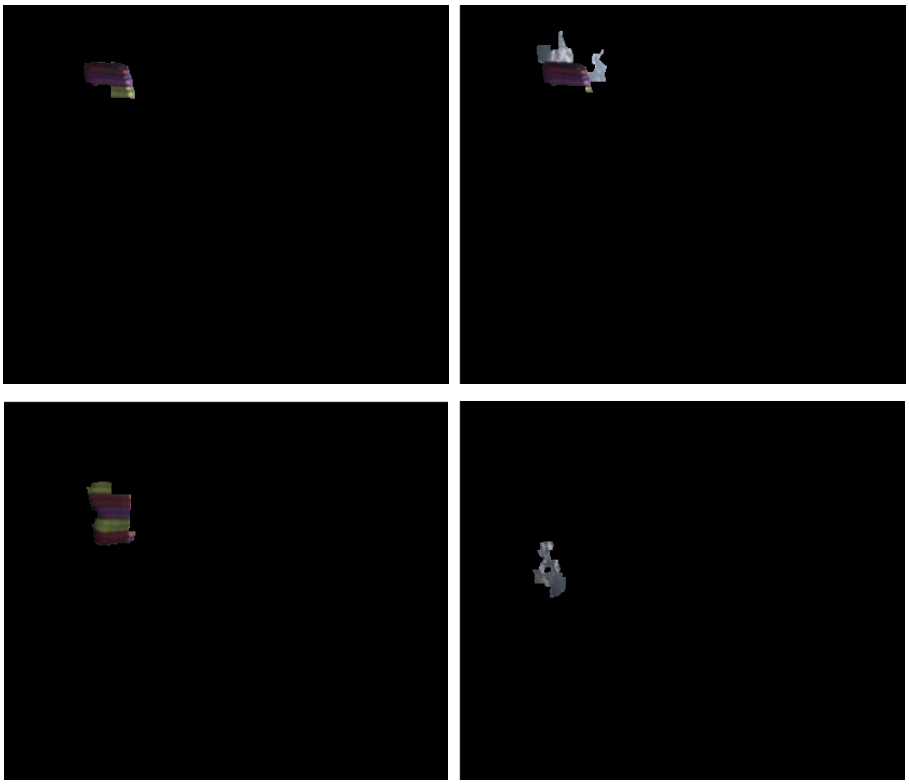

Figure 3.4. Particles (1-4) generated via union of regions of the over-segmented partition. There are N generated particles in the proper algorithm.

ALGORITHM 1:

Choose an image with a partition and a set of vectors associated. Define the model state choosing a set of regions that define the object of interest. Define the particles choosing a set of regions that define the object of interest. In a real tracking case the particles would be fitted to the model; the regions associated with each particle would be the regions of the proper model.

As in the color-based approach, the structure of the region-based particle filter can be divided in four steps: propagation, evaluation, estimation and resampling.

3.2. Resampling

The resampling only considers the support points of the tracked pdf represented by the particles and their associated weights. These weights have been previously computed in the Evaluation step. Thus, the resampling algorithm described for the color-based particle filter in Section **2** (*Color-based particle filter*) is applied at this point [3]. This step is performed if $N_{eff} < N_{eff\ threshold}$ for avoiding the *degeneracy* problem.

3.3. Propagation and perturbation

The propagation step consists on creating a new set of particles in time ***k*** based on the particles in time ***k-1***. In this project, where the particles are unions of regions, this step will try to match the particles of time ***k-1*** with the particles of time ***k*** and perturb the particles of time ***k-1*** around the particles of time ***k***.

Propagation will be calculated in two steps: prediction and perturbation.

Figure 3.5. Labeled elements of the fine partition's contours of t_{k-1} (right) and t_k (left).

A labeled element is a contour element. From the labeled elements the propagation will be performed for the particles from t_{k-1} to t_k. This propagation will use different data (i.e. angles of the contours, color histogram around a labeled element…) in order to perform properly. The number of labeled elements of the partition in t_{k-1} and t_k is very likely to change (apparition of new objects, occlusions, movement of the elements of the image generally).

The labeled elements will be handled in a larger space than the original image and partition in order to define the contours for each region. This new space will be defined as:

- I_k is a matrix of (M_I x N_I) dimensions.
- $Labeled\_elements_k$ is a matrix of ($2M_I+1$ x $2N_I+1$) dimensions.
- Each labeled element has an angle associated. It is considered to be vertical or horizontal depending on its place at the new $labeled\_elements_k$ space. If it is placed on an odd column it will mean that the contour is vertical; if it is placed on an even column it will mean that the contour is horizontal. A graphical example is shown below.

Let's suppose a fine partition

In the image $\boldsymbol{I_k}$ space

$\boldsymbol{P_k} = \{\boldsymbol{R_k^1}, \boldsymbol{R_k^2}, \boldsymbol{R_k^3}\}$

$M_I = 3$;

$N_I = 3$;

1	1	2
1	1	2
3	3	2

We transform it to the

$\boldsymbol{Labeled\_elements_k}$ space

$M_L = 7$;

$N_L = 7$;

0	0	0	0	0	0	0
0	1	0	1	0	2	0
0	0	0	0	0	0	0
0	1	0	1	0	2	0
0	0	0	0	0	0	0
0	3	0	3	0	2	0
0	0	0	0	0	0	0

The labeled elements will be computed as mentioned

even: horizontal

odd: vertical

0	0	0	0	0	0	0
0	1	0	1	|	2	0
0	0	0	0	0	0	0
0	1	0	1	|	2	0
0	—	0	—	0	0	0
0	3	0	3	|	2	0
0	0	0	0	0	0	0

The labeled elements for this partition will be:

They are labeled from left to right and top to bottom.

0	0	0	0	0	0	0
0	0	0	0	**1**	0	0
0	0	0	0	0	0	0
0	0	0	0	**2**	0	0
0	**3**	0	**4**	0	0	0
0	0	0	0	**5**	0	0
0	0	0	0	0	0	0

The contours (labeled elements) will have an angle associated. Each angle exists on a range of (0,180]. They will be computed as 8 possible angles spaced π/8, 22.5 in degrees: 22.5, 45, 67.5, 90, 112.5, 135, 157.5 and 180.

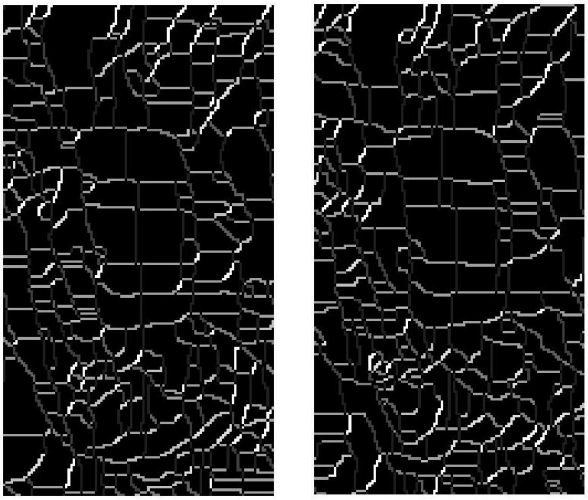

Figure 3.6. Angles associated to the labeled elements represented [0,255] of t_{k-1} (right) and t_k (left).

