ΠΑΡΑΚΟΛΟΥΘΗΣΗ ΤΗΣ ΚΙΝΗΣΗΣ ΣΕ ΕΙΚΟΝΟΣΕΙΡΕΣ ΜΕ ΜΟΝΤΕΛΑ ΜΙΚΤΩΝ ΚΑΤΑΝΟΜΩΝ

Η ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Μηχανικών Η/Υ και Πληροφορικής

Εξεταστική Επιτροπή

από τον

ΒΑΣΙΛΕΙΟ ΚΑΡΑΒΑΣΙΛΗ

ως μέρος των Υποχρεώσεων για την λήψη του

ΔΙΔΑΚΤΟΡΙΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΔΕΚΕΜΒΡΙΟΣ 2015

COMMITTEES

Advisory Committee

- **Christophoros Nikou**, Associate Professor, Department of Computer Science and Engineering, University of Ioannina, Greece (Supervisor)
- **Aristidis Likas**, Professor, Department of Computer Science and Engineering, University of Ioannina, Greece
- **Lisimachos-Paul Kondi**, Associate Professor, Department of Computer Science and Engineering, University of Ioannina, Greece

Examination Committee

- **Christophoros Nikou**, Associate Professor, Department of Computer Science and Engineering, University of Ioannina, Greece (Supervisor)
- **Aristidis Likas**, Professor, Department of Computer Science and Engineering, University of Ioannina, Greece
- **Lisimachos-Paul Kondi**, Associate Professor, Department of Computer Science and Engineering, University of Ioannina, Greece
- **Stefanos Zafeiriou**, Senior Lecturer, Department of Computing, Imperial College London, United Kingdom
- **Ioannis Kakadiaris**, Professor, Department of Computer Science, University of Houston, USA
- **Nikolaos Mitianoudis**, Assistant Professor, Department of Electrical and Computer Engineering, Democritus University of Thrace, Greece
- **Konstantinos Blekas**, Associate Professor, Department of Computer Science and Engineering, University of Ioannina, Greece

DEDICATION

I wish to dedicate this work to my family, my teachers and my friends.

ACKNOWLEDGEMENTS

I would like to express my sincere thanks and gratitude to my advisor Prof. Christophoros Nikou for the valuable guidance, advice and encouragement he has offered during the elaboration of this thesis. For the time and effort he spent on my work throughout all these years, dating back to 2008. Our collaboration has been a pleasant and memorable experience that has helped me develop strong research skills.

I would also like to thank the Greek State Scholarships Foundation (IKY) for providing me with a scholarship for my doctoral studies.

December 2015
Vasileios Karavasilis

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ABSTRACT

Vasilios Karavasilis
PhD, Department of Computer Science and Engineering, University of Ioannina, Greece.
December, 2015.
Thesis Title: Visual tracking in image sequences using mixture models.
Thesis Supervisor: Christophoros Nikou.

An important field in computer vision is visual tracking, which is the procedure of generating inference about motion of an object or target in a sequence of images. Solutions to this problem have a variety of applications, some of them being surveillance, action and gesture recognition, motion-based video compression, teleconferencing and video indexing. In tracking problems, it is assumed that the model of the object is known and based on a set of measurements in a video the object's position should be estimated. In this thesis, we focus on the application of clustering methods to model the target's appearance and on the optimization of a cost function to estimate the position of the target and we propose algorithms that improve the state of the art performance or reduce the computational complexity of existing methods.

The first algorithm proposed in this thesis is an extension to the Differential Earth Mover's Distance (DEMD) algorithm for tracking. The contribution of this work is twofold. At first, the representation of the object is accomplished by Gaussian mixture models (GMM) instead of histogram signatures employed in the standard algorithm. This leads to reduced computational cost for real time applications as the algorithm avoids the large dimensionality of histograms. Also, the DEMD algorithm is combined with a Kalman filter to handle occlusions which is a problem not addressed by the original algorithm.

The second algorithm is a variant of the mean shift algorithm where a Gaussian mixture model is employed at each iteration to smooth the differences in the histogram bins representing the appearance of the object. By these means, the algorithm is capable of handling color changes due to variations in the illumination of the scene.

The next algorithm that is proposed herein also relies on Gaussian mixture modeling of the target's appearance. However, compared to the previous approach, the GMM parameters are estimated in the first frame of the image sequence in order to define the appearance of the target. In subsequent frames, the target's position is estimated by maximizing the weighted likelihood of the mixture model by assuming that pixels near the target's geometric center contribute more to the estimation of its position. The advantages of this method are a close-form update for the

target's position, a lower dimension of the target's representation and a reduced computational complexity. Moreover, an update framework is proposed in order to handle cases when the target changes its color due to pose and illumination variations.

An algorithm robust to illumination changes is also proposed which employs only the hue component of the target. As the hue component is periodic, a Gaussian mixture can not model it properly and therefore, a mixture of von Mises distributions is used, which is a circular distribution modeling accurately the hue component of an image. Moreover, the fact that the hue is one dimensional is exploited to discretize it to a finite number of values, which may be computed a priori, thus, speeding up the tracking procedure significantly.

Finally, a framework for visual object tracking based on clustering trajectories of image key points is proposed. The main contribution of the method is that the trajectories are automatically extracted from the image sequence and they are provided directly to a model-based clustering approach. In most other methodologies, the latter constitutes a difficult part since the resulting feature trajectories have a short duration, as the key points disappear and reappear due to occlusion, illumination, viewpoint changes and noise. We present a sparse, translation invariant regression mixture model for clustering trajectories of variable length. The overall scheme is converted into a maximum a posteriori approach, where the Expectation–Maximization (EM) algorithm is used for estimating the model parameters.

Εκτεταμένη Περίληψη στα Ελληνικά

Βασίλειος Καραβασίλης του Χρήστου και της Ειρήνης.
Phd, Τμήμα Μηχανικών Η/Υ και Πληροφορικής, Πανεπιστήμιο Ιωαννίνων.
Δεκέμβριος, 2015.
Τίτλος Διατριβής : Παρακολούθηση της κίνησης σε εικονοσειρές με μοντέλα μικτών κατανομών.
Επιβλέπων: Χριστόφορος Νίκου.

Ένα σημαντικό πεδίο στην περιοχή της υπολογιστικής όρασης είναι η οπτική παρακολούθηση, που είναι η διαδικασία εκτίμησης της κίνησης ενός αντικειμένου ή στόχου σε μια ακολουθία εικόνων. Η επίλυση αυτού του προβλήματος έχει εφαρμογές στην επιτήρηση περιοχών, στην αναγνώριση των κινήσεων ή των χειρονομιών, στην συμπίεση βίντεο με βάση την κίνηση, στις τηλεδιασκέψεις και την κατηγοριοποίηση βίντεο. Στα προβλήματα οπτικής παρακολούθησης, το μοντέλο των αντικειμένων είναι συνήθως γνωστό, και με βάση κάποιες μετρήσεις κατά την διάρκεια του βίντεο, πρέπει να εκτιμηθεί η θέση του αντικειμένου. Η παρούσα εργασία, επικεντρώνεται στην χρήση μεθόδων ομαδοποίησης και πιο συγκεκριμένα στις μικτές κανονικές κατανομές, με σκοπό την μοντελοποίηση της εμφάνισης του στόχου και στην βελτιστοποίηση μιας συνάρτησης κόστους με σκοπό την εκτίμηση της θέσης του στόχου. Προτείνουμε αλγορίθμους που έχουν βελτιωμένη απόδοση σε σχέση με ήδη υπάρχουσες υλοποιήσεις ή μειώνουν την υπολογιστική πολυπλοκότητα ήδη υπαρχόντων μεθόδων.

Ο πρώτος αλγόριθμος που προτείνεται στην παρούσα εργασία είναι μια επέκταση του αλγόριθμου Differential Earth Mover's Distance (DEMD) . Η συνεισφορά αυτής της επέκτασης έχει δύο πλευρές. Αρχικά, για την αναπαράσταση του μοντέλου του αντικειμένου χρησιμοποιούνται μικτές κανονικές κατανομές αντί για υπογραφές ιστογράμματος που χρησιμοποιούνται στον αρχικό αλγόριθμο. Με αυτό τον τρόπο μειώνεται το υπολογιστικό κόστος για εφαρμογές πραγματικού χρόνου, καθώς ο αλγόριθμος αποφεύγει την μεγάλη διάσταση ενός ιστογράμματος. Επίσης, ο αλγόριθμος DEMD συνδυάζεται με το φίλτρο Kalman για να μπορεί να χειριστεί αποκρύψεις του αντικειμένου, που είναι ένα πρόβλημα το οποίο δεν αντιμετωπίζει ο αρχικός αλγόριθμος.

Ο δεύτερος αλγόριθμος είναι μια επέκταση του αλγορίθμου μέσης μετατόπισης στον οποίο μικτές κανονικές κατανομές χρησιμοποιούνται σε κάθε επανάληψη για να εξομαλύνουν τις διαφορές μεταξύ των στηλών του ιστογράμματος που μοντελοποιεί την εμφάνιση του αντικειμένου. Με αυτό τον τρόπο, ο αλγόριθμος μπορεί να χειριστεί αλλαγές στο χρώμα του

αντικειμένου που οφείλονται σε μεταβολές της φωτεινότητας της σκηνής.

Ο επόμενος αλγόριθμος που προτείνεται βασίζεται επίσης σε μικτές κανονικές κατανομές για να μοντελοποιήσει την κατανομή του χρώματος του αντικειμένου. Ωστόσο, σε αντίθεση με την προηγούμενη προσέγγιση, οι παράμετροι της μικτής κανονικής κατανομής που περιγράφει την αναπαράσταση του αντικειμένου υπολογίζονται μόνο στην πρώτη εικόνα της εικονοσειράς. Στις υπόλοιπες εικόνες, η θέση του αντικειμένου υπολογίζεται μεγιστοποιώντας τη σταθμισμένη πιθανοφάνεια του μικτού μοντέλου, με βάση την υπόθεση ότι τα εικονοστοιχεία κοντά στο γεωμετρικό κέντρο του αντικειμένου συνεισφέρουν πιο πολύ στον υπολογισμό της θέσης του. Τα πλεονεκτήματα αυτής της προσέγγισης είναι η κλειστή μορφή της εξίσωσης που εκτιμά την θέση του αντικειμένου, η αντιμετώπιση της μεγάλης διάστασης και οι μικρές απαιτήσεις σε υπολογιστική ισχύ. Επιπλέον, προτείνεται μια μέθοδος για την ενημέρωση του μοντέλου που αναπαριστά το αντικείμενο σε περιπτώσεις που το χρώμα του αντικειμένου αλλάζει λόγω μεταβολών στην επιφάνεια του αντικειμένου ή τον φωτισμό της σκηνής.

Επίσης, προτείνεται ένα αλγόριθμος που είναι ευσταθής σε αλλαγές της φωτεινότητας επειδή χρησιμοποιεί μόνο την τιμή της απόχρωσης του στόχου. Επειδή η απόχρωση είναι περιοδική, δεν μπορεί να χρησιμοποιηθεί η μικτή κανονική κατανομή για να την μοντελοποιήσει επαρκώς. Για το λόγο αυτό, χρησιμοποιείται η μικτή κατανομή von Mises, που είναι περιοδική και μπορεί να μοντελοποιήσει με ακρίβεια την συνιστώσα της απόχρωσης μιας εικόνας. Επιπλέον, το γεγονός ότι οι η συνιστώσα της απόχρωσης είναι μονοδιάστατη χρησιμοποιείται για την διακριτοποίησή της μικτής κατανομής σε πεπερασμένο πλήθος τιμών, που μπορούν να υπολογιστούν εκ των προτέρων, και επομένως να βελτιώσουν την ταχύτητα εκτέλεσης του αλγορίθμου.

Τέλος, προτείνεται ένας αλγόριθμος για την ανίχνευση πολλαπλών αντικειμένων που βασίζεται στην ομαδοποίηση των τροχιών κάποιων σημείων ενδιαφέροντος των αντικειμένων. Η κύρια συνεισφορά της μεθόδου είναι ότι οι τροχιές υπολογίζονται αυτόματα από την εικονοσειρά και χρησιμοποιούνται απευθείας στην διαδικασία κατηγοριοποίησης. Σε άλλες προσεγγίσεις, η κατηγοριοποίηση είναι δύσκολη γιατί οι τροχιές των σημείων ενδιαφέροντος μπορεί να έχουν μικρή διάρκεια, καθώς τα σημεία αυτά εξαφανίζονται και επανεμφανίζονται λόγω επικαλύψεων, αλλαγών στην φωτεινότητα, μεταβολή της θέσης θέασης και θόρυβο. Παρουσιάζουμε ένα αραιό, ανεπηρέαστο από την μετατόπιση μικτό μοντέλο παλινδρόμισης που χρησιμοποιείται για την κατηγοριοποίηση καμπυλών μεταβλητού μεγέθους. Η διαδικασία ομαδοποίησης μεταφράζεται σε μια μεγιστοποίηση της εκ των υστέρων πιθανοφάνειας, όπου ο αλγόριθμος Expectation – Maximization (EM) χρησιμοποιείται για την εκτίμηση των παραμέτρων του μοντέλου.

CHAPTER 1

INTRODUCTION

1.1 Visual Tracking

Computer vision is a field of computer science and engineering that designs methods for extracting semantic information from images. Usually, the input to such methods is an image and the objective is to extract some information or infer an interpretation from the the image content.

In this process, the first step is image acquisition, where images are captured by sensors like cameras and are represented as 2D signals. When a series of images are recorded from the same camera in consecutive time moments an image or video sequence is obtained. As sampling is relatively dense, the differences between consecutive frames are not very large and the variations between them result from camera motion, partial or total scene motion, occlusions and their combinations.

The objective of visual tracking, which is the topic in this thesis, is to estimate the position of an object in an image sequence. Usually, the position and shape of the object in the first frame of the image sequence is known. Thus, in every frame, except from the first one, a tracking algorithm has to locate the object using information on the position of the object in the previous frames and the appearance of the object.

Tracking can be difficult for a number of reasons. At first, the individual images may probably have large sizes (generally over 1 Mpixel) and the recording frame rate is relatively large (generally over 25 pictures per second). Therefore, as the image resolution and the sampling rate increase, the demands for storage space and computational power also increases. Nevertheless, in visual tracking, the localization of objects must be done in real time in many applications. Consequently, the algorithms must be efficient with respect to execution time in order to be

able to process each image before the appearance of the next frame in a video sequence. On the other hand, there are cases where a number of frames may be dropped or the algorithm has to handle low resolution frames, leading to loss of measurement and consequently missing of the tracked target. Another issue may be the change of the object's appearance, color or both. Changes in appearance may occur due to physical properties (e.g. a person walking) or due to occlusions (e.g. a car passes in front of another car). Changes in color may happen due to rotation (e.g. a colorful ball rolling) or due to changes in the illumination of the scene (e.g. reflections). Finally, a major issue arises when the objects to be tracked have similar color with other objects which are considered as parts of the background. Other factors making tracking difficult may be the projection from the three dimensional real word to the two dimensional image space, acquisition noise, complex non-rigid movement of the object and the prerequisite for real time execution.