As the objective of this project is the optimization of the algorithm, a bounding box over the particles will be used for the propagation step, as all the other information would be redundant.

As the matching distance of the labeled elements used in *co-clustering* is 10 positions of the ***labeled_elements*** space, a box surrounding the particle support partition, with its width and height based on the distance to search multiplied by 2 ($2 \cdot D_w$ pixels), will be defined; this way it's assured that no useful labeled element is left outside the calculation.

Considering the particles (1-4) from the Section **3.2** (*Particle and model definition*) and the support partition of from the *Particle support partition*, the bounding box considered for the propagation step will be:

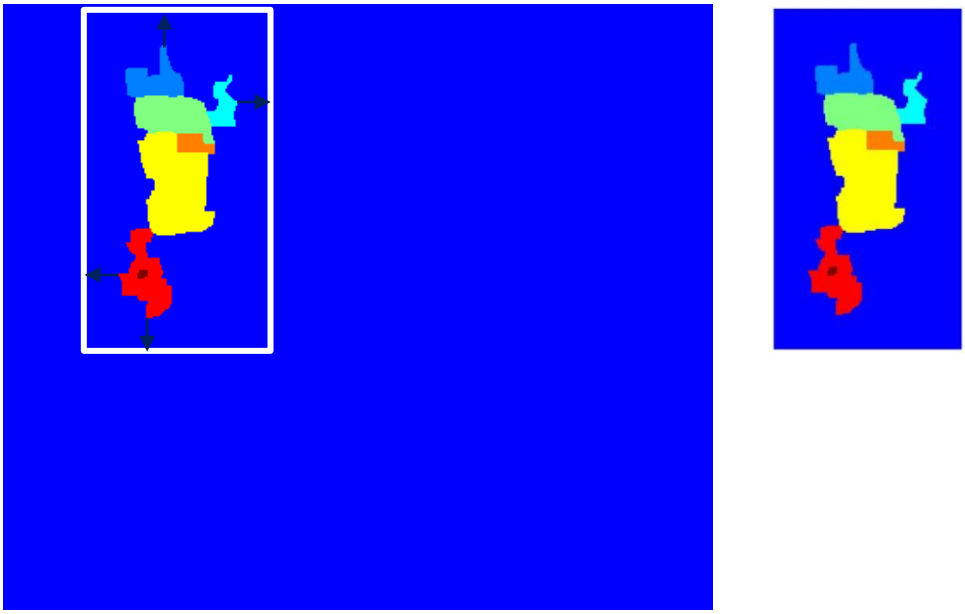

Figure 3.7. Bounding box associated with the particle support partition of t_k.

By doing this it's assured that, if the four corner regions aren't represented in the particles, the computational cost will be less than the computation cost of the whole image.

In case that those regions were represented by the particles and they were located at the 4 corners of the image I_k, the propagation step would be the same as computing the whole image (as the bounding box would be the whole image).

3.3.1. Prediction step

Considering a pair of images and the associated partitions in ***k-1*** and ***k***, the prediction step tries to ensure a minimum quality of the particles estimation. This step will create a set of new particles optimizing a certain score function over the representation of the object in two partitions between consecutive time instants (***k-1*** and ***k***). In order to compute a single operation for all particles an adapted co-clustering for tracking of both partitions is performed [3].

The two most important steps of the prediction step are the *particle support partition* definition and the *co-clustering* of both partitions.

3.3.1.1. Particle support partition

An extra partition will be generated from the particles information in order to jointly propagate them from two time instants. This partition is created taking in account the intersections between all particles. It is used to do the proper propagation of the particles as the importance is not to match each region of t_{k-1} with the regions of time t_k but to match the regions of interest (regions used by the particles). Moreover, the propagated regions should allow the algorithm to reconstruct the particles at the next time instant.

Using the information contained in the particle support partition, at time k the corresponding set of particles will be able to be generated, as the particles propagated from the previous time instant.

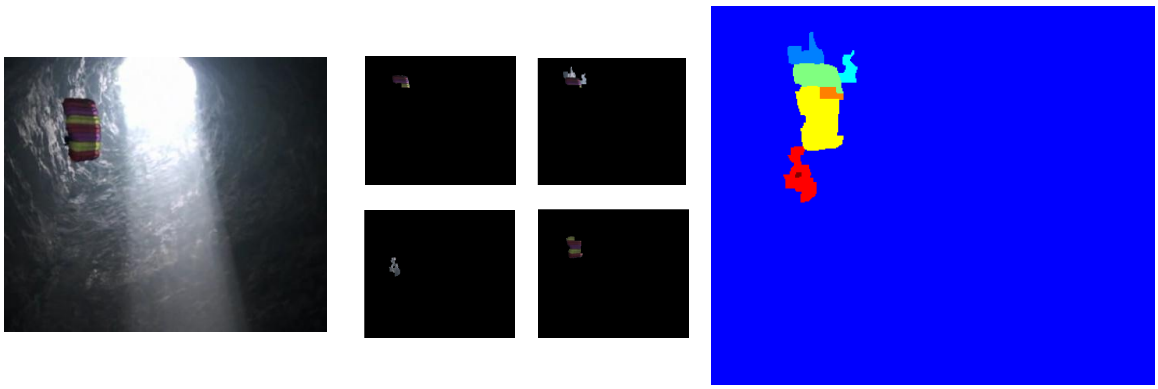

Figure 3.8. Particle support partition of the defined particles (1-4).

Particles	IDs fine partition	IDs Support Partition
1	[217,218,300,301,320]	[4,6]
2	[109,161,170,171,209,217,218,301]	[2,3,4]
3	[296,335,342,419,420,450,464]	5
4	[517,547,599,600,604,631,632,649,662,663,685,686]	7

Table 3.1. Relabeled regions of the particle. The relabeling is used in order to not to have different identifiers from one partition to the other, as the partitions from t_{k-1} and t_k may have different regions.