Despite the fact that tracking an object efficiently in every possible scenario is an open issue, many approaches have been proposed which solve the problem when various constraints are assumed to apply to the motion model or the appearance of the object. For instance, many methods assume that the motion is smooth without abrupt changes, or the illumination of the object does not change dramatically, or the acceleration of the object is constant for some time period or the object's shape can be modeled by a primitive geometric shape. These simplifications enable the design of algorithms that can track objects efficiently for a specific application.

Finally, tracking can be seen as a standalone process, where we are only interested in the position and shape of a target through time, or as an intermediate step in order to handle a broader problem. Knowing the object's position can be useful in many applications, such as action recognition based on motion (e.g. recognition based on gait or gesture), human identification, video-based surveillance, human – computer interaction, automatic vehicle driving and character animation in computer graphics.

1.2 Contribution of the thesis

In this thesis, we study the problem of visual tracking in image sequences by using mixture models in order to model the appearance of the target or its trajectory. Mixture models provide a compact representation which is robust to slight color changes and provide the foundation for real time tracking by eliminating the curse of dimensionality. The structure of the rest of the thesis is organized as follows:

In Chapter 2, the related literature on visual tracking is reviewed with respect to the methods proposed in this thesis.

In Chapter 3, an extension to the Differential Earth Mover's Distance method is proposed which uses Gaussian mixture models to model the appearance of the target. We demonstrate how the DEMD may be used for visual tracking in synergy with Gaussian mixtures models. According to the appearance model, motion between adjacent frames results in variations only of the mixing proportions of the Gaussian components representing the object to be tracked.

These variations are computed by minimizing the differential EMD between Gaussian mixtures, yielding a very fast algorithm with high accuracy, without recurring to the EM algorithm in each frame. Moreover, we also propose a framework to handle occlusions, where the estimation of the object's location is forwarded to an adaptive Kalman filter whose parameters are estimated online by the motion model already observed. Thus, the algorithm can not only handle occlusions but also predict the objects future position.

In Chapter 4, an extension to the mean shift algorithm is proposed which smooths the histogram in order to handle global scene illumination changes. Mean shift is based on the minimization of the distance between the discrete histogram of the target and the discrete histogram of the neighborhood of a candidate image location. While the algorithm performs well when the target's appearance and the lighting conditions are constant, it may fail when these conditions are not met because the ideal histogram is generally shifted with respect to the reference histogram. In this chapter, we propose to compute the initial histogram of the target using a Gaussian mixture model rather than impulses generated by simple counting. This mixture plays the role of a weighting function, in the histograms computed in subsequent frames, in order to make them smoother and increase the overlapping area with the initial histogram. By these means, sudden illumination changes between consecutive frames may exhibit smoother transitions between the two histograms and the involved distance is not trapped into local minima. This methodology is not tightly tied to mean shift and can be applied to other methods that use histograms in order to represent the target appearance and do not use cross bin metrics.

In Chapter 5, a new method that uses weighted Gaussian mixture model and likelihood maximization is proposed. In the probabilistic real time tracking algorithm that is proposed, the target's feature distribution is also represented by a Gaussian mixture model. However, compared to the approach of Chapter 3, there is only one Gaussian mixture model estimated. The target localization in the image sequence is achieved by maximizing the weighted likelihood with respect to the object location having the GMM parameters constant. The role of the weight in the likelihood definition is important as it allows gradient based optimization to be performed, which would not be feasible in a context of standard likelihood representations. Moreover, the algorithm handles scale and rotation changes of the target, as well as appearance changes, which modify the components of the GMM. The proposed appearance update framework uses only in the previous target positions in order to determine if an update to the appearance must take place and can be combined with other tracking approaches.

In Chapter 6, a new method based on a weighted von Mises mixture model is proposed in order to represent the hue component of the target. The mixture weights, which are provided by a spatial kernel, along with the hue values are used in order to estimate the parameters of the weighted von Mises mixture model. As the hue component is periodic, approaches that use distributions that are designed for linear spaces (e.g. Gaussian distribution) can not be applied. The von Mises distribution is suitable for circular data and it is employed in order to eliminate drawbacks in kernel-based tracking caused by eventual shifts of the target's histogram bins. The weights allow a mean shift-like gradient based optimization by maximizing the weighted likelihood, which would not be feasible in the context of a standard von Mises mixture. Moreover,

as only the hue component of the target is involved, many quantities of the algorithm may be pre-calculated for fixed parameters and therefore the algorithm can perform in real time.

In Chapter 7, we present a framework for visual object tracking based on clustering trajectories of image key points extracted from an image sequence. The main contribution of our method is that the trajectories are automatically extracted from the image sequence and they are provided directly to a model-based clustering approach. In most other methodologies, the latter constitutes a difficult part since the resulting feature trajectories have a short duration, as the key points disappear and reappear due to occlusion, illumination, viewpoint changes and noise. We present a sparse, translation invariant regression mixture model for clustering trajectories of variable length. The overall scheme is converted into a maximum a posteriori approach, where the expectation -maximization (EM) algorithm is used for estimating the model parameters. The proposed method simultaneously detects the distinct objects in the input image sequence by assigning each trajectory to a cluster, and provides their motion which is represented by the mean trajectory of each cluster.

In Chapter 8, we provide an overall review of the proposed methods, summarize the basic conclusions and indicate open issues and interesting directions for future work.

CHAPTER 2

VISUAL TRACKING

Visual target tracking is a preponderant research area in computer vision with many applications such as surveillance, targeting, action recognition from motion, motion-based video compression, teleconferencing, video indexing and traffic monitoring. Tracking is the procedure of generating an inference about motion given a sequence of images. Based on a set of measurements in image frames the object's true position should be estimated. Various approaches have been proposed in order to solve this problem. However, classifying the various tracking algorithms into categories is not a straightforward process [107, 130, 140]. This results from the fact that the tracking process may be split in phases, with a different approach for each phase. The first phase is to create a representation of the object based on its initial view. In this phase, various features can be employed, like color, texture, or higher level features. The second phase is the estimation of the object's position. This can be accomplished by exhaustive search, methods using gradient based optimization, filtering, or data association. The final step is the model update. This is related to the first step, but it can also employ information about the motion. In [53], it shown that an accurate appearance model is considerably more effective than a strong motion model. Thus, the appearance model that is chosen in the first phase has great impact on the strategy that is followed in the second phase to locate the target. Moreover, some algorithms may not include a phase at all or integrate more than one phases simultaneously. In this chapter, an overview of recent tracking algorithms is presented, with respect to the appearance and the localization of the target.

2.1 Target representation

Images in visual tracking are represented by two dimensional matrices with elements being scalar (in grayscale images) or vectors (in color images). Moreover, many features can be used in order to describe each pixel. In grayscale images one may employ the illumination while in RGB images the color is more frequently used. In gradient images the gradient in each direction (or its equivalent angle and magnitude) are employed. Even though there are some differences among them, the majority of the algorithms handle this issue in the generic case of two dimensional matrix, with the scalar image being a special case of a multi-channel image. The object to be tracked is a region of the image in a specific place. In what follows, the terms object, target and image subregion of the object are considered interchangeable. Thus, in order to represent the object various methodologies have been employed.

2.1.1 Standard representation approaches

The most straight forward approach is to represent the target by a template [38, 85, 93, 132, 121, 148, 144, 145, 154]. In this case, the appearance model can be thought as an array of values, which can be formed by concatenating the columns of the target's image subregion. This approach has the advantage that it is easy to implement, but if the target changes its appearance, for example by rotating, the new template will be significantly different.

Another approach estimates the distribution of some features from the target [66, 71, 99, 136, 152]. The feature distribution is usually represented by a histogram, although some other approaches use histogram signatures or some sort of mixture model. This approach has the advantage that the distribution does not change when some simple variations to the shape of the object take place, for example in case of rotation, but it has the disadvantage that the object is not strictly defined. Moreover, for higher dimensional features, for example RGB histograms, the model's accuracy in representing the target's appearance can not be easily evaluated.

Finally, other methods represent the object by a set of points [34, 50, 52, 77, 91, 96, 139, 146], small image patches [38, 59, 118, 147, 154] or a combination of them [131, 138]. Some constrains may apply to this set of points in order to enforce them form a structure as each point is individually tracked using optical flow [78, 105]. However, the representation of each point relies on simple approaches such the ones mentioned above, for example each key point may be represented by a template of the color of its surrounding window. Algorithms in this category may track each point individually and afterward estimate the position of the complete object, or use the complete set of points in order to estimate the position of the object in one step.

2.1.2 Gaussian mixture models for target representation

An alternative method of representing the distribution of the features instead of a histogram is to use a continuous distribution. Due to their theoretical simplicity and the efficiency of modeling many distributions, the Gaussian mixture models will be presented as an example. Gaussian mixtures have been widely used in computer vision for image segmentation [92], background

subtraction [26, 89], image classification [21] and human pose estimation [40]. In visual tracking, GMM have been employed to model the appearance of the target or as a support to the tracking procedure. One work in the latter category is presented in [97], where a generic online multi-target track-before-detect method is proposed that is applicable on confidence maps used as observations. The main novelty is the inclusion of the target identity in the particle state, enabling the algorithm to deal with unknown and large number of targets. In order to avoid identity switches of close targets, the state estimate of a target is performed via mean shift clustering and supported by GMM in order to enable an accurate assignment of identities within each single cluster. In other works employing particle filters for visual tracking [75, 82] the transition model of the particles is described by a GMM around an approximation of the state posterior distribution of the previous frame.

The appearance of the target using a variation of the Gaussian distribution is proposed in [36]. The asymmetric generalized Gaussian distribution is formulated by having two variance parameters, one for the left part and one for the right part of the distribution, and it is capable of modeling non-Gaussian asymmetrical data. The proposed mixture of multidimensional asymmetric generalized Gaussian distributions is used for pedestrian detection and multiple target tracking. A standard Gaussian mixture model for target appearance modeling is proposed in [62], where Gaussian mixtures are used to represent the appearance of the target. The target position is estimated using particles whose weights are computed by marginalizing out the appearance models. The target is divided in subregions; the features of the pixels inside each subregion are used to estimate the parameters of a GMM and the appearance distribution of the whole target is a combination of the distributions of the non-overlapping subregions.

2.2 Tracking by filtering

The algorithms based on filtering assume that the moving object has an internal state which may be measured and, by combining the measurements with the model of state evolution, the object's position is estimated. In every frame, the previous state of the algorithm is combined with the measurement in order to provide the current state. Subsequently, based on the current state, the state transition model gives the next state. The first method of that category is the Kalman filter [106] which successfully tracks objects even in the case of occlusion if the assumed type of motion is correctly modeled [32]. Another approach in this category are the particle filters [6, 76, 117, 123, 137, 149]. This category also includes Condensation [54] and ICondensation [55] algorithms which are more general than Kalman filters, as they do not assume specific type of densities and, using factored sampling, have the ability to predict an object's location under occlusion as well. The term particle filter is usually used when the same tracker is applied with different parameters (e.g. various initial positions). However, in [10], the results of multiple trackers are merged using a weighting scheme. Moreover, trackers having consistently poor performance are removed and continuous trajectories are favored. In the same spirit, trackers for single person and multiple persons are combined in [112] in order to track

people in crowded scenes. In [73], multiple image patches that are robust for visual tracking are identified through a particle-filter based method. The patches can overlap with each other and they are tracked using a base tracker. In [60], multiple samples taken from different cameras are used in order to locate a target. Due to the fact that a camera may have less information about the target with respect to the other cameras (e.g. the target may not be visible in its image), a weighted approach is proposed in order make the samples affect the state of the target differently. The problem of multiple targets tracking is handled by solving the matching problem between multiple measurements per camera and their corresponding states.

These methods have the drawback that the type of object's movement should be correctly modeled. The motion model in the case of Kalman filter is important, as it only has one internal state and if this singe state fails to estimate the position of the object, it will fail. On the other hand, due to the fact that particle filters usually have many internal states, a small subset of them may be sufficient to predict the correct state. However, due to the sampling procedure between state transitions, the particle filters must have mechanisms in order to prevent the particles from concentrating to only one state. This problem is bypassed by incorporating initialization of new particles and inclusion of noise in state transitions. In [79], a modified evolutionary computing method for the Condensation algorithm is introduced which resolves the particle impoverishment under a proper size of particle population.

Another problem may be the sampling procedure. These methods can be combined with external algorithms in order to handle the sampling of each step. For example, in the case of Kalman filter, the target's position can be estimated using another tracking algorithm, while in particle filters, the similarity measure can be the difference of the template of the target candidate and the target model. This implies that the external algorithms must be efficient in terms of computation time, especially in the case of particle filters, where one sample must be drawn for each particle.

The advantage of these methods is their implementation simplicity and the easy integration with other algorithms in the sampling step. Moreover, due to their transition model, a prediction for the future object's position can be made, which can be employed in order to successfully track the object under partial or full occlusions.

2.3 Tracking by gradient based optimization

A major subset of trackers in visual tracking address the problem of model-free shape by employing spatial kernels and modeling the color distribution of the target [27, 70, 71, 72, 74, 120]. In this category, tracking algorithms employ a probabilistic model of the object appearance and try to detect this model in consecutive frames of the image sequence. More specifically, color or texture features of the object, masked by an isotropic kernel, are used in order to create their histogram. Then, the object's position is estimated by minimizing a cost function between the target's model and candidate histograms. The key idea of this family of algorithms is the representation of the target by an primitive shape, which is usually an ellipse. Combining the

ellipse with a spatial kernel eliminates the effect of varying object dimensions (e.g. a long thin object) and allows tracking of a wide variety of targets. Each pixel inside the ellipse is assigned a weight, with the maximum weight characterizing the pixel at the center of the ellipse. The intuition behind this modeling is that pixels near the center of the ellipse are more likely to belong to the object in contrast to pixels near the boundary. Masking the object with a kernel allows a gradient-based optimization of a cost function instead of a brute force search for target localization and real-time performance may be achieved on a standard personal computer.

A representative method in this category is the mean shift algorithm [20, 30] where the object is supposed to be inside an ellipse and the histogram is constructed from pixel values inside that ellipse. In the first frame, it estimates a target model which is represented by a histogram. In consecutive frames, the location in which the corresponding histogram is similar to the target model is estimated. In [150], the mean shift algorithm is extended in order to estimate the orientation and scale of the target. In [153], scale invariant features are used and a similarity measure between two neighboring frames in terms of color and SIFT correspondence is computed and the expectation-maximization algorithm is employed in order to estimate a maximum likelihood solution. In [135], various distance measures are associated with the mean shift algorithm and in [125], the advantages of using a more detailed shape model instead of a generic ellipse for target representation is investigated. Other approaches using multiple kernels [39] and a Newton style optimization procedure [46] were also proposed.

However, the original algorithm shows some limitations which were recently addressed. More specifically, mean shift fails to track the object when the histogram of the model changes during time. It compares only the corresponding bins between histograms, so if the bins values are shifted, then the object may be lost. This is common due to illumination changes (where the histogram bins are shifted), view point changes (i.e. 3D rotation) or reappearance after occlusion and the algorithm may not handle the overall drift in the histogram of the target. To tackle these limitations, the tracker in [151, 152] minimizes the EMD between the target model and the target candidate histograms. The movement in each iteration of the algorithm is one pixel, due to the fact that there is no closed form solution in order to update the center of the ellipse. In [69], a tracker that minimizes the EMD for the case of 1D feature histograms and a tracker which minimizes a cross-bin metric that is based on histogram smoothing for multidimensional features histogram are proposed. In [49], the target consists of overlapping regions, whose weighted histograms are estimated. The histogram of the spatially corresponding regions between the target model and the target candidate are compared using the EMD distance and an exhaustive search framework is employed in order to estimate the target's position. Moreover, the histogram of model's regions that their distance with the corresponding candidate regions is above a threshold are updated, as the appearance of the target in these regions is likely to have changed. The work in [70], enables mean shift to use multiple reference histograms obtained from different target views or from different target states and the convex hull of these histograms is used as the target model. In [74], the target appearance is modeled using a sparse coding histogram based on a learnt dictionary. A sparse representation-based voting map is used to regularize the mean shift algorithm in order to adapt it to appearance changes and limit

the drifting. The HSV color model can be employed in order to eliminate the issues caused from illumination change. The hue component is a flexible representation, due to the fact that is closely related to what humans perceive as color. Moreover, hue is unrelated to illumination changes, as these changes are encoded in the saturation and value components. These properties are highlighted in [20] in order to support the authors decision to use only the hue component. Another advantage of using the hue component of the HSV color space instead of the full RGB color space is that the dimensions of the problem are reduced to one instead of three. The hue component does not depend on illumination changes, but this does not prevent its histogram bins to be shifted. In these cases, we can not directly apply the approaches that were proposed for the RGB histogram based methodologies, due to the fact that the hue is periodic with period 2π and these methodologies have been proposed for linear color spaces.

Another case where mean shift fails is when the object's motion is abrupt and the target ellipses in two consecutive frames do not intersect, which results from the local optimization performed in the framework of kernel-based trackers. Combination with Kalman filter [8] or particle filters [126] may give a solution to this problem, when the predicted state of the filter is close to the next frame's target's position. The work in [72] addresses this drawback by employing a pyramidal decomposition to capture distant targets between consecutive frames. An extension of the main algorithm is proposed in [71], which may handle cases where the color of the target is similar with the color of the background and the displacements are large. The disambiguation between target and background is achieved by a model incorporating information about the spatial context of the target and large displacements are handled by increasing the candidate scales.

2.4 Multi-target tracking

The above methods track only one object at a time. Other works track many objects simultaneously [5, 9, 12, 87, 113, 124, 134] and in these cases occlusions may be detected more efficiently. These methods assume that a partial or total occlusion between objects may take place and use this information in order to make the tracking procedure more robust. In [1], prior knowledge of objects' movement is used in order to detect occlusions. Moreover, the object to be tracked is usually represented by its color histogram, but this is not always necessary. A GMM was used in [122] to represent the object in a joint spatial-color space and in [110] for background subtraction. Furthermore, the object may be represented by a contour [103, 141] or a level set [31, 81, 90, 95, 101]. Other approaches that employ level-sets tracks the objects [13, 51] by optimally grouping regions whose pixels have similar feature signatures. Combining multiple object representations could make the tracking procedure more robust [47, 88, 116, 129]. Also, in [16], multiple views of an object are learnt through principal component analysis (PCA) and a support vector machine (SVM) classifier was also used in [7]. In [22, 94], graph cuts [18] were employed in order to segment each frame into possible objects. An application in vehicle tracking is presented in [80] where multiple vehicles are tracked by

initially assigning each pixel either to background, or foreground and applying next a Kalman filter to estimate the vehicle position and associate each foreground pixel with a single object. In cases of articulated objects (e.g. pedestrians) stereo depth information has been used in order to acquire a 3D articulated structure per object and then estimate the 3D trajectory of each object [44]. Moreover, combining multiple object representations could make the tracking procedure more robust [47].

Motion segmentation constitutes a significant application of tracking algorithms, which aims at identifying moving objects in an image sequence. It can be seen either as the post-processing step of a tracking algorithm, or as an assistive mechanism of the tracking algorithms by incorporating knowledge to the number of individual motions or their parameters. It has been considered as an optical flow estimation [15] where violations of brightness constancy and spatial smoothness assumptions are addressed. In [127], an alternative scheme is used where small textured patches with uniform optical flow are detected and clustered into layers, each one having an affine flow. Likewise in [100], image features are clustered into groups and the number of groups is updated automatically over time in.

Trajectories of image key-points extracted from an image sequence have also been used for tracking. Grouping 3D trajectories is proposed in [23] using an agglomerative clustering algorithm where occlusions are handled by multiple tracking hypotheses. Finite mixtures of hidden Markov models (HMMs) were also employed in [4] where training is made using the EM algorithm. In [142], it is assumed that trajectories belonging to different objects lie in different subspaces, thus, the segmentation can be obtained by grouping together all the trajectories that generate these subspaces. The grouping is obtained by the eigenvectors of an affinity matrix which contains the pairwise distances between trajectories computed in the corresponding subspaces. The number of motions can be estimated based on the number of eigenvalues of the symmetric normalized Laplacian matrix. In [143], a two stage procedure is proposed. Initially, an iterative clustering scheme is applied that groups temporally overlapping trajectories with similar velocity direction and magnitude. Next, the clusters created are merged covering larger time spans. In [128], faces are detected in each frame and tracked for a short period. Then, the short trajectories are iteratively clustered and linked into longer trajectories by finding long tracks of faces that are consistent in motion and appearance.

Furthermore, spectral clustering approaches were also proposed, such as in [67] where the motions of the tracked feature points are modeled by linear subspaces, and the approach in [57], where missing data from the trajectories are filled in by a matrix factorization method. Moreover, in [133] a linear manifold is estimated for every trajectory and then spectral clustering is employed to separate these subspaces. In [63], motion segmentation is accomplished by computing the shape interaction matrices for different subspace dimensions and combine them to form an affinity matrix that is used for spectral clustering.

Finally, many methods proposed independently rely on the separation of the image into layers. For example, in [83], tracking is performed in two stages: at first foreground extracted blobs are tracked using graph cut optimization and then pedestrians are associated with blobs and their motion is estimated by a Kalman filter.

CHAPTER 3

TRACKING USING THE EARTH MOVER'S DISTANCE BETWEEN GAUSSIAN MIXTURES

3.1 Introduction

In the Differential Earth Mover's Distance tracking algorithm [151, 152], the object is represented by a histogram (called a signature) and the distance between signatures in consecutive frames to be minimized is the Earth Mover's Distance [102]. The computational complexity of the EMD prevents a direct implementation in many real time applications. To overcome this drawback, the DEMD algorithm based on sensitivity analysis of the simplex method provides an acceleration compared with its standard counterpart [151, 152].

Motivated by the efficiency of the differential EMD tracking algorithm [151, 152] and the compactness of the representation of probability densities using Gaussian mixture models [14], we propose to first model the appearance of the target by a Gaussian mixture model trained on a weighted likelihood and then to employ the differential EMD approach for tracking. According

to our model, motion between adjacent frames results in variations of the mixing proportions of the Gaussian components representing the object. These variations affect the distance between the mixtures, at the same image location, representing the object in consecutive frames. By these means, the gradient of the EMD, namely the differential EMD [151, 152], between Gaussian mixtures shows the direction of the minimum and consequently the target location.

Moreover, in a second part of this chapter, we propose to consider the estimated location of the target as a measurement (observation) of a time-varying Kalman filter in order to address cases presenting occlusions. Hence, the prediction for the object's location is forwarded to a Kalman filter whose state matrix parameters are not constant but they are updated on-line based on recent history of the estimated motion.

The contribution of the presented chapter is twofold. At first, the proposed approach leads to a significant improvement in terms of execution time with respect to the differential EMD tracking algorithm [151, 152] without compromising the accuracy of the method. At second, based on the motion model already observed, occlusions are successfully handled by modifying on-line the state matrix of a Kalman filter.

The remainder of the chapter is organized as follows: In section 3.2, the modeling of the object to be tracked by a Gaussian mixture is presented. The tracking algorithm relying on the minimization of the Earth Mover's Distance between Gaussian mixtures is presented in section 3.3. In section 3.4, the extension of the algorithm in order to address the problem of occlusion is described. Experimental results are shown in section 3.5 which are followed by our conclusions in section 3.6.

3.2 Target appearance modeling

In this section we present the basic idea of minimizing the Earth Mover's Distance between Gaussian mixture models for tracking. We describe the GMM as a way of representing an object's appearance and define the EMD as a distance between two GMM.

3.2.1 Background on weighted Gaussian Mixture Models

A one dimensional Gaussian distribution has a probability density function given by

$$\mathcal{N}(I_n; \mu, \sigma) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(I_n - \mu)^2}{2\sigma^2}\right) \tag{3.1}$$

Where I_n is the intensity of the n^{th} pixel, μ is the mean value and σ^2 is the variance of the distribution.

Let two Gaussian distributions be:

$$f_1(I_n) = \mathcal{N}(I_n; \mu_1, \sigma_1), \qquad f_2(I_n) = \mathcal{N}(I_n; \mu_2, \sigma_2). \tag{3.2}$$

The Gaussian Mixture Model (GMM) is a convex combination of Gaussian components

[14]. A single component is given by (3.1) and the GMM with m components is expressed by

$$f(I_n; \mu, \sigma, \pi) = \sum_{i=1}^{m} \pi_i \mathcal{N}(I_n | \mu_i, \sigma_i) \tag{3.3}$$

where $\mu = \{\mu_i\}_{i=1,\ldots,m}$, $\sigma = \{\sigma_i\}_{i=1,\ldots,m}$ and $\pi = \{\pi_i\}_{i=1,\ldots,m}$, are the model parameters. The parameters π_i represent the importance of each component and satisfy the constraints $\sum_{i=1}^{m} \pi_i = 1$ and $\pi_i \geq 0, \forall i = 1, \ldots, m$.

We assume that we have grayscale images, and each object may be described by the intensities of its pixels. An object is represented by an ellipsoidal region, and the object's pixels are those lying inside that region. The usual way to represent an object is by histograms of m_h bins. This approach has the disadvantage that the number of the bins must be specified *a priori*. However, it is a very common and efficient way of modeling the object to be tracked in the majority of the state of the art trackers [30].

In this chapter, we propose the representation of an object using a GMM. The parameters of the GMM are estimated by clustering the density values of object's pixels using the EM algorithm [14]. An advantage of the GMM representation is that the number of components m is significantly smaller than the number of distinct intensities.

Every object may be represented by an ellipsoidal region with finite precision. As an effect, inside the ellipse, there will be regions not belonging to the object. Usually, these regions exist at the edges of the ellipse. To eliminate the influence of regions not belonging to the object, the ellipse is weighed by a kernel as will be explained bellow.

At first, we assume that the center of the ellipse is in the spatial location $(0, 0)$. Then, the ellipse is normalized to a unit circle by dividing each pixel coordinates by h_x and h_y, which are the sizes of the ellipse in the horizontal and vertical directions respectively. Let the normalized pixel locations be (x_n, y_n). An isotropic kernel, with profile $k(x)$, is applied to pixels inside the unit circle to attribute corresponding weights at every pixel. The weight for a pixel indexed by n is defined by

$$w_n = \frac{k(x_n^2 + y_n^2)}{\sum_{i=1}^{N} k(x_i^2 + y_i^2)}. \tag{3.4}$$

Notice that $x_n^2 + y_n^2 \leq 1$ because the point (x_n, y_n) is inside unit sphere and $\sum_{n=1}^{N} w_n = 1$. The kernel profile $k(x)$ is a convex monotonic decreasing function such that $k : [0, \infty) \rightarrow \Re$ and g is the negative derivative of the kernel function, $g(x) = -k'(x)$. We use a kernel with Epanechnikov profile [30]

$$k(x) = \begin{cases} \frac{1}{2}(1 - x) & if \quad x \leq 1 \\ 0 & otherwise \end{cases} \tag{3.5}$$

Let $\mathbf{I} = \{I_n\}_{n=1,\ldots,N}$ be the intensity values of the pixels inside the unit circle as defined above and let $\mathbf{W} = \{w_n\}_{n=1,\ldots,N}$ be the corresponding normalized weights of each sample. The

weighted likelihood of the model is expressed by

$$L(\mathbf{I}, \mathbf{W}; \mu, \sigma, \pi) = \sum_{n=1}^{N} w_n \log \left(f(I_n; \mu, \sigma, \pi) \right)$$
$$= \sum_{n=1}^{N} w_n \log \left(\sum_{i=1}^{m} \pi_i \mathcal{N}(I_n; \mu_i, \sigma_i) \right) \tag{3.6}$$

and the update equations of the Expectation - Maximization (EM) algorithm that maximize this likelihood are:

- Expectation step: Compute responsibilities

$$\gamma_{ni} = \frac{\pi_i \mathcal{N}(I_n; \mu_i, \sigma_i)}{\sum_{j=1}^{m} \pi_j \mathcal{N}(I_n; \mu_j, \sigma_j)}. \tag{3.7}$$

- Maximization step: Estimate parameters

$$\hat{\pi}_i = \sum_{n=1}^{N} w_n \gamma_{ni}, \tag{3.8}$$

$$\hat{\mu}_i = \frac{\sum_{n=1}^{N} w_n \gamma_{ni} I_n}{\sum_{n=1}^{N} w_n \gamma_{ni}}, \tag{3.9}$$

$$\hat{\sigma}_i^2 = \frac{\sum_{n=1}^{N} w_n \gamma_{ni} (I_n - \hat{\mu}_i)^2}{\sum_{n=1}^{N} w_n \gamma_{ni}}. \tag{3.10}$$

The above iterations are repeated until convergence of the likelihood. In our method the EM algorithm is applied at the initialization step and when significant changes are observed, in order to infer the GMM parameters that are to be tracked in the following frames.