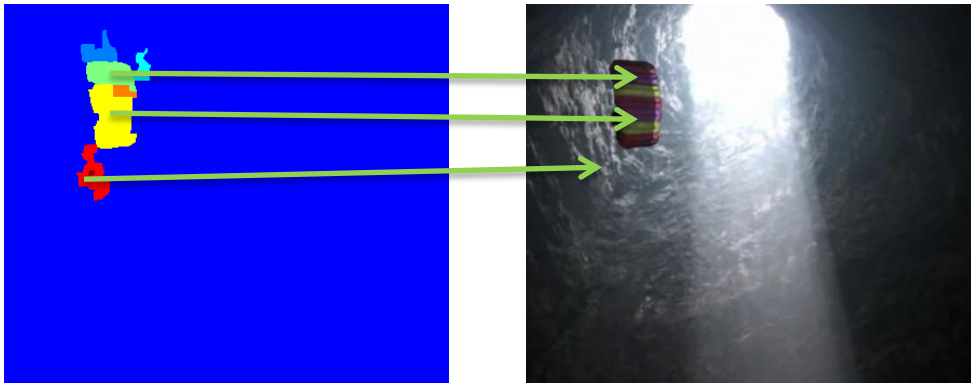

Figure 3.9. Expected primal propagation.

Using the particle support partition all particles can be jointly propagated with a single optimization process. Regions from the fine partition of the next time instant are labeled with labels of regions from the previous particle support partition [3].

ALGORITHM 2:

Generate a matrix of relations *relation_matrix* between regions and particles (regions x particles).

For each particle:

    Fill the *relation_matrix* with its information.

end

For each region:

    Recursive function that returns the information of each adjacent regions that correspond to the same region of the support partition.

end

Return the particle support partition and the particles with the relabeled regions.

3.3.1.2. Co-clustering

The contour-based joint clustering (co-clustering) of multiple segmentations is based on: given two or more closely-related images, such as nearby frames in a video sequence,

generate a joint segmentation of the images [4]. With this technique the connection between the partition in time ***k-1*** and the partition in time ***k*** may be determined.

The aim is to generate clusters defined by regions of the fine partition. This method can be used to determine the clusters of regions of the partition in time ***k*** that fit to the support partition of time ***k-1*** without merging regions from the particle support partition. The definition of the used *co-clustering* method can be found at [3].

Let us consider a pair of images $\{\boldsymbol{I_t}\}_{\boldsymbol{j=k-1,k}}$ and their associated partitions $\{\boldsymbol{P_t}\}_{\boldsymbol{j=k-1,k}}$. Each of these partitions is formed by a group of n_j regions $\boldsymbol{P_t} = \{\boldsymbol{R_t^1}, \boldsymbol{R_t^2}, \dots, \boldsymbol{R_t^{n_j}}\}$. A co-clustering between partitions is defined by $\boldsymbol{X} \in \{\mathbf{1}, \mathbf{0}\}^{\boldsymbol{nxc}}$, where $\boldsymbol{n} = \boldsymbol{n_{k-1}} + \boldsymbol{n_k}$ and $\mathbf{c}$ is the number of clusters. Each column $\boldsymbol{x_l}$ with $\boldsymbol{l} \in \{\mathbf{1}, \dots, \boldsymbol{c}\}$ corresponds to a single cluster. Regions from partitions are assigned to only one cluster if matrix $\boldsymbol{X}$ is constrained to have unitary rows. The score associated with $\boldsymbol{X}$ is computed as:

$$\boldsymbol{tr}\left(\boldsymbol{X^T Q X}\right) = \sum_{\boldsymbol{l=1}}^{\boldsymbol{c}} \boldsymbol{x_l^T Q x_l} \tag{3.3}$$

where $\boldsymbol{Q} \in \mathbb{C}^{\boldsymbol{nxn}}$ is a matrix that measures affinities between regions.

Matrix $\boldsymbol{Q}$ is computed with similarities between pairs of regions from the same partition (Intra image similarities) and from different partitions (Inter image similarities).

Intra image similarity is proportional to the number of contour elements that share both regions and their color similarity.

Inter image similarities are captured comparing the HOG descriptor of their contour elements that are closer than 20 pixels. Then, co-clustering of both partitions becomes an optimization problem:

$$\max_{\boldsymbol{X}} \boldsymbol{tr}\left(\boldsymbol{X^T Q X}\right) \tag{3.4}$$

$$\boldsymbol{X_{y,z}} \in \{\mathbf{0}, \mathbf{1}\}\ \forall \boldsymbol{i}, \boldsymbol{j} \qquad \sum_{\boldsymbol{j}} \boldsymbol{X_{y,z}} = \mathbf{1} \forall \boldsymbol{i} \tag{3.5}$$

Further approaches make use of distances defined over cliques in a region adjacency graph. Considering this conditions, optimization can be stated as:

$$\min_{D} \sum_{y,z} q_{y,z} \cdot d_{y,z} \tag{3.5}$$

$$0 \leq d_{y,z} \leq 1 \quad d_{y,y} = 0 \forall y \quad d_{y,z} = d_{y,z} \; \forall y, z \tag{3.6}$$

$$d_{y,z} \leq d_{y,k} + d_{k,z} \quad \forall e_{y,z}, e_{y,k}, e_{k,z} \in G \tag{3.7}$$

where:

- G is the previous adjacency graph
- $d_{y,z} \in \{0, 1\}$ is the distance between regions y and z. Regions that belong to the same cluster have distances equal to 0.
- $q_{y,z}$ is the similarity between regions y and z.

Extending this *co-clustering* to the tracking problem, two types of similarities are settled: Intra and Inter.

Intra: similarity between regions from the same partition. The similarity $q_{y,z}$ for the same partition is computed as:

$$q_{y,z} = \lambda \cdot v_{yz} \cdot u_{yz} \tag{3.8}$$

where:

- λ is the fusion factor (the higher the λ the less regions).
- v_{wz} is the number of contour elements between regions y and z.

- u_{wz} is the color similarity between regions y and z.

A matrix Q_i or Q_j (depending on which partition i or j) stores the $q_{y,z}$ elements for each region.

Inter: similarity between regions from different partitions. For each contour element, a descriptor w_e and the direction of the vector normal to the contour b_e are computed. The similarity between descriptors of elements k *and* l is represented by w_{kl}.

$$q_{yz} = \sum_{k,l} w_{kl} \cdot b_k \cdot b_l \tag{3.9}$$

where:

- w_{kl} is the similarity between element descriptors.

- b_{kl} is the normal vector at the contour element.

A matrix W will contain the similarities between labeled elements from partitions i and j. A matrix B_i or B_j will contain the information with the vectors normal to each contour element pointing outside each region.

Then, the region containing similarities from different partitions can be computed as:

$$Q_{ij} = B_i' \cdot W \cdot B_j \tag{3.10}$$

Finally, the matrix Q will be generated as:

$$Q = \begin{bmatrix} \lambda \cdot Q\_i & Q\_ij \\ Q\_ij & \lambda \cdot Q\_j \end{bmatrix} \tag{3.11}$$

For a better overview a list of important variables and definitions used for the *co-clustering* implementation is handled at table 3.2.