3.2.2 Earth Mover's Distance between Gaussian Mixture Model

Having computed the parameters $\mu = \{\mu_i\}_{i=1,\ldots,m}$, $\sigma = \{\sigma_i\}_{i=1,\ldots,m}$, $\pi^M = \{\pi_i^M\}_{i=1,\ldots,m}$ of the object's model, in the next frame we assume that the new center of the ellipse comprising the target is located at the normalized coordinates $\mathbf{y}$ of the next frame. In the above notation, the exponent M in π^M represents the object's model. We assume that the colors of the object and the background do not change abruptly, so the target GMM candidates have the same mean and variance as their counterpart in the initial frame and the only difference is the importance (mixing proportion) of each component. The model for the background may change between frames but it should have limited overlap with the model of the object. The only problem is when pixels from the background have the same intensity with pixels belonging to the object (camouflage). In case the mixing proportions of the GMM do not change smoothly between frames, this is an indication that important illumination changes occur. Therefore, the GMM needs to be trained again to take into account the new illumination conditions.

An illustrative example is presented in figure 3.1, where the number of components $m = 3$, highlights the change in each component's importance. The value for m is appropriate for this example. This parameter actually depends on the colors of the object (e.g. it can be determined by the number of colors belonging to the object and those belonging to the background). Statistical criteria may also be employed in order to estimate more accurately the number of components [41]. In the images at the left, the ellipse remains at the same spatial location, while the racket is moving downwards. In the right figures, the horizontal axis represents the gray levels and the vertical axis represents the probability of each gray level (3.3). Each GMM has three components. As the racket is moving outside of the ellipse, the mixing proportions associated with the object get smaller while the mixing proportion representing the background is increasing.

Therefore, the GMM parameters for the candidate object in the next frame are (described by an exponent C) are $\mu = \{\mu_i\}_{i=1,\ldots,m}$, $\sigma = \{\sigma_i\}_{i=1,\ldots,m}$, $\pi^C(\mathbf{y}) = \{\pi_i^C(\mathbf{y})\}_{i=1,\ldots,m}$, where the mixing proportions depend on the location $\mathbf{y}$. This means that the centers μ_i and the variances σ_i^2 remain unchanged through time. Also, by assuming that $\pi_i^C(\mathbf{y})$ do not change dramatically through time, equations (3.7) and (3.8) of the EM algorithm may be used to estimate the proportions $\pi_i^C(\mathbf{y})$. We must point out that the EM is used only in the initial image. In all other images, only computation of the proportion $\pi^C(\mathbf{y})$ is made, as means μ and variances σ remain unchanged (due to the fact that the color and the luminance of the object remain unchanged). By substituting $\pi_i \leftarrow \pi_i^M$ and $\hat{\pi}_i \leftarrow \pi_i^C(\mathbf{y})$ in (3.7) and (3.8) respectively, the mixing proportions for the candidate object is:

$$\pi_i^C(\mathbf{y}) = \frac{1}{N}\sum_{n=1}^{N} w_n^C(\mathbf{y}) \frac{\pi_i^M \mathcal{N}(I_n^C(\mathbf{y}); \mu_i, \sigma_i)}{\sum_{j=1}^{m} \pi_j^M \mathcal{N}(I_n^C(\mathbf{y}); \mu_j, \sigma_j)} \tag{3.11}$$

where $I_n^C(\mathbf{y})$ is the image intensity of the n^{th} pixel of the candidate object at location $\mathbf{y}$, $w_i^C(\mathbf{y})$ are the normalized pixel weights inside the unit circle in the next image and N is the number of the pixels.

As the means μ_i and the variances σ_i^2 of the GMM in the initial and the current frames are the same, the parameters for the first GMM are defined by $\mu = \{\mu_i\}_{i=1,\ldots,m}$, $\sigma = \{\sigma_i\}_{i=1,\ldots,m}$, $\pi^M = \{\pi_i^M\}_{i=1,\ldots,m}$ and for the second one are $\mu = \{\mu_i\}_{i=1,\ldots,m}$, $\sigma = \{\sigma_i\}_{i=1,\ldots,m}$, $\pi^C(\mathbf{y}) = \{\pi_i^C(\mathbf{y})\}_{i=1,\ldots,m}$. We define the Earth Mover's Distance (EMD) between two GMM as [102]:

$$EMD(\mathbf{y}) = \min_{f_{uv}} \left[\sum_{u=1}^{m}\sum_{v=1}^{m} f_{uv}(\mathbf{y}) d_{uv} \right] \tag{3.12}$$

subject to

$$\begin{array}{ll} \sum_{u=1}^{m} f_{uv}(\mathbf{y}) = \pi_v^C(\mathbf{y}), & 1 \le v \le m \\ \sum_{v=1}^{m} f_{uv}(\mathbf{y}) = \pi_u^M, & 1 \le u \le m \\ \sum_{u=1}^{m}\sum_{v=1}^{m} f_{uv}(\mathbf{y}) = 1 & \\ f_{uv}(\mathbf{y}) \ge 0, & 1 \le u \le m, 1 \le v \le m \end{array} \tag{3.13}$$

where d_{uv} is the symmetric Kullback-Leibler distance given by

$$d_{u,v} = \frac{1}{2}\left[\frac{\sigma_u^2}{\sigma_v^2} + \frac{\sigma_v^2}{\sigma_u^2} + (\mu_u - \mu_v)^2 \left(\frac{1}{\sigma_u^2} + \frac{1}{\sigma_v^2} \right) - 2 \right] = D_{KL}(f_u|f_v) + D_{KL}(f_v|f_u), \tag{3.14}$$

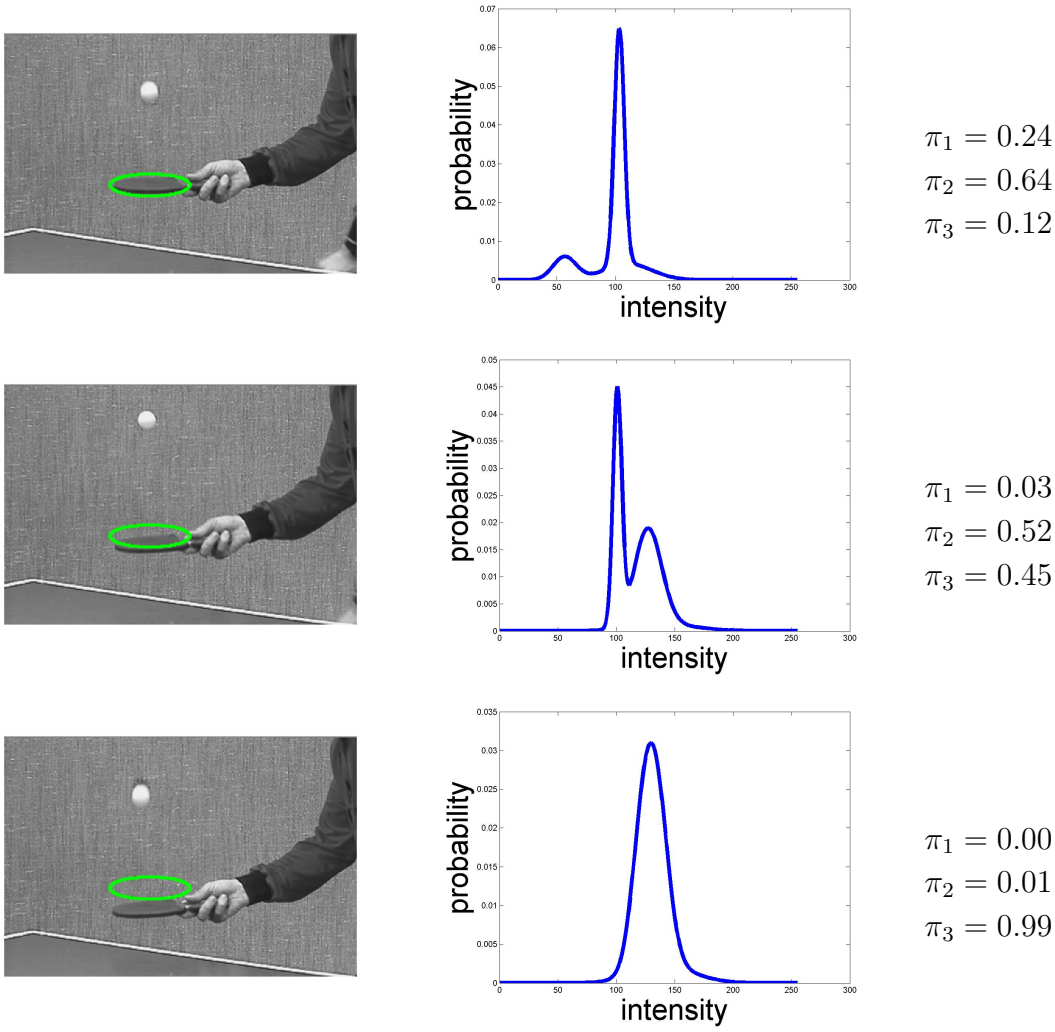


Figure 3.1: Variations of the GMM parameters during tracking. As the racket moves, the component that corresponds to the background (π_3) increases its proportion in the GMM due to the fact that more pixels belonging to background are inside the ellipse. On the other hand, components corresponding to the object (π_1 and π_2) reduce their responsibilities γ_{ni} in the model because pixels belonging to the object get out of the ellipse. Nevertheless, the means and variances of the model components remain unchanged because the object and background colors change smoothly.

where $D_{KL}(f_1|f_2)$ is the Kullback-Leibler divergence [14] between f_1 and f_2 is defined as

$$D_{KL}(f_1|f_2) = \frac{1}{2}\left[\log\left(\frac{\sigma_2^2}{\sigma_1^2}\right) + \frac{\sigma_1^2}{\sigma_2^2} + \frac{(\mu_2 - \mu_1)^2}{\sigma_2^2} - 1\right]. \tag{3.15}$$

The product $f_{uv}(\mathbf{y})d_{uv}$ represents the work needed to transfer a quantity of $f_{uv}(\mathbf{y})$ amount of solid to a distance d_{uv}. These transfers must be performed in such a way that the total work is minimum. Hence, $EMD(\mathbf{y})$ represents the work which must be produced to fill the holes of the second GMM using earth of the first GMM. We must notice that this fill is always possible because $\sum_{u=1}^{m} \pi_u^M = 1$ and $\sum_{v=1}^{m} \pi_v^C(\mathbf{y}) = 1$. In other words, the amount of earth in the hills is exactly equal to the amount needed by the holes to be fulfilled.

3.3 Target tracking

To locate the target, we must find the ellipse with center located at $\hat{\mathbf{y}}$ which is most similar to the ellipse of the model. In other words, a local minimum of $EMD(\mathbf{y})$ must be found:

$$\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmin}}[EMD(\mathbf{y})] \tag{3.16}$$

The computation of the EMD between the GMM of the target model and the GMM of the target candidate is computationally expensive if it is repeated for every possible location $\mathbf{y}$ in the target frame. In order to accelerate the procedure, following the principles proposed in [151, 152], we initialize the center $\mathbf{y}$ at the old center estimated in the previous frame, we calculate the gradient of the $EMD(\mathbf{y})$ with respect to $\mathbf{y}$ and use it to determine a new location. The same step is repeated until the value of $EMD(\mathbf{y})$ in the new location increases.

To solve the optimization problem (3.16), we have to calculate the derivative $\nabla_{\mathbf{y}} EMD(\mathbf{y})$ and choose the neighbor pixel in the direction of the derivative. Using the chain rule [151, 152] yields

$$\nabla_{\mathbf{y}} EMD(\mathbf{y}) = \sum_{v=1}^{m} \frac{\partial EMD(\mathbf{y})}{\partial \pi_v^C(\mathbf{y})} \nabla_{\mathbf{y}} \pi_v^C(\mathbf{y}) \tag{3.17}$$

The calculation of $\nabla_{\mathbf{y}} EMD(\mathbf{y})$ is described in [151, 152]. Here we summarize the key steps. The difference with [151, 152] is the usage of GMM instead of color signatures, which yields to a different formula for the computation of $\nabla_{\mathbf{y}} \pi_v^C(\mathbf{y})$ and consecutively for the computation of $\nabla_{\mathbf{y}} EMD(\mathbf{y})$. The formula for $\frac{\partial EMD(\mathbf{y})}{\partial \pi_v^C(\mathbf{y})}$ is the same as in [151, 152]. At first, equation (3.12) and the constraints in (3.13) are transformed to matrix-vector form. There are $m \times m$ variables $f_{uv}(\mathbf{y})$ and $m \times m$ constants d_{uv} stacked in vectors $\mathbf{f}(\mathbf{y})$ and $\mathbf{d}$ both of size $m^2 \times 1$. Taking together the first three constraints in (3.13), a matrix ($\mathbf{S}$ of size $m^2 + 1) \times m^2$ is created whose elements are 0 or 1. We also define the vector $\mathbf{b}(\mathbf{y}) = [(\pi^C(\mathbf{y}))^T, (\pi^M)^T, 1]^T$. Using these notations, equation (3.12) may be written as

$$EMD(\mathbf{y}) = \min_{\mathbf{f}} \mathbf{d}^T \mathbf{f}(\mathbf{y}) \tag{3.18}$$

and the constraints in (3.13) now become

$$\begin{aligned} &\mathbf{S}\mathbf{f}(\mathbf{y}) = \mathbf{b}(\mathbf{y}) \\ &\mathbf{f}(\mathbf{y}) \geq \mathbf{0} \end{aligned} \tag{3.19}$$

The above linear programming problem is solved by the simplex method [28]. Since the matrix $\mathbf{S}$ has rank $2m-1$, there are $2m-1$ basic variables, which will be denoted by $\mathbf{f}_B(\mathbf{y})$. Also there are $m^2 - 2m + 1$ non basic variables, which will be denoted by $\mathbf{f}_{NB}(\mathbf{y})$. Similarly, we denote by $\mathbf{d}_B$ and $\mathbf{d}_{NB}$ the elements of vector $\mathbf{d} = [\mathbf{d}_B \mathbf{d}_{NB}]^T$. Finally, $\mathbf{S}_B$ and $\mathbf{S}_{NB}$ are the columns of matrix $\mathbf{S}$ corresponding to the basic and non basic variables $\mathbf{f}_B(\mathbf{y})$ and $\mathbf{f}_{NB}(\mathbf{y})$ respectively. Equation (3.19) can now be written as

$$\begin{bmatrix} \mathbf{S}_B & \mathbf{S}_{NB} \end{bmatrix} \begin{bmatrix} \mathbf{f}_B(\mathbf{y}) \\ \mathbf{f}_{NB}(\mathbf{y}) \end{bmatrix} = \mathbf{b}(\mathbf{y}) \tag{3.20}$$

By performing sensitivity analysis the derivatives $\frac{\partial EMD(\mathbf{y})}{\partial \pi_v^C(\mathbf{y})}$ are computed as [151, 152]:

$$\frac{\partial EMD(\mathbf{y})}{\partial \pi_v^C(\mathbf{y})} = k_v - \sum_{\substack{j=1 \\ j \neq v}}^{m} k_j \frac{b_j}{\sum_{\substack{l=1 \\ l \neq v}}^{m} b_l} \tag{3.21}$$

where $k_v = \sum_{l=1}^{2m-1} (\mathbf{d}_B)_l (\mathbf{S}_B^{-1})_{lv}$.