Variables and definitions

Variable name	Description	Dimension
I_i, I_j	Frames in t_{k-1} and t_k respectively	(M_I x N_I)
seg_i, seg_j	Segmentations associated with each image (***I_i***, ***I_j***).	(M_I x N_I)
vectors_i, vectors_j	Angles associated with each labeled element. At the beginning they have a value from [1,8] depending on the direction; later on, they will be transformed to proper angles (degrees).	($2\cdot M_I+1$ x $2\cdot N_I+1$)
angles_i, angles_j	Angles computed from ***vectors_i, vectors_j*** (degrees).	($2\cdot M_I+1$ x $2\cdot N_I+1$)
λ	Fusion factor	Scalar
D_w	Maximum distance to look at when the algorithm is searching for surrounding labeled elements. Also is the distance that will determine the bounding box for the particles.	Scalar
labeled_elements_i, labeled_elements_j	Contour elements calculated from the fine partition.	($2\cdot M_I+1$ x $2\cdot N_I+1$)
RAG_i, RAG_j,	Compute adjacency between regions of the same partition. This adjacency is used to reduce the number of variables in the process.	(#_regions x #_regions)
RAG_ij	Compute adjacency between partitions. This adjacency is used to reduce the number of variables in the process.	(#_regions_i x #_regions_j)
B_i, B_j	B matrices contain the vectors normal to the contour at each contour element (labeled element) pointing outside the region.	(#_labeled_elements x #_regions) ***Sparse representation***

Variable name	Description	Dimension
d_i, d_j	Descriptors computed for each labeled element	((rgb_bins + grad_bins) x #_labeled_elements
W	Contains the similarities between labeled elements from ***i*** and ***j***.	(#_labeled_elements_i x #_labeled_elements_j) ***Sparse representation***
Q_i, Q_j	Contains the similarities between regions of the same partition. (Intra)	(#_regions x #_regions) ***Sparse representation***
Q_ij	Contains the similarities between regions of different partitions. (Inter)	(#_regions_i x #_regions_j) ***Sparse representation***
Q	Q matrix contains similarities between regions from both partitions (Intra and Inter).	((#_regions_i +#_regions_j)) x ((#_regions_i +#_regions_j)) ***Sparse representation***
result_seg_i, result_seg_j	Resulting partitions from the co-clustering (relabeled).	(M_I x N_I)

Table 3.2. Table of variables

The idea is to connect the contour elements from a partition to another, in order to determine the approximate positions of certain indexed regions of time ***k-1*** at time ***k***. At the co-clustering step the match between the fine partition in ***k*** and the support partition in ***k-1*** will be computed.

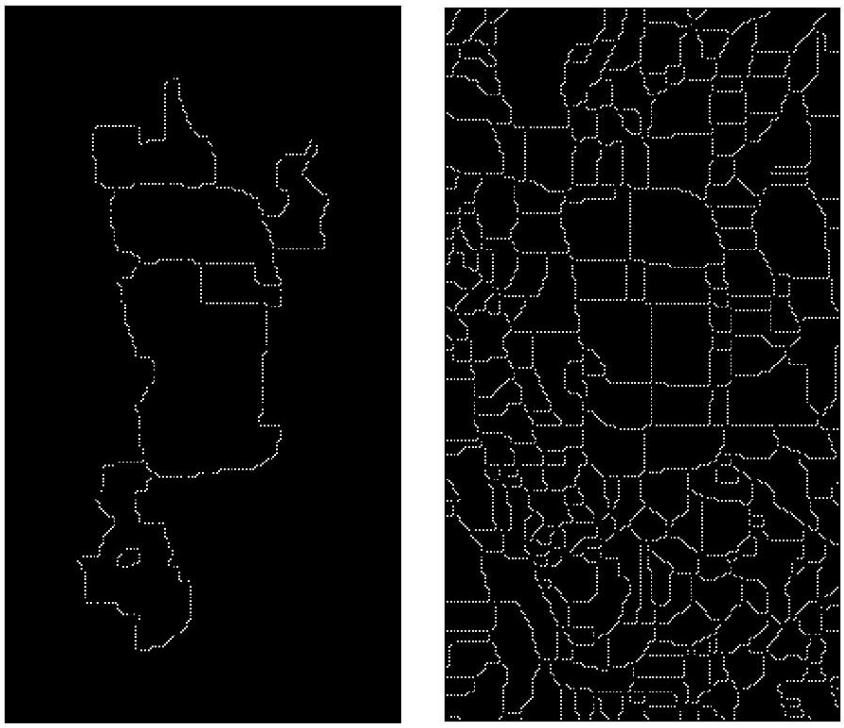

Figure 3.10. Contour elements of the support partition in t_{k-1} (left) and the next frame partition t_k (right).

The connection between regions of the partition in ***k-1*** and the partition in ***k*** are computed through their adjacency. For this reason, the matrices ***RAG_i***, ***RAG_j*** (for the internal adjacency of regions or Intra) and ***RAG_ij*** (for the adjacency between two different partitions or Inter) are be calculated. These matrices are for forward calculations.

ALGORITHM 3:

```
For each labeled element
        Compare if there are two different regions at its adjacent values.
        If there are two different regions
                Those regions are adjacent
        Else
                Those regions are not adjacent
end
```

ALGORITHM 4:

```
For each element of seg_i
        Determine inter adjacency observing the same position at the
        matrix seg_j. The region of seg_i and seg_j at that position will be
        adjacents in an Inter way.
end
```

Three measures determine the similarity between two regions: color, texture and distance. Each measure is taken for each labeled element. For every labeled element in ***k*** the similarity between it and the surrounding contour elements in ***k-1*** is computed.

The descriptors that determine the similarity between contour elements are color and texture. The distance measure will not be considered as a differential factor between labeled elements because the distance of the window (D_w) is small enough to determine the closer labeled elements.

The information of the distance is exploited computing similarities only between contour elements inside a distance window (D_w) of each contour element.

- **Color**

The color information is obtained from a local histogram of pixels from the image in a pixel neighborhood for each contour element. The color information will be the most relevant to determine which labeled element in ***k-1*** corresponds to which labeled element in ***k***. A multiplying factor of 15 is applied to each color measure in order to emphasize its relevance.

- **Texture**

The angle of the vectors normal to the contour at each contour element is used as a parameter. A local histogram of angles (HOG) in a labeled elements neighborhood is calculated. As the texture measure is less relevant a multiplying factor of 3 is applied to each gradient measure.

An example of the normalized histograms of color (RGB) and texture (Grad) of a labeled element from the particle support partition is shown in table 3.3.

R	G	B	Grad
0,04	0,40	0,26	0
0,54	0,22	0,32	0
0,18	0,14	0,18	0
0,24	0,24	0,18	0
0	0	0,06	0
0	0	0	0
0	0	0	1
0	0	0	0

Table 3.3. Sample from the variable d_i at time ***k-1*** for a labeled element

ALGORITHM 5:

```
Transform the frame to the labeled elements space

For each labeled element:
    For a window of radius Dw
        Store in an array the RGB value of each pixel of the frame
        that surrounds the labeled element.
        Store in an array the angle value of each labeled element
        that surrounds the labeled element.
    end
    Compute the color and gradient histogram.
end
```

The similarities of the regions from the same partition will be placed in the matrices ***Q_i*** and ***Q_j*** for each partition ***seg_i*** and ***seg_j***. The similarities between regions from different partitions will be placed in a matrix ***Q_ij***. Once both Intra and Inter similarities are computed, they will be handled in matrix ***Q***. This matrix is used for the calculation of the cost function.

The Intra matrices ***Q_i*** and ***Q_j*** are computed taking in account the similarities between the contours and the similarities between the color of adjacent regions within each partition.

ALGORITHM 6:

```
Generate a matrix where to put all the color histogram values inside a matrix H.

For each region
        Compute the color similarities between each region using H
end
For each labeled element
        Calculate the number of contour elements for each pair of regions
end

Calculate Q_i or Q_j
```

The inter matrix ***Q_ij*** is computed taking into account the normal vector of each contour element stored in ***B*** and its similarities between the element descriptors based on ***d_i*** and ***d_j***; these similarities are handled in the ***W*** matrix.

The matrix that contains the normal vector of each labeled element pointing outside the region will be designed as the matrix ***B***. Each region will have normal vectors for each labeled element which belongs to the contour of that region.

ALGORITHM 7:

```
For each labeled element
        Compute the normal vector. Its sign will be calculated to aim
        outside the region. Each normal vector is associated to 2 regions.
end
```

The matrix containing the similarities between the element descriptors is called ***W***. It will be generated with the information from ***d_i*** and ***d_j***. Each labeled element from ***labeled_elements_j*** will be matched with the labeled elements from ***labeled_elements_i*** inside a window D_w in order to determine its similarity.

Figure 3.11. Window taken for each labeled element. The distance of the window (D_w) will be 10 positions of the ***labeled_elements*** space.

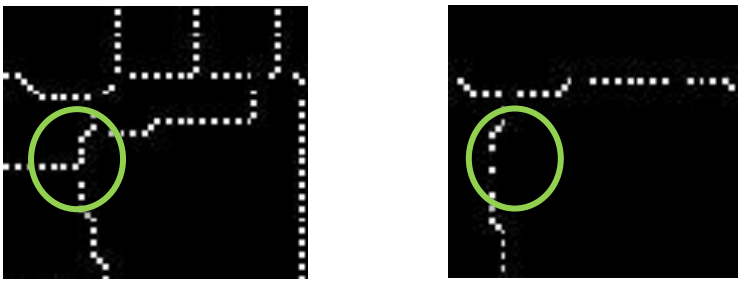

Figure 3.12. Window taken for a labeled element at time t_k (left) and matched with the labels that are at a certain distance from it.

ALGORITHM 8:

```
For each labeled_elements_j
        For each labeled_elements_i in a distance <= Dw

                Compute similarities between element descriptors.

        end
end
```

The information that has been generated to obtain the $\boldsymbol{Q}$ matrix is used for the constraints computation, the creation of cost function and the optimization process; thus, no additional computation is needed to obtain the co-clustering result. Finally, using the co-clustering result, the resulting partitions will be generated.

The result of the co-clustering for the example particle support partition defined in *Particle support partition* is:

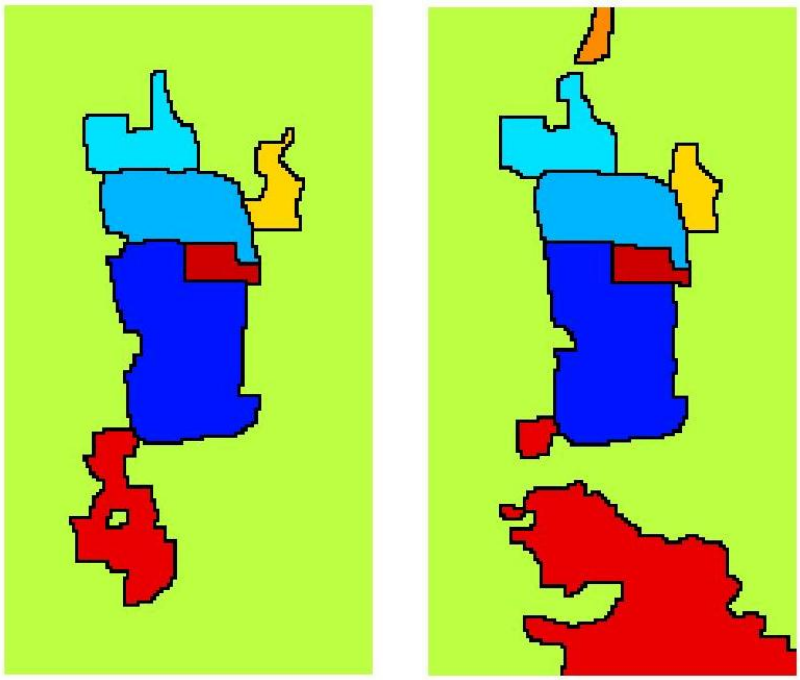

Figure 3.13. Co-clustering result for the particles (1-4).

The global co-clustering algorithm is presented in ALGORITHM 9:

ALGORITHM 9:

Calculate the labeled_elements_i, labeled_elements_j and the angles_i, angles_j for each partition.

Compute RAG_i, RAG_j and RAG_ij

Compute B_i and B_j

Compute descriptors d_i and d_j

Compute W matrix

Compute Q matrix

Optimize the process:

compute contraints
create cost function
optimize

Generate the clustered partitions

- **Results with particles fitting the model**

The results of a sequence of images with the particles fitted to the model are shown in figure 3.14. The apparition of undesired regions is provoked by the proper co-clustering, who finds contours in ***k*** that fit to the contours in ***k-1***. Once the first region outside the model appears, the degeneracy becomes intractable using co-clustering as it does not eliminate regions, it can only search for the better contour fitting.