To calculate $\nabla_{\mathbf{y}} \pi_v^C(\mathbf{y})$, which is the different part of our method with respect to the original DEMD paper [152], the derivative of (3.11) with respect to $\mathbf{y}$ must be computed. After some algebraic manipulation this leads to

$$\nabla_{\mathbf{y}} \pi_v^C(\mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} \left[B_n C_{n,v} + w_n(\mathbf{y}) A_{n,v} I_n'^C(\mathbf{y}) \right], \tag{3.22}$$

where

$$w_n(\mathbf{y}) = \frac{k(\|x_n - \mathbf{y}\|^2)}{\sum_{i=1}^{m} k(\|x_i - \mathbf{y}\|^2)}, \tag{3.23}$$

$$A_{n,v} = \sum_{i=1}^{m} \left[C_{n,v}\, C_{i,v} \left[\frac{I_n^C(\mathbf{y}) - \mu_v}{\sigma_v^2} - \frac{I_n^C(\mathbf{y}) - \mu_i}{\sigma_i^2} \right] \right], \tag{3.24}$$

$$B_n = \frac{2g(\|x_n - \mathbf{y}\|^2)(x_n - \mathbf{y}) \sum_{i=1}^{N} k(\|x_i - \mathbf{y}\|^2) - 2k(\|x_n - \mathbf{y}\|^2) \sum_{i=1}^{N} g(\|x_n - \mathbf{y}\|^2)(x_i - \mathbf{y})}{\left[\sum_{i=1}^{N} k(\|x_i - \mathbf{y}\|^2) \right]^2}, \tag{3.25}$$

$$C_{n,v} = \frac{\pi_v^M N(I_n^C(\mathbf{y}); \mu_v, \sigma_v)}{\sum_{j=1}^{m} \pi_j^M N(I_n^C(\mathbf{y}); \mu_j, \sigma_j)}. \tag{3.26}$$

In the above equations, $I_n'^C(\mathbf{y})$ is the spatial derivative of the intensity of the n^{th} pixel of the candidate object at location $\mathbf{y}$.

Substituting (3.21) and (3.22) in (3.17) yields the gradient of the EMD energy in closed form:

$$\nabla_{\mathbf{y}} EMD(\mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} \left[B_n \sum_{v=1}^{m} \left[\frac{\partial EMD(\mathbf{y})}{\partial \pi_v^C(\mathbf{y})} C_{n,v} \right] + w_n I_n'^C \sum_{v=1}^{m} \left[\frac{\partial EMD(\mathbf{y})}{\partial \pi_v^C(\mathbf{y})} A_{n,v} \right] \right] \tag{3.27}$$

The overall tracking algorithm is summarized in algorithm 1. We call this algorithm Mixture-based DEMD (MDEMD). After the computation of the derivative, one of the eight neighbor pixels is chosen. This pixel is the one that its center is most closer to the line that is defined by the gradient.

In order to handle target scaling changes, the main idea is to try different sizes for the ellipse and select the one with the minimum EMD. An extension to the notation must be used to introduce the time variable. At time t, the ellipse has axes h_x^t and h_y^t. The canonical coordinates x_n (used by algorithm 1) of the pixels inside the ellipse at time t, are computed by taking into account h_x^t and h_y^t. The current ellipse, representing the object, is obtained using algorithm 1 (MDEMD). Then, two new GMMs are constructed. The first GMM is trained using the pixels of a smaller ellipse (same center, smaller axes with respect to the current ellipse), while the

Algorithm 1 Differential EMD with GMM (MDEMD)

Input: The center $\mathbf{y}^{i-1}$ of the object in the previous frame $i-1$ and the GMM parameters of the object to be tracked.

Output: The center $\mathbf{y}^i$ of the object in the current frame i.

1 Initialization: Set $\mathbf{y}_0 = \mathbf{y}^{i-1}$ and evaluate $EMD(\mathbf{y}_0)$ using (3.12).

2 Compute $\nabla_{\mathbf{y}} EMD(\mathbf{y}_0)$ using (3.27).

3 Choose one of the 8 neighbors of $\mathbf{y}_0$ in the direction of the gradient $\nabla_{\mathbf{y}} EMD(\mathbf{y}_0)$. Let $\mathbf{y}_1$ be the coordinates of this pixel. Evaluate $EMD(\mathbf{y}_1)$ using (3.12).

4 If $EMD(\mathbf{y}_1) < EMD(\mathbf{y}_0)$ set $\mathbf{y}_0 \leftarrow \mathbf{y}_1$ and goto step 2.
Else return $\mathbf{y}^i \leftarrow \mathbf{y}_0$.

other is trained using the pixels of a bigger ellipse (same center, bigger axeswith respect to the current ellipse). If both of the new GMMs have greater EMD with the initial GMM compared to the EMD of the current ellipse with the initial GMM then the procedure stops. Otherwise the ellipse with the smaller EMD is selected and this procedure is repeated. This procedure is summarized in algorithm 2.

Algorithm 2 Scale adaptation on MDEMD

Input: The center $\mathbf{y}^{i-1}$ and the axes size h_x^{i-1} and h_y^{i-1} of the object in the previous frame $i-1$, and the GMM parameters of the object to be tracked.

Output: The center $\mathbf{y}^i$ and the axes size h_x^i and h_y^i of the object in the current frame i.

1 Initialization: Set $\mathbf{y}_0 = \mathbf{y}^{i-1}$, $h_x = h_x^{i-1}$ and $h_y = h_y^{i-1}$.

2 Call MDEMD with input $\mathbf{y}_0$ and axes size h_x and h_y. Store the center return by MDEMD to $\mathbf{y}^i$.

3 Compute $e = EMD(\mathbf{y}^i)$ using (3.27), using size axes h_x and h_y.

4 Compute $e^+ = EMD(\mathbf{y}^i)$ using (3.27), using size axes $1.1h_x$ and $1.1h_y$.

5 Compute $e^- = EMD(\mathbf{y}^i)$ using (3.27), using size axes $0.9h_x$ and $0.9h_y$.

6 if $e^+ < e^-$ and $e^+ < e$, set $\mathbf{y}_0 = \mathbf{y}^i$ and axes size $h_x = 1.1h_x$ and $h_y = 1.1h_x$. Go to step 2.

7 if $e^- < e^+$ and $e^- < e$, set $\mathbf{y}_0 = \mathbf{y}^i$ and axes size $h_x = 0.9h_x$ and $h_y = 0.9h_x$. Go to step 2.

8 return $\mathbf{y}^i$ and the axes size $h_x^i = h_x$ and $h_y^i = h_y$

3.4 Robustness to Occlusions

In this section we combine the differential MEMD algorithm with a Kalman filter to handle occlusions.

3.4.1 Background on Kalman filter

In general, we assume that there is a linear process governed by an unknown inner state producing a set of measurements. More specifically, there is a discrete time system and its state at time n is given by vector $\mathbf{x}_n$. The state in the next time step $n+1$ is given by

$$\mathbf{x}_{n+1} = \mathbf{F}_n \mathbf{x}_n + \mathbf{w}_n \tag{3.28}$$

where $\mathbf{F}_n$ is the transition matrix from state $\mathbf{x}_n$ to $\mathbf{x}_{n+1}$ and $\mathbf{w}_n$ is white Gaussian noise with zero mean and covariance matrix $\mathbf{Q}_n$.

The measurement vector $\mathbf{z}_n$ is given by

$$\mathbf{z}_n = \mathbf{H}_n \mathbf{x}_n + \mathbf{v}_n \tag{3.29}$$

where $\mathbf{H}_n$ is the measurement matrix and $\mathbf{v}_n$ is white Gaussian noise with zero mean and covariance matrix $\mathbf{R}_n$. In equation (3.29), the measurement $\mathbf{z}_n$ depends only on the current state $\mathbf{x}_n$ and the noise vector $\mathbf{v}_n$ is independent of the noise $\mathbf{w}_n$.

The Kalman filter approach computes the minimum mean-square error estimate of the state $\mathbf{x}_k$ given the measurements $\mathbf{z}_1, \ldots, \mathbf{z}_k$. The solution is obtained using a recursive procedure [106].

3.4.2 Differential EMD with GMM and Kalman filter

The main idea behind the combined approach is to find the position of the object with algorithm 1 (measurement) and forward it to Kalman filter to obtain the current position of the object (estimation). The transition matrix $\mathbf{F}_n$ is not known in the beginning and is estimated by the algorithm.

We assume that the object is described by its center coordinates (x, y) and the axes (h_x, h_y) of the ellipse around it and that the size of the ellipse does not change through time. The state vector $\mathbf{x}_n = [x_n, y_n, 1]^T$ is the position of the center in the image in homogenous coordinates (x_n and y_n are the horizontal and vertical coordinate respectively) and its position varies over time as described in equation (3.28). The matrix $\mathbf{F}_n$ is defined as:

$$\mathbf{F}_n = \begin{bmatrix} 1 & 0 & dx_n \\ 0 & 1 & dy_n \\ 0 & 0 & 1 \end{bmatrix}$$

where dx_n dy_n are the horizontal and vertical translations of the object's center. Parameters dx_n dy_n are not constant in time, but they are computed dynamically as it will be explained below. The noise vector $\mathbf{w}_n = [w_{n_x}, w_{n_y}, 1]^T$ has covariance matrix $\mathbf{Q}$.

We employ algorithm 1 to obtain the measurement vector $\mathbf{z}_n = [x'_n, y'_n]^T$ where x'_n and y'_n are the horizontal and vertical coordinates of the ellipse center. In general, these measurements differ from the state variables x_n and y_n of vector $\mathbf{x}_n$ due to the presence of noise $\mathbf{v}_n$. The relation between measurement $\mathbf{z}_n$ and state $\mathbf{x}_n$ is given by (3.29), where

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

and the measurement noise $\mathbf{v}_n = [v_{n_x}, v_{n_y}]^T$ has covariance matrix $\mathbf{R}$.

The only problem that remains to be solved is the automatic evaluation of dx_n and dy_n. Using algorithm 1 we obtain:

- the measurement $\mathbf{z}_n$,
- the distance between the mixture components of the model of the target and the target candidate.

The main idea is to use the computed distance to determine if the object was found or not. This provides a quality measure of the current estimate of the object. If the distance is small, then we have a good chance that the object's center is near the predicted center. If this distance is large, then, the target is lost. This distance is expressed as a normalized coefficient:

$$a(\mathbf{y}) = \exp(-cEMD(\mathbf{y})) \tag{3.30}$$

where $EMD(\mathbf{y})$ is given by (3.12) and it is the EMD distance between the source and target GMM at position $\mathbf{y}$ and c is a constant. From experimental results, we found that the value of c does not affect the correctness of the algorithm significant. The value for c used for the experiments in this chapter is equal to 10. The a is an estimation of how confident we are that the object is found. If we are not sure the object is correctly located, then we follow the previous movement of the object, assuming that occlusion took place. On the other hand, when we are convinced that the object is inside the ellipse, we update our knowledge about the object's movement. Relying on the value of a in (3.30), parameter $\mathbf{d}_n = [dx_n, dy_n]^T$ is automatically updated by:

$$\mathbf{d}_{n+1} = (1 - a(\hat{\mathbf{x}}_n))\mathbf{d}_n + a(\hat{\mathbf{x}}_n)(\hat{\mathbf{x}}_n - \hat{\mathbf{x}}_{n-1}) \tag{3.31}$$

where $\hat{\mathbf{x}}_n$ is the vector containing the estimated values of the horizontal and vertical coordinates of the ellipse center at time n. In view of (3.31), the estimate $\hat{\mathbf{x}}_n$ contributes to the updates of the displacement $\mathbf{d}_n$ only when the current estimate resembles the source object model, that is when $a(\hat{\mathbf{x}}_n) \to 1$. On the other hand when $a(\hat{\mathbf{x}}_n) \to \exp(-c)$, the displacements included in the state matrix $\mathbf{F}_n$ remain nearly unchanged, as they were in step $n-1$, assuming that the object is occluded. This process has the advantage that the matrix $\mathbf{F}_n$ incorporating information on the object movement can be updated by the tracking algorithm.

Algorithm 3 summarizes the differential MEMD tracking algorithm with Kalman filtering. Note that step 4 uses a tracking algorithm to estimate the position of the object. This algorithm can be also mean shift or DEMD.

Algorithm 3 Differential EMD with GMM and Kalman filter

1 Initialization: $\hat{\mathbf{x}}_0$ = initial object location:

$$\mathbf{P}_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{Q} = \begin{bmatrix} h_x & 0 & 0 \\ 0 & h_y & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix} h_x & 0 & 0 \\ 0 & h_y & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{F}_0 = \mathbf{I}_{3\times 3}.$$

2 Compute initial GMM_0 in the first frame as described in section 3.2.1.

3 Prediction:

$$\hat{\mathbf{x}}_n^- = \mathbf{F}_n \hat{\mathbf{x}}_{n-1},$$
$$\mathbf{P}_n^- = \mathbf{F}_n \mathbf{P}_{n-1} \mathbf{F}_n^T + \mathbf{Q},$$
$$\mathbf{G}_n = \mathbf{P}_n^- \mathbf{H}_n^T [\mathbf{H}_n \mathbf{P}_n^- \mathbf{H}_n^T + \mathbf{R}]^{-1}.$$

4 Measurement: Compute the new center ($\mathbf{z}_n$), the new GMM and the distance between GMM and GMM_0 using MDEMD (Algorithm 1).

5 Estimation:

$$\hat{\mathbf{x}}_n = \hat{\mathbf{x}}_n^- + \mathbf{G}_n(\mathbf{z}_n - \mathbf{H}_n \hat{\mathbf{x}}_n^-),$$
$$\mathbf{P}_n = (\mathbf{I} - \mathbf{G}_n \mathbf{H}_n)\mathbf{P}_n^-.$$

The output $\hat{\mathbf{x}}_n$ is the object's new location.

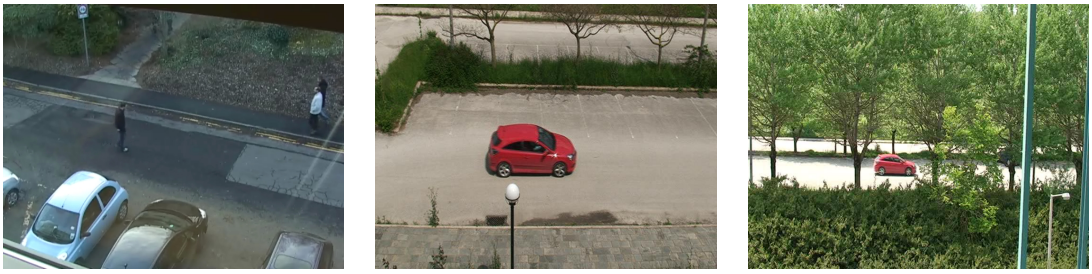
6 Update the elements of $\mathbf{F}_n$ using (3.31).
Goto the Prediction step for the next iteration.

3.5 Experimental Results

To evaluate the proposed algorithm MDEMD, we have performed comparisons with the mean shift algorithm [30] and the standard DEMD tracking method [151, 152]. In the same context, we have also evaluated the Kalman based DEMD tracker (MDEMD-K). The proposed adapted Kalman filter is also combined with the mean shift (MS-K) and the DEMD algorithm (DEMD-K) in order to have a complete overview of its behavior. Six test sequences were employed in the evaluation consist of outdoor testing situations. The length of the sequences varies between 80 and 400 frames with one object to be tracked in every image sequence. Representative frames are shown in figure 3.2. Each object is described by its center, in image coordinates, and the size of the ellipse around it (the ellipse has axes parallel to the image axes). The ground truth in every image was determined manually. All the algorithms assume knowledge of the objects position only in the first frame. The object position is estimated in the following frames, using the corresponding algorithm. In all tests, the number of histogram bins for the mean shift is 16 and for the standard DEMD algorithms was 8 and 16 as suggested in [30, 151, 152]. The number of the components in the proposed GMM based tracker was selected to be 3, 4 and 6.

The respective number of components may not be suitable for every problem and it depends on the complexity of the object to be tracked. As a rule of thumb, the number of components is equal to the colors of the object plus the colors of the background. For instance, by using three components, we assume that two components belong to the object (e.g. in *Seq5* a red car with black windows) and one component corresponds to the background (e.g. the gray road). All the examples were carried out with a core 2 Duo 1.66 GHz processor with 2GB RAM under Matlab.

Seq1 (129 frames) *Seq2* (80 frames) *Seq3* (400 frames)



Seq4 (221 frames) *Seq5* (106 frames) *Seq6* (285 frames)

Figure 3.2: Representative frames of the image sequences used in the experiments.

Sequences *Seq1* and *Seq2* show a person walking from left to right in an underground station (PETS 2006 workshop). *Seq3* show a car moving (PETS 2001 workshop). *Seq4* shows a person walking in an outdoor environment (BEHAVE dataset, University of Edinburgh, School of Informatics, http://groups.inf.ed.ac.uk/vision/BEHAVEDATA/INTERACTIONS/). The last two sequences (*Seq5* and *Seq6*) are created by our group. They show a red car moving from left to right without occlusion (*Seq5*) and with occlusion (*Seq6*).

To estimate the accuracy of the compared algorithms we measure the normalized Euclidean distance between the true center ($\mathbf{c}$) of the object (as determined by the ground truth) and the estimated location of the ellipse center ($\hat{\mathbf{c}}$). The normalized Euclidean distance is defined by

$$NED\left(\mathbf{c},\hat{\mathbf{c}}\right)=\sqrt{\left(\frac{\mathbf{c}_x-\hat{\mathbf{c}}_x}{h_x}\right)^2+\left(\frac{\mathbf{c}_y-\hat{\mathbf{c}}_y}{h_y}\right)^2} \tag{3.32}$$

where we recall that h_x and h_y are the ellipse dimensions. This implies that if $NED\left(\mathbf{c},\hat{\mathbf{c}}\right)<1$, then the estimated ellipse center $\hat{\mathbf{c}}$ is inside the ground truth ellipse. By these means, the image size and the ellipse dimensions do not influence the relative distance between ($\mathbf{c}$) and ($\hat{\mathbf{c}}$).

Table 3.1: Tracking accuracy. The average normalized Euclidean distance between the true object center and the estimated object center is presented for the compared methods.

Sequence	*Seq1*	*Seq2*	*Seq3*	*Seq4*	*Seq5*	*Seq6*
Frames	129	80	400	221	106	285
MS (16 bins)	0.44	0.38	0.46	4.01	0.07	2.08
DEMD (8 bins)	**0.16**	0.17	0.39	4.09	**0.05**	2.56
DEMD (16 bins)	0.17	0.16	**0.36**	0.41	**0.05**	2.50
MDEMD (3 components)	0.20	**0.15**	0.38	**0.38**	**0.05**	2.14
MDEMD (4 components)	0.23	**0.15**	0.40	0.51	0.06	1.98
MDEMD (6 components)	0.97	0.27	0.48	4.03	**0.05**	2.11
MS-K (16 bins)	0.55	0.47	0.48	5.47	0.09	0.72
DEMD-K (16 bins)	0.19	0.22	0.42	0.44	0.06	0.56
MDEMD-K (4 components)	0.25	0.25	0.39	0.94	**0.05**	**0.48**

Table 3.1 summarizes the comparisons in terms of tracking accuracy. As it can be seen, the proposed algorithm has high accuracy (the computed center is inside the ellipse of the actual object). The standard DEMD algorithm confirms its efficiency with respect to mean shift, as it is presented in [151, 152]. Moreover, the proposed algorithm based on GMM is favorably compared with mean shift (table 3.1). Generally, all of the compered methods present high performances with little differences. DEMD performs better at *Seq1* and *Seq3*, MDEMD is more accurate at *Seq2* and *Seq4* and all of the methods achieve similar results at *Seq5*. Also, the employment of Kalman filter has the ability to track objects when occlusions occur, while the other methods fail. When $NED > 1$ the target is lost, which is the case for the methods not using the adapted Kalman filter (*Seq6*). Moreover, MDEMD-K with four components provides highly better accuracy with respect to MS-K and DEMD-K.

As it can also be observed in table 3.1 when DEMD employs relatively few bins its results deteriorate in comparison with configurations using larger number of bins. The opposite stands for the proposed MDEMD. The is also confirmed by *Seq4* (table 3.1) where DEMD with 8 bins totally misses the target (this is also true for MDEMD with 6 components). This is a relative difficult sequence as an indoor camera records a person moving from left to right outside. There is a glass between the camera and the moving person. This sequence has particular difficulties such as reflections due to the glass and illumination changes between frames. These difficulties are responsible for the failure of mean shift (MS) even when it is jointly applied with the adapted Kalman filter (MS-K).

The comparison of the three algorithms employing the Kalman filter (last three rows of table 3.1) reveals that DEMD and MDEMD show similar accuracies (in any case they are better than mean shift).

In table 3.2, the execution times (sec/frame) of the compared methods are shown. The proposed GMM based methods (MDEMD and MDEMD-K) are significantly faster with respect

Table 3.2: Average execution times for the compared methods (sec/frame).

Sequence	*Seq1*	*Seq2*	*Seq3*	*Seq4*	*Seq5*	*Seq6*
Frames	129	80	400	221	106	285
MS (16 bins)	1.64	2.91	0.66	0.55	8.05	0.36
DEMD (8 bins)	1.14	1.88	0.58	0.48	2.30	0.40
DEMD (16 bins)	2.42	3.82	1.39	1.42	4.67	2.45
MDEMD (3 components)	0.53	0.89	**0.23**	0.21	1.81	0.20
MDEMD (4 components)	**0.52**	0.77	0.24	0.20	1.71	0.22
MDEMD (6 components)	0.55	0.80	0.27	0.23	1.66	**0.17**
MS-K (16 bins)	2.19	3.85	0.68	0.64	10.53	0.38
DEMD-K (16 bins)	3.48	4.86	1.32	1.68	6.38	1.04
MDEMD-K (4 components)	0.58	**0.49**	0.24	**0.16**	**1.56**	0.28

to DEMD (table 3.2). This occurs because the number of GMM components is less than the number of histogram bins. Therefore, the integration of mixtures in DEMD tracking preserves the tracking accuracy and simultaneously reduces the computational complexity.

Table 3.3: Average number of iterations per frame for the compared methods.

Sequence	*Seq1*	*Seq2*	*Seq3*	*Seq4*	*Seq5*	*Seq6*
Frames	129	80	400	221	106	285
MS (16 bins)	2.98	3.86	**1.00**	1.90	2.17	**1.10**
DEMD (8 bins)	4.76	4.90	2.40	3.46	3.36	2.78
DEMD (16 bins)	4.71	4.82	2.56	3.77	3.46	2.77
MDEMD (3 components)	2.43	2.62	1.23	2.34	1.88	1.51
MDEMD (4 components)	2.27	2.22	1.27	2.10	1.75	1.39
MDEMD (6 components)	**2.05**	2.05	1.17	1.95	1.67	1.29
MS-K (16 bins)	3.86	5.31	1.03	2.37	2.93	1.34
DEMD-K (16 bins)	6.57	7.15	2.46	4.87	4.54	2.89
MDEMD-K (4 components)	2.58	**1.48**	1.54	**1.66**	**1.44**	1.30

In table 3.3 we present the mean number of iterations needed for each method to converge in a single frame. The values of this table are independent from the machine used for the experiments and better highlight the rate of convergence of the various algorithms. As it can be seen the new MDEMD algorithm converges in fewer iterations than the standard DEMD. Let us also notice that DEMD converges in approximately the same number of iterations with 8 and 16 bins. However the average execution time per frame (as depicted in table 3.2) is almost doubled when using 16 bins. This is due to the augmented complexity in the optimization algorithm. Furthermore, the application of the adaptive Kalman filter reduces the number of iterations if

Figure 3.3: *Seq5*. Representative frames with the estimates of the ellipse for the compared algorithms.

the motion model is correctly estimated.

A representative example of *Seq5* is shown in figure 3.3. An example with occlusion is presented in figure 3.4 and 3.5 where the red car is masked by the trees. All of the compared algorithms successfully track the object until it reaches the trees. However, only the algorithms employing the proposed adaptive Kalman filter achieve in predicting its motion.

3.6 Conclusions

In this chapter, we have proposed a method for visual object tracking relying on modeling the appearance of the object in the first frame using a Gaussian mixture. The EM algorithm is applied to compute the initial GMM. In the following frames, the location of the object is estimated in a differential framework by the direction of the gradient of the EMD with respect to the bi-dimensional image space [151, 152]. This gradient is computed in closed form and the key issue in this computation, is the change in the responsibilities of the GMM components between adjacent frames. In these images the EM is not applied, because means and variances do not change. Therefore, the algorithm is significantly faster than the standard DEMD tracker [151, 152] while retaining the same high tracking accuracy. Also, the proposed algorithm is combined with a Kalman filter to efficiently handle occlusions. The prediction of the GMM-based DEMD tracker is considered as the observation of a Kalman filter whose state parameters are automatically determined based on recent motion history. By these means, partial or total occlusions may be successfully addressed.

Figure 3.4: *Seq6*. Representative frames with the estimates of the ellipse for the compared algorithms.

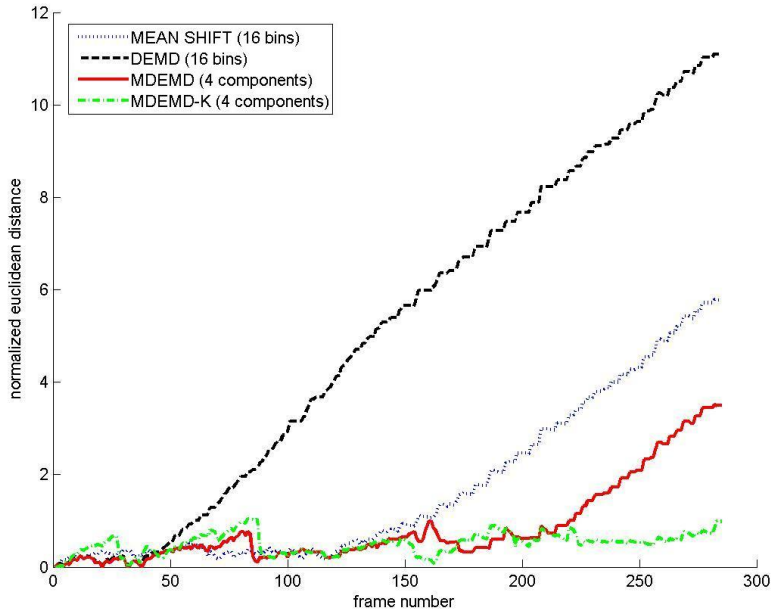


Figure 3.5: *Seq6*. The normalized Euclidean distances between the ground truth and the estimates of the ellipse center for the compared algorithms.

CHAPTER 4

VISUAL TRACKING UNDER ABRUPT ILLUMINATION CHANGES

4.1 Introduction

In this chapter, we propose a variant of the mean shift algorithm [30], in order to make the tracking procedure more robust to uniform or nonuniform abrupt light changes, such as flicker, light switch or shadow casts. In these cases, as the whole image becomes darker or brighter, the histogram of the target is shifted with respect to the initial histogram and the affinity between them would be close to zero, thus, making the mean shift algorithm to miss the object. In our approach, we propose to estimate the initial histogram of the target in the first frame by a GMM and consider this mixture as a weighting function for the calculation of the histogram in the next frames. By these means, all of the mixture components contribute to the value of a specific bin and the histogram becomes smoother as its original values are diffused to neighboring bins.

In the remaining of the chapter, the mean shift method is described in section 4.2, the proposed evaluation of histogram is described in section 4.3, experimental results are presented in section 4.4 and the conclusions are drawn is section 4.5.