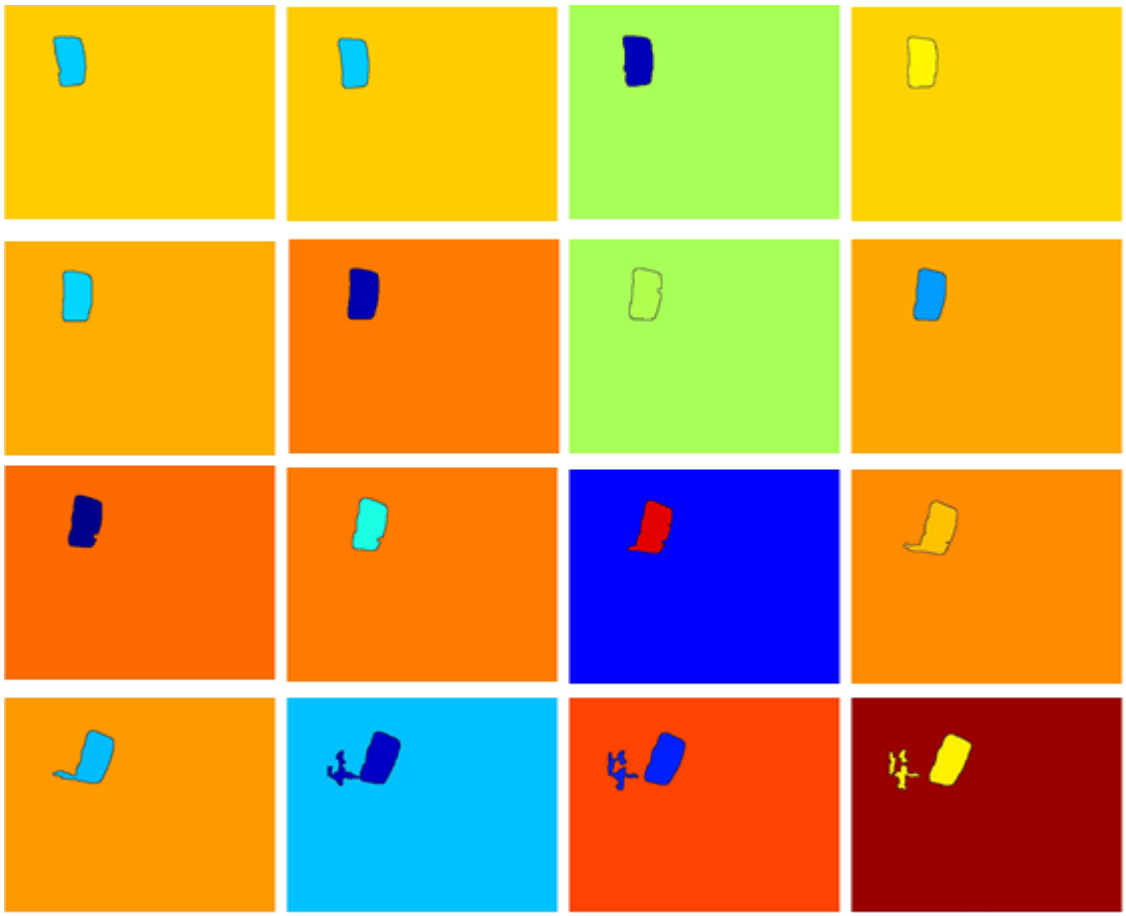

Figure 3.14. Co-clustering result for a sequence of frames where the particles fitted the model in t_0.

To mitigate this effect, the perturbation step is in charge of adding or subtracting regions from the propagated particles, in order to keep the correct object estimation.

3.3.2. Perturbation

For each particle, ***N*** changes are randomly proposed separately. These changes consist on adding/removing regions that belong to the particle or its neighborhood. Then, a

greedy algorithm is proposed in which those changes that improve the similarity measure (Bhattacharya) with the model are stored and combined to form the final particle. [3]

The main idea is handled in two steps. First of all, the distance from the regions inside the bounding box that are not part of the particle to the contour elements of the particle is computed. In order to optimize this process, this distance is calculated only with the labeled elements that form the contour of the regions of the particle that are in contact with regions not belonging to the particle; this information is handled by ***B***.

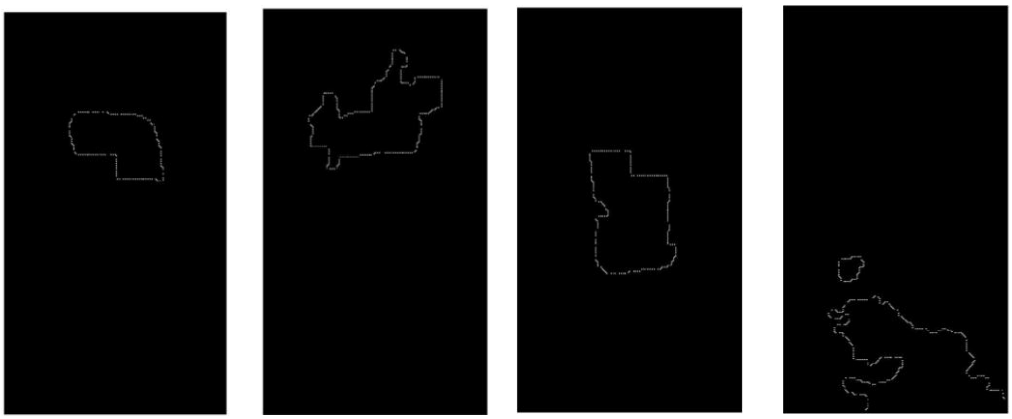

Figure 3.15. Labeled elements of the particles (1-4) to compute distance with all the regions outside the particle.

The distance for each pixel from the regions outside the particle to the closest labeled element of the particle is calculated. Then, for each region, its global distance $\boldsymbol{d}_{\boldsymbol{r-l}}^{(r)}$ (*region to labeled element* for each region) is obtained. This distance is calculated taking in account the distance from every pixel inside each region to the labeled elements of the particle $\boldsymbol{d}_{\boldsymbol{p-l}}^{(r)}$ and taking the minimum value, as it will be the closest labeled element of the particle to that pixel. Then, the global distance between each region and the labeled elements of the particle is calculated:

$$\boldsymbol{d}_{\boldsymbol{r-l}}^{(r)} = \sum_{\boldsymbol{i=1}}^{\#\boldsymbol{pixels}^{(r)}} \frac{\mathbf{min}(\boldsymbol{d}_{\boldsymbol{p(i)-l}}^{(r)})}{\#\boldsymbol{pixels}^{(r)}} \tag{3.12}$$

The computation for the regions closer than a certain threshold to the particle will be granted.

Once the distance of each region to each labeled element of the particle is computed, the regions closer than a certain threshold will be added one by one. If the weight of the particle $\boldsymbol{i}$ increase that region identifier is stored into an array; if it makes weight decrease, it will do nothing. Once the array is filled, $\boldsymbol{N}$ random changes will be proposed for each particle.

The next step is based on substracting regions from the particle. If the weight increases when substracting a region, its identifier will be stored into an array; this change may be considered later, when the random election of which regions to add from the bounding box or remove from the particle $\boldsymbol{x_k^{(i)}}$ takes place.

ALGORITHM 10:

For each pixel:

Calculate minimum distance to the particle surrounding labeled elements.