4.2 Mean shift algorithm

The mean shift [30] is a target representation and localization algorithm trying to locate the object by finding the local maximum of a function. Here we give a brief review. The object target pdf is approximated by a histogram of m bins $\hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1\ldots m}$, $\sum_{u=1}^{m} \hat{q}_u = 1$, with $\hat{q}_u$ being the u-th bin. To form the histogram, only the pixels inside an ellipse surrounding the object are taken into account. The center of the ellipse is assumed to be at the origin of the axes. Due to the fact that the ellipse contains both object pixels and background pixels, a kernel with profile $k(x)$, $k : [0, \infty) \to \Re$ is applied to every pixel to make pixels near the center of the ellipse to be considered more important. To reduce the influence of an eventual difference in the length of the ellipse axes, the pixel locations are normalized by dividing the pixel's coordinates with the ellipse's semi-axes lengths h_x and h_y. Let $\{\mathbf{x}_i^*\}_{i=1\ldots n}$ be the normalized pixel's spatial location. The u-th histogram bin is given by:

$$\hat{q}_u = C \sum_{i=1}^{n} k(\|\mathbf{x}_i^*\|^2)\delta[b(\mathbf{x}_i^*) - u] \tag{4.1}$$

where $b : \Re^2 \to \{1 \ldots m\}$ associates each pixel with each bin in the quantized feature space, δ is the Kronecker delta function and C is a normalization factor such as $\sum_{u=1}^{m} \hat{q}_u = 1$.

In the next image, the object candidate is inside the same ellipse with its center at the normalized spatial location $\mathbf{y}$. Let $\{\mathbf{x}_i\}_{1\ldots n}$ be the normalized pixel coordinates inside the target candidate ellipse. The pdf of the target candidate is also approximated by an m-bin histogram $\hat{\mathbf{p}}(\mathbf{y}) = \{\hat{p}_u(\mathbf{y})\}_{u=1\ldots m}$, $\sum_{u=1}^{m} \hat{p}_u(\mathbf{y}) = 1$, with each histogram bin given by

$$\hat{p}_u(\mathbf{y}) = D \sum_{i=1}^{n} k\left(\|\mathbf{y} - \mathbf{x}_i\|^2\right) \delta[b(\mathbf{x}_i) - u] \tag{4.2}$$

where D is a normalization factor such as $\sum_{u=1}^{m} \hat{p}_u(\mathbf{y}) = 1$.

The distance between $\hat{\mathbf{q}}$ and $\hat{\mathbf{p}}(\mathbf{y})$ is defined as:

$$d(\mathbf{y}) = \sqrt{1 - \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}]} \tag{4.3}$$

where

$$\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y})\hat{q}_u} \tag{4.4}$$

is the similarity function between $\hat{\mathbf{q}}$ and $\hat{\mathbf{p}}(\mathbf{y})$ (Bhattacharyya coefficient).

To locate the object correctly in the image, the distance in (4.3) must be minimized, which is equivalent to maximize (4.4). The ellipse center is initialized at a location $\hat{\mathbf{y}}_0$ which is the ellipse center in the previous image frame. The probabilities $\{\hat{p}_u(\hat{\mathbf{y}}_0)\}_{u=1\ldots m}$ are computed and using linear Taylor approximation of (4.4) around these values:

$$\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] \approx \frac{1}{2} \sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{\mathbf{y}}_0)\hat{q}_u} + \frac{D}{2} \sum_{u=1}^{n} w_i k\left(\|\mathbf{y} - \mathbf{x}_i\|^2\right), \tag{4.5}$$

where

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}} \delta[b(\mathbf{x}_i) - u]. \tag{4.6}$$

As the first term of (4.5) is independent of $\mathbf{y}$, the second term of (4.5) must be maximized. The maximization of this term may be accomplished by employing the mean shift algorithm [30], which yields the following update:

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n} \mathbf{x}_i w_i g\left(\|\hat{\mathbf{y}}_0 - \mathbf{x}_i\|^2\right)}{\sum_{i=1}^{n} w_i g\left(\|\hat{\mathbf{y}}_0 - \mathbf{x}_i\|^2\right)}, \tag{4.7}$$

where $g(x) = -k'(x)$. The complete algorithm [30] is summarized in algorithm 4.

Algorithm 4 Mean shift tracking procedure

Input: The target model $\{\hat{q}_u\}_{u=1\ldots m}$ and its location $\hat{\mathbf{y}}_0$ in the previous frame.

1. Initialize the center of the ellipse in the current frame at $\hat{\mathbf{y}}_0$, compute $\{\hat{p}_u(\hat{\mathbf{y}}_0)\}_{u=1\ldots m}$ using (4.2).

2. Compute the weights $\{w_i\}_{i=1\ldots n}$ according to (4.6).

3. Compute the next location of the target candidate according to (4.7).

4. If $\|\hat{\mathbf{y}}_1 - \hat{\mathbf{y}}_0\| < \epsilon$ Stop.
Otherwise set $\hat{\mathbf{y}}_0 \leftarrow \hat{\mathbf{y}}_1$ and go to Step 2.

4.3 Target modeling by a GMM

If global, uniform or nonuniform, illumination changes take place, then the whole histogram $\hat{p}_u(\mathbf{y})$ in (4.2) will be (uniformly or not) shifted with respect to the initial histogram $\hat{q}_u$ in (4.1). For the sake of clarity, this issue is illustrated by a simple example in figure 4.1, where the initial histogram is shown in fig. 4.1(a) and the histogram of the target in the next frame (under abrupt illumination change) is shown in fig. 4.1(b). Notice that, ideally, this should be the histogram corresponding to the maximum of (4.3). However, by simple inspection, this distance is close to zero and the algorithm would respond with an erroneous image location for the target due to the influence of this distance in the computation of the weights in (4.6). Although this issue could be overcome for simple global uniform illumination changes (e.g. by subtracting the mean image value) the problem becomes more intricate if the involved changes in lighting conditions are highly non uniform. In figure 4.1(c), the GMM representing the density of the target in fig.4.1(a) is shown.

In order to estimate the GMM parameters we define the log-likelihood function of the color

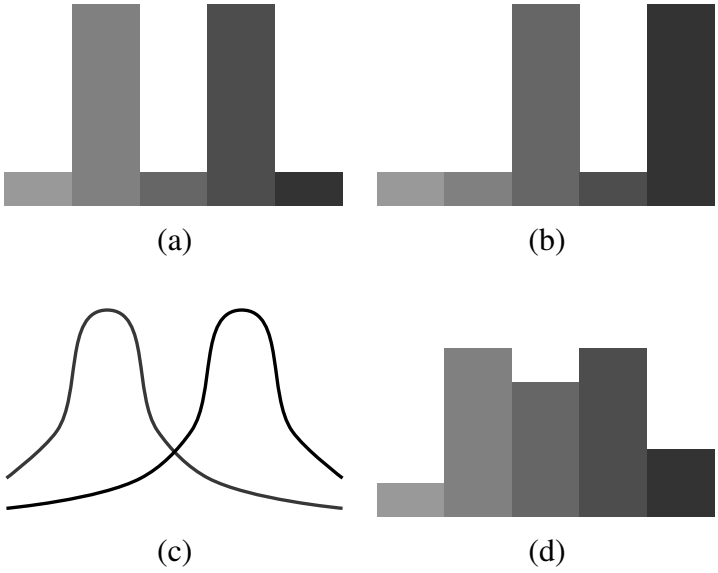


Figure 4.1: a) The histogram of the target in the initial image. b) The histogram of the target in the next image is shifted due to an abrupt illumination change. c) The GMM of the target in the initial image. d) The resulting smooth histogram using (4.14).

of pixels inside an ellipse:

$$L(\boldsymbol{I};\boldsymbol{\pi},\boldsymbol{\mu},\boldsymbol{\Sigma}) = \sum_{i=1}^{n} \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{I}_i;\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k) \tag{4.8}$$

where $\boldsymbol{I} = \{\boldsymbol{I}_i\}_{i=1,\ldots,n}$ denote the color of every pixel, $\boldsymbol{\pi} = \{\pi_k\}_{k=1,\ldots,K}$ are the mixing proportions, $\boldsymbol{\mu} = \{\boldsymbol{\mu}_k\}_{k=1,\ldots,K}$ are the mean vectors, $\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_k\}_{k=1,\ldots,K}$ are the covariance matrices and K denotes the number of the GMM components. The estimation of the GMM parameters is achieved through the EM algorithm [14]:

E-Step:

$$z_{k,i} = \frac{\pi_k \mathcal{N}(\boldsymbol{I}_i|\boldsymbol{\mu}_k,\boldsymbol{\Sigma}_k)}{\sum_{l=1}^{K} \pi_l \mathcal{N}(\boldsymbol{I}_i|\boldsymbol{\mu}_l,\boldsymbol{\Sigma}_l)}. \tag{4.9}$$

M-Step:

$$\pi_k = \frac{\sum_{i=1}^{n} z_{k,i}}{n}, \tag{4.10}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{i=1}^{n} z_{k,i} \boldsymbol{I}_i, \tag{4.11}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{i=1}^{n} z_{k,i} \left(\boldsymbol{I}_i - \boldsymbol{\mu}_k\right) \left(\boldsymbol{I}_i - \boldsymbol{\mu}_k\right)^T, \tag{4.12}$$

The EM algorithm is employed in the first frame in order to estimate the GMM parameters. Having computed $\boldsymbol{\pi} = \{\pi_k\}_{k=1,\ldots,K}$, $\boldsymbol{\mu} = \{\boldsymbol{\mu}_k\}_{k=1,\ldots,K}$ and $\boldsymbol{\Sigma} = \{\boldsymbol{\Sigma}_k\}_{k=1,\ldots,K}$ we can estimate the equivalent histograms' bins in (4.1) and (4.2).

We assume that every bin is computed by the mixture of all components and the u-th histogram bin in (4.1) is now given by:

$$\hat{q}_u = C \sum_{i=1}^{n} k(\|\mathbf{x}_i^*\|^2) \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{I}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{4.13}$$

Equivalently, in the next image the u-th histogram bin is given by:

$$\hat{p}_u(\mathbf{y}) = D \sum_{i=1}^{n} k\left(\|\mathbf{y} - \mathbf{x}_i\|^2\right) \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{I}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{4.14}$$

which corresponds to the toy example in fig. 4.1(d). Notice that the transitions between bins are now smoother and the basin of attraction of the smoother histogram may be large enough to capture the reference histogram in fig. 4.1(a). The reference histogram looks now more similar to the histogram at the ideal target location.

By following the same reasoning as in section 4.2, we end up in the same update equation for y as in (4.7). However the weights w_i are given by:

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}} \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{I}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k). \tag{4.15}$$

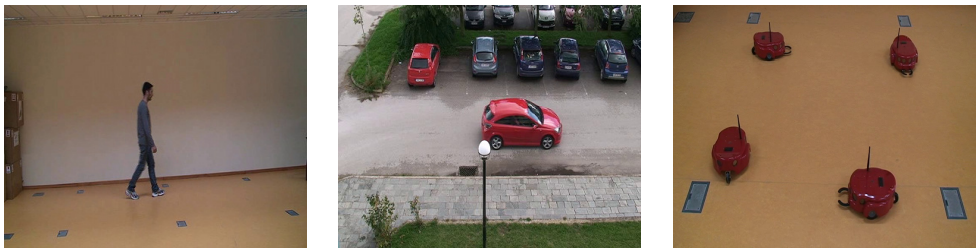
Therefore, the tracking procedure is the same as described in algorithm 4, but we use (4.13), (4.14) and (4.15) instead of (4.1), (4.2) and (4.6) respectively.

4.4 Experimental results

The evaluation of the proposed tracking algorithm was performed using three datasets (Fig. 4.2). In *Seq1*, a man is walking from left to right, in *Seq2*, a car is moving from left to right and in *Seq3*, four robots are moving at different directions. The ground truth for these image sequences was manually determined. We compared our approach (referred as MSGMM) with the standard mean shift algorithm [30]. We use RGB images where and the number of the histogram bins is set to 16 in each channel, resulting to 16^3 bins totally.

We evaluated the performance of the tracking algorithm both in terms of position error and execution time. We define the position error as the average Euclidian distance between the ground truth's ellipse center and the ellipse estimated by the tracking algorithm (in normalized coordinates). The execution time is defined as the average time (seconds) needed per frame by the tracking procedure.

Firstly, we evaluate the proposed method for different numbers of the GMM components K (Table 4.1). Comparing the position error for $K = 1, \ldots, 5$, we observe that the best results are obtained for $K = 2$. This happens due to the fact that the targets have relatively few colors. In terms of average time per frame, the fewer the components of the GMM are, the less execution time is needed. Here, we must point out that all these variants are executed in real time. For this



Seq1 (300 frames) | *Seq1* (700 frames) | *Seq2* (250 frames)

Figure 4.2: Representative frames of the datasets used in the experiments.

Table 4.1: The performance of the proposed method for different GMM components number (K) in terms of average position error and average execution time.

Sequece	$K=2$	$K=3$	$K=4$	$K=5$
	Position Error			
Seq1	**0.216**	0.226	0.242	0.270
Seq2	**0.461**	0.487	0.545	0.517
Sqe3	**0.342**	0.392	0.390	0.401
	Seconds/Frame			
Seq1	**0.012**	0.014	0.019	0.021
Seq2	**0.013**	0.017	0.019	0.024
Sqe3	**0.010**	0.012	0.014	0.016

set of experiments, we choose $K = 2$ for comparison with the standard mean shift in flicker conditions.

In order to evaluate the proposed method during illumination changes, we used six more sequences that are generated from sequences *Seq1*, *Seq2* and *Seq3*. The sequences with the subscript *a* are produced from the initial sequences by keeping the first frame the same and making the rest of the frames significantly brighter. The sequences with the subscript *b* are produced from the initial sequences by making the odd-numbered frames brighter and the even-numbered frames dimmer (Fig. 4.3).

In Table 4.2, the comparative results between the standard mean shift algorithm and the proposed method with $K = 2$ GMM components are given. In normal conditions (*Seq1*, *Seq2* and *Seq3*), mean shift provides the best results in terms of position error in two out of three datasets. In terms of execution time, the MSGMM algorithm needs approximately twice the time needed by mean shift in all nine sequences. This results from the fact that computations involving the evaluation of the normal distribution for every pixel are performed.