Calculate distance $d^{(r)}_{r\text{-}l}$ for the region where that pixel is settled.

end

For each particle:

For each region with $d^{(r)}_{r\text{-}l} < d_{threshold}$ outside the particle calculate if $w_{new} < w_{old}$ if the region to the particle is added

If($w_{new} < w_{old}$)

Put the region identifier to an array.

Else

Do nothing.

For each region of the particle calculate if $w_{new} < w_{old}$ if the region from the particle is removed

If($w_{new} < w_{old}$)

Put the region identifier to an array.

Else

Do nothing.

end

The perturbation step using the *greedy algorithm* gives good results for the regions to add to the particle and some problems for the regions to remove. This might be due to a bad election of the weighting (using *Bhattacharya*) as this coefficient does not lead to a good perceptual performance or a bad implementation of the method. For further research this topic will be studied.

3.4. Evaluation

Each particle will be weighted according to their *Bhattacharya distance* defined in Section **2.4** (*Weighting*). This is a measure for calculating the similarity between distributions. In this project, those distributions are color histograms. The color for each channel is distributed in 10 bins. The frame will be transformed from the RGB space to HSV for less luminance variation.

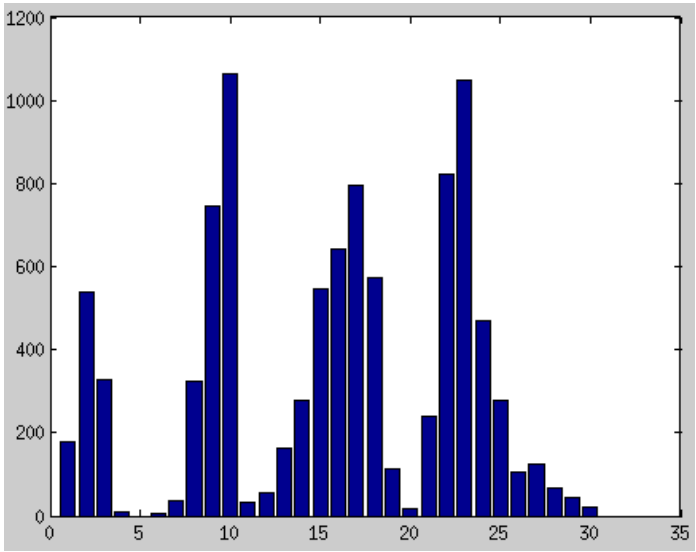

Figure 3.16. Model's color histogram. The 3 channels are concatenated to a single histogram.

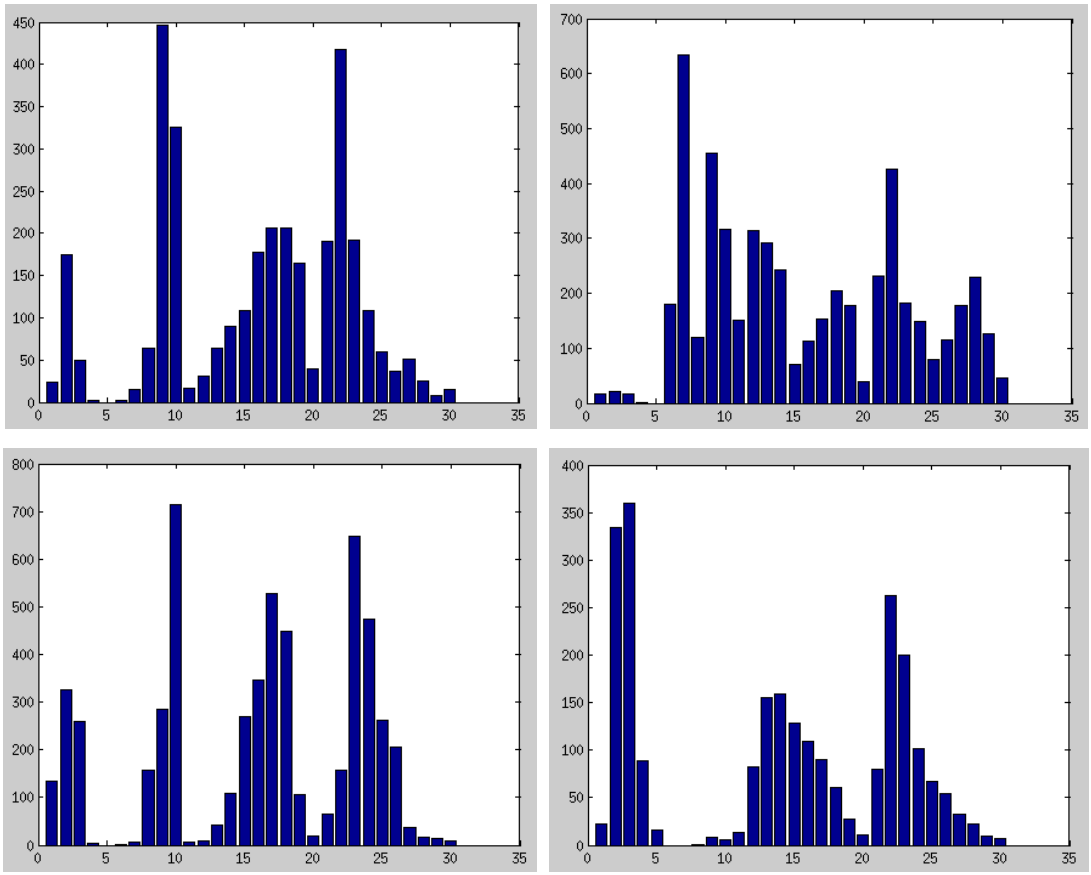

Figure 3.17. Particles color histogram. Ordered from left to right and from the top to the bottom (1-4)

particles	ρ	weight	Normalized weight
Particle 1	0.9691	0.7342	0.3818
Particle 2	0.8332	0.1887	0.0981
Particle 3	0.9756	0.7837	0.4075
Particle 4	0.8470	0.2166	0.1126

Table 3.4. Bhattacharya coefficient and weight of the proposed particles for the simulation. In a real tracking problem the order of 60 to 100 particles is used.

ALGORITHM 11:

Transform the image to HSV space.

For each particle:

Calculate the weight comparing the color histogram of the model with the particle.

Assign a weight to each particle.

end

3.5. Estimation

Once the comparisons with the model are performed, an estimation of the object is given by the particles and their associated weights. The average mask is given by:

$$A_k^M = \sum_{i=1}^{N_s} w_k^{(i)} \cdot M_k^{(i)} \tag{3.13}$$

Where:

- A_k^M is the average mask of the particles at time k.
- $w_k^{(i)}$ is the weight of particle i at time k.
- $M_k^{(i)}$ is the mask associated to the particle i at time k.
- N_s is the number of samples or particles.

Figure 3.18. Object in a weighted binary mask of the proposed particles (1-4).

ALGORITHM 12:

Generate a new partition initialized to background

For each particle:

Generate a binary mask.

Weight and normalize the mask.

Add the mask from each particle to the new partition. This new partition will have the estimation of the object (all non-object regions will be background).

end

For the estimation of the object an assumption is done. If the weighted binary mask in a certain position is less than a threshold, that position will be considered background. This way, the object estimation is:

Figure 3.19. Object binary mask with threshold applied.

Figure 3.20. Object color estimation for the proposed particles (1-4).

In figure 3.20 the reconstruction of the object using particles from (1-4) can be observed. The final shape of the object is no longer a geometrical form.

ALGORITHM 13:

Generate a new RGB image identical to the frame of which the object of interest wants to be estimated.

For each value of the mask:

If the value is less than a certain threshold put it to 0. If it equals or is above the threshold, put the value to 1.

Multiply the value with the new RGB image.

end

The global algorithm of the Region-based particle filters can be observed at ALGORITHM 14.

ALGORITHM 14:

Load frame 0, associated UCM and associated vectors calculated in an anterior step. Generate partition.

Generate model and particles.

Calculate the particle support partition and the relabelled particles for the particle support partition.

Determine a certain λ.

For frame 0 to frame N

- Load frame i+1, associated UCM and associated vectors. Generate partition. Frame i will be the support partition passed as argument.
- Create a bounding box defined by the particle support partition.
- Compute co-clustering of frame i and frame i+1.
- Compute greedy algorithm. Randomly choose the regions that increase the weight.
- Estimate the object.
- Resample particles depending on their weight.
- Calculate the particle support partition of the particles associated to frame i+1. Pass this support partition as argument for the next iteration.

end

4. Results

The results given by our algorithm will be tested with the dataset "Video Occlusion/Object Boundary Detection Dataset"[5], which includes 30 low-movement short sequences. For each sequence, the annotation of a single frame (reference frame) is provided. An object is annotated at the reference frame. This object will be used to calculate the Jaccard index between each sequence frame.

E0 consists on calculating the Jaccard index from the leaves partition against the object mask without using the *co-clustering* technique. This method assures the best Jaccard index as it uses all the regions from the reference partition.

E1 contains the previous work in *co-clustering*. In this experiment, the segmentation of the reference frame is obtained as the one that maximizes the Jaccard index. A *co-clustering* between the reference and each frame is computed considering different $\boldsymbol{\lambda}$ values in the range of [1,10].

E2 contains the results from the same experiment of **E1** with the *co-clustering* method of this project.

	Bench	Car	Chair1	Cmu sign	Coffee stuff	Couch color	Couch corner	Fencepost	Hand2	Hand3	Intrepid	Intrepid corner	Intrepid corner2	Linus1	Mugs	Mugs2	Post	Rocking horse	Squirrel2	Squirrel3	Squirrel4	Staplers	Trash	Trash can	Tree	Walking legs	Wooden man	Zoe1	Zoe2	Zoe3
E0	97	96	96	99	96	94	99	93	96	97	98	99	99	93	94	98	96	98	93	92	91	91	96	99	97	92	93	96	96	96
E1	96	90	95	99	96	90	98	90	96	92	96	90	85	90	93	98	96	96	77	74	74	88	93	95	97	92	87	95	92	87
E2	96	90	95	99	95	90	98	90	96	91	96	90	85	90	93	98	96	96	77	73	74	88	93	95	97	92	87	95	91	87

Table 4.1. Results of the *co-clustering* experiments for all the sequences of the database [5].

As it can be observed at table 4.1, the results in terms of maximized Jaccard index between experiments **E1** and **E2** are very similar. The observed differences are due to changes in the algorithm methodology. Passing from Matlab to C/C++ code also produced little variations as the decimals truncation.

The *co-clustering* algorithm implemented in this Final Career Project has been proven about 20 times faster than the *co-clustering* from [4].

5. Budget

Considering as a junior engineer a remuneration of 8€ per hour:

Hours per day	Months in the project	Mean days worked per month	Total days worked
7	5	22	110
Total hours	**Remuneration per hour**	**Total remuneration**	
770	8 €	6.160 €	

As the used Software was Matlab:

Matlab individual license
2.000 €

The global cost of the project would be:

Overall budget
8.160 €

6. Conclusions and future development

In this thesis, the methodology of the Region-based particle filter was studied in order to design and implement an efficient structure for it. In order to achieve this, each block of the filter previously explained in Section **3** (*Region-Based particle filter*) was defined.

The two most important blocks of the filter were the *particle support partition* and *co-clustering*. It was considered in a meeting that a generic *co-clustering* method would be the priority of the project, as it is the key part of the Region-based particle filter and it would provide service to the Image Processing Group (GPI in Catalan) once completed.

The designed *co-clustering* method is being tested and used by PhD students working in fields such as 3D representation, segmentation and coding.

The proposed implementation of the co-clustering method presents good results, both in computation time and performance. The computation load of the *co-clustering* has been reduced from 60 seconds [4] to 2.5 seconds (approximately) for a complete partition to partition computation using for both tests an Intel core I5 CPU 650 @ 3.20GHz. Using the *particle support partition* for a specific object propagation this method has been proven to require less than a second.

It was observed in Section **4** (results) that the *co-clustering* method of this project handles very similar results to the original *co-clustering* using less computation time. The little discrepancies between this project and the previous work are due to change of methodology in some algorithms. This *co-clustering* algorithm will be sent to *OpenCV vision challenge* competing in the tracking category.

For future development the perturbation of the particles in Section **3.3.2** (*perturbation*) will be reconsidered. Possible improvements are the *Bhattacharya coefficient* for the weight calculation or the proper *greedy algorithm*. Also specific applications for the filter, such as face or person tracking, will be studied in order to tune the algorithm for these cases.

Bibliography:

[1] M.S. Arulampalam, S. Maskell, N. Gordon and T. Clapp. "A tutorial on particle filters for online nonlinear/non-gaussian Bayesian tracking", *IEEE Trans. on Signal processing*, vol. 50, no. 2, pp. 174-188, 2002

[2] K. Nummiaro, E. Koller-Meier, L. Van Gool. "A Color-based Particle Filter", *First International Workshop on Generative-Model-Based Vision, pp.53-60, 2002*

[3] D. Varas and F. Marques "Region-based particle filter for video object segmentation", *IEEE conference on computer vision and pattern recognition 2014, CVPR 2014*

[4] D. Glasner, S. Vitaladevuni, and R. Basri "Contour-based joint clustering of multiple segmentations", In *CVPR, 2011*.

[5] A. Stein and M. Hebert. Occlusion boundaries from motion: Low-level detection and mid-level reasoning. International Journal on Computer Vision, 82(2):325–357, April 2009.