When the lighting conditions change, MSGMM clearly outperforms mean shift. Especially, in $Seq3_a$ and $Seq3_b$, mean shift fails to track the object from the beginning. On the other hand,

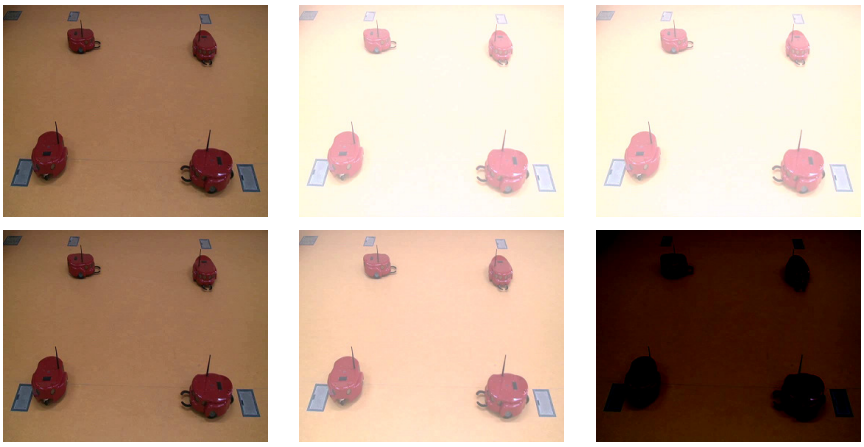


Figure 4.3: Top row: the first frames of $Seq3_a$. Bottom row: the first frames of $Seq3_b$.

Table 4.2: The performance of the compared methods (mean shift and MSGMM with $K = 2$) in terms of average position error and average size error.

	Position Error		Seconds/Frame	
Sequece	Mean shift	MSGMM	Mean shift	MSGMM
Seq1	0.264	**0.216**	**0.007**	0.012
$Seq1_a$	0.572	**0.544**	**0.006**	0.013
$Seq1_b$	0.904	**0.875**	**0.012**	0.026
Seq2	**0.289**	0.461	**0.005**	0.013
$Seq2_a$	0.608	**0.251**	**0.006**	0.018
$Seq2_b$	0.726	**0.566**	**0.018**	0.037
Sqe3	**0.155**	0.342	**0.005**	0.010
$Seq3_a$	-	**0.919**	**0.005**	0.015
$Seq3_b$	-	**1.029**	**0.011**	0.022

MSGMM successfully tracks the object with a position error of 0.919 for $Seq3_a$ and 1.029 for $Seq3_b$. The error around 1 means that the estimated center of the target is around the edge of the ellipse representing the object, but there is still common area between the ground truth ellipse and the estimated ellipse.

4.5 Conclusions

In this chapter, we modified the mean shift algorithm in order to make the tracking procedure more robust to illumination changes. We used Gaussian mixture model for the evaluation of histogram bins. This modification also affects the weights of every pixel during the tracking process. As shown by the experimental results, this approach can successfully track objects

when the light conditions change dramatically.

CHAPTER 5

VISUAL TRACKING USING SPATIALLY WEIGHTED LIKELIHOOD OF GAUSSIAN MIXTURES

5.1 Introduction

In this chapter, we address both problems of feature dimensionality and changes in model appearance. We present a tracking algorithm relying on a probabilistic representation of the object to be tracked and its subsequent localization in the image sequence. It is assumed that the appearance of the target may be described by a GMM instead of a histogram or histogram signatures, as it is the case in [30, 152, 155]. Using a GMM instead of a histogram has certain advantages. At first, GMM provide a more compact representation of the feature space as a few parameters are generally sufficient to model the color distribution of the target. At second, if high dimensional features are employed the bins of a standard histogram increase exponentially, while the number of GMM components remains relatively low.

In this framework, masking the object with a spatial kernel results to a weighted likelihood which inherits the advantages of kernel based approaches. Firstly, the pixels of the target do not contribute equally to the likelihood of the target but they are weighted with respect to their distance from the center of the object. Following the assumption adopted in kernel-based tracking methods [20, 30, 35, 152], it is considered that pixels near the center are more probable to

belong to the object and they contribute more to the total likelihood. On the other hand, pixels which are more distant from the center may be part of the background and their contribution to the object's likelihood should be smaller. Secondly, the weight at each pixel depends on the target location and the maximization of the likelihood is easily obtained with respect to it. This is not the case for a standard GMM likelihood function which cannot be employed in this framework. The localization of the target is obtained by maximizing the weighted likelihood along the frames of the image sequence. Another significant advantage of the method is that the spatial regularization induced by the weight make the similarity function to be smooth and therefore suitable for gradient descent optimization methods.

Furthermore, changes in the appearance of the object are handled by updating the GMM which represents the target. The proposed approach is independent of the target appearance and motion model. When a new color component is observed, it is not generally known if it belongs to the background or the object. The ambiguity is resolved by integrating a new component into the GMM of the target and tracking the target backwards in time. If the backward trajectory does not vary significantly from the forward trajectory, the new color component is accepted as a target's GMM component. Moreover, the algorithm handles scale and rotation changes of the object and numerical experiments showed that it provides, in general, more accurate target localization than state of the art algorithms.

In the remaining of the chapter, section 5.2 describes the tracking algorithm relying on the maximization of the weighted target likelihood, experimental results are presented in section 5.3 and conclusions are drawn is section 5.4.

5.2 Tracking by weighted likelihood

We assume that the object, which is represented by an ellipse, is known in the first frame of the image sequence. Using color and intensity features inside this ellipse a GMM is constructed by employing the EM algorithm. In the rest of the frames, during the tracking procedure, the initial position of the ellipse in the current frame is the same with the position of the ellipse in the immediately previous frame. Starting from this initial position, we move the ellipse along the gradient of the weighted log-likelihood. We continue to move the ellipse until the weighted log-likelihood is reduced. In this chapter, we present the estimation of the GMM parameters and the tracking procedure.

In the first frame we assume that we know the position of the object (the center and the axis of the corresponding ellipse). Let $\boldsymbol{y}$ be a vector representing the coordinates of the center of the ellipse and $\boldsymbol{h} = [h^{(1)}, h^{(2)}]^T$ be a vector with components the lengths of the major and minor axis of the ellipse. The coordinates of the n-th pixel of the image are represented by $\boldsymbol{x}_n = [x_n^{(1)}, x_n^{(2)}]^T$ and the corresponding feature by $\boldsymbol{I}_n$. No ordering of the pixels is implied. The feature $\boldsymbol{I}_n$ carries information on the RGB values of the current pixel. Inclusion of neighboring pixels is straightforward, as the vector $\boldsymbol{I}_n$ may have any dimension. We assign a weight $w_n(\boldsymbol{y})$

to every pixel by masking the ellipse with a kernel $k(\cdot)$:

$$w_n(\boldsymbol{y}) = k\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right), \tag{5.1}$$

where

$$\begin{aligned} f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right) &= \left(\frac{x_n^{(1)} - y^{(1)}}{h^{(1)}}\right)^2 + \left(\frac{x_n^{(2)} - y^{(2)}}{h^{(2)}}\right)^2 \\ &= (\boldsymbol{x}_n - \boldsymbol{y})^T \boldsymbol{H}^{-1} (\boldsymbol{x}_n - \boldsymbol{y}), \end{aligned} \tag{5.2}$$

is the squared Mahalanobis distance between $\boldsymbol{x}_n$ and $\boldsymbol{y}$ with diagonal covariance matrix $\boldsymbol{H} = diag(h^{(1)}, h^{(2)})$.

The kernel $k(\cdot)$ has a decreasing profile and assigns bigger weights to pixels near the center of the ellipse than to pixels near the boundary of the ellipse. For pixels outside the ellipse $k(\cdot) = 0$.

By using function f in (5.2) the drawback of the difference in axis lengths is overcome because the normalized pixel coordinates, for pixels inside the ellipse, are now in the interval $[-1, 1]$.

The log-likelihood of the n-th pixel:

$$L_n = \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{5.3}$$

is described by a GMM of K components with mixing proportions π_k such that $\sum_{k=1}^{K} \pi_k = 1$ with mean vectors $\boldsymbol{\mu}_k$ and covariance matrices $\boldsymbol{\Sigma}_k$, for $k = 1, \ldots, K$.

We now define the *weighted* log-likelihood function for the ellipse with center $\boldsymbol{y}$:

$$L(\boldsymbol{I}, \boldsymbol{w}(\boldsymbol{y}); \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} w_n(\boldsymbol{y}) L_n, \tag{5.4}$$

where N is the number of pixels, $\boldsymbol{I} = \{\boldsymbol{I}_n\}_{n=1,\ldots,N}$, $\boldsymbol{w}(\boldsymbol{y}) = \{w_n(\boldsymbol{y})\}_{n=1,\ldots,N}$, where $w_n(\boldsymbol{y})$ denotes the (non normalized) importance of the n-th pixel to the model.

To estimate the model parameters, the EM algorithm [14] will be used to maximize the weighted log-likelihood. We assume that for each pixel there is a hidden variable $\boldsymbol{z}_n$, which is a vector of K components $\boldsymbol{z}_n = [z_{n,1}, z_{n,2}, \ldots, z_{n,K}]^T$ having all of its components equal to zero except the one responsible for generating the observation $\boldsymbol{I}_n$. Following the standard EM terminology, the pair $(\boldsymbol{I}, \boldsymbol{z})$, where $\boldsymbol{z} = \{\boldsymbol{z}_n\}_{1,\ldots,N}$, forms the complete data. Thus, the complete data log-likelihood:

$$\ln p(\boldsymbol{I}, \boldsymbol{w}(\boldsymbol{y}), \boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \tag{5.5}$$

should be maximized with respect to $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{\pi}$. As the values of the hidden variables are not known, we make use of their posterior distribution:

$$\begin{aligned} \mathcal{L} &= p(\boldsymbol{z}; \boldsymbol{I}, \boldsymbol{w}(\boldsymbol{y}), \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \\ &\propto \prod_{n=1}^{N} \prod_{k=1}^{K} [\pi_k \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_{n,k} w_n(\boldsymbol{y})}, \end{aligned} \tag{5.6}$$

where the difference with the standard GMM definition is that each observation exists with probability $w_n(\boldsymbol{y})$ instead of 1. Under this posterior, the expectation $E[z_{n,k}] = r(z_{n,k})$ can be estimated. Thus, the expectation of the complete-data log-likelihood function conditioned on the expectations of the hidden variables $r(z_{n,k})$ is given by:

$$Q = \sum_{n=1}^{N} w_n(\boldsymbol{y}) \sum_{k=1}^{K} r(z_{n,k})[\ln \pi_k + \ln \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]. \tag{5.7}$$

The EM algorithm can now be employed in order to maximize the weighted log-likelihood (5.4) with respect to $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{\pi}$.

In the E-step, the expectations $r(z_{n,k})$ are computed:

$$E[z_{n,k}] = r(z_{n,k}) = w_n(\boldsymbol{y}) \frac{\pi_k \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^{K} \pi_l \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}. \tag{5.8}$$

In the M-Step, the complete-data log likelihood (5.7) is maximized with respect to the parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\pi}$ leading to the following updates:

$$N_k = \sum_{n=1}^{N} r(z_{n,k}), \tag{5.9}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} r(z_{n,k}) \boldsymbol{I}_n, \tag{5.10}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} r(z_{n,k}) \left(\boldsymbol{I}_n - \boldsymbol{\mu}_k\right) \left(\boldsymbol{I}_n - \boldsymbol{\mu}_k\right)^T, \tag{5.11}$$

$$\pi_k = \frac{N_k}{\sum_{n=1}^{N} w_n(\boldsymbol{y})}. \tag{5.12}$$

We consider that in the first frame, the center $\boldsymbol{y}$ and its size $\boldsymbol{h}$ of the ellipse, which represents the target, are known. For computational purposes, in order to estimate the GMM parameters we use only pixels inside the ellipse, as pixels outside of the ellipse have weight $w_n(\boldsymbol{y}) = 0$. Using the pixels inside this ellipse, we estimate the GMM parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{\pi}$ employing the EM algorithm described above. During the EM algorithm components with importances π_k below a threshold are removed. A limitation of the method is that it can not capture concave objects or objects with highly contaminated background. We partially address this issue by also modeling the background with a GMM and removing components if they are similar with the components of the target. More specifically, we construct another standard GMM (i.e. without weights) for the background using the pixels belonging to an area around the ellipse which represents the object. For the area around the object we used another ellipse whose size is three times the size of the ellipse which represents the object. We use a standard GMM without weights to represent the background in order to treat all these pixels equally (on contrary, the weighted GMM gives more weight to pixels near the center of the ellipse). Afterwards, we remove the components of the object's GMM having centrers $\boldsymbol{\mu}$ which have a small Euclidian distance with any component's center that belongs to the background's GMM.

In the next frame, we seek to estimate the center of the ellipse whose pixels gives the maximum weighted log-likelihood in that frame. Due to the big amount of candidate centers, which are all the pixels of the image, exhaustive search is not feasible as the tracking must be done in real time. Thus, a gradient method is used in order to move the center in order to reach a local maximum of the weighted log-likelihood.

5.2.1 Gradient based update

In order to estimate the position of the object in the next frame, the gradient of the weighted likelihood (5.4) with respect to $\boldsymbol{y}$ must be computed:

$$\begin{aligned} \frac{dL}{d\boldsymbol{y}} &= \frac{dL(\boldsymbol{I}, \boldsymbol{w}(\boldsymbol{y}); \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{d\boldsymbol{y}} \\ &= \sum_{n=1}^{N} \frac{dk\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right)}{d\boldsymbol{y}} L_n, \end{aligned} \tag{5.13}$$

where L_n is the log-likelihood for the n-th pixel defined in (5.3) and

$$\frac{dk\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right)}{d\boldsymbol{y}} = \begin{bmatrix} \frac{dk(f(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}))}{dy^{(1)}} \\ \frac{dk(f(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}))}{dy^{(2)}} \end{bmatrix}. \tag{5.14}$$

By defining the negative derivative of the kernel function as $g(x) = -\frac{dk(x)}{dx}$, we have:

$$\frac{dk\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right)}{d\boldsymbol{y}} = 2\boldsymbol{A}_n(\boldsymbol{y}) g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right), \tag{5.15}$$

where

$$\boldsymbol{A}_n(\boldsymbol{y}) = \left[\frac{x_n^{(1)} - y^{(1)}}{{h^{(1)}}^2}, \frac{x_n^{(2)} - y^{(2)}}{{h^{(2)}}^2} \right]^T, \tag{5.16}$$

leading to:

$$\frac{dL}{d\boldsymbol{y}} = \sum_{n=1}^{N} 2\boldsymbol{A}_n(\boldsymbol{y}) g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right) L_n. \tag{5.17}$$

Once (5.17) is computed, we move the center $\boldsymbol{y}$ along the gradient vector to one of its 8 neighboring pixels, as it is proposed in [152], in order to ensure a smooth motion between frames. Based on the angle of the vector $\frac{dL}{d\boldsymbol{y}}$ we chose one of the 8 neighboring pixels which are adjacent to the current pixel which represents the center $\boldsymbol{y}$. Then the same procedure is repeated for the new center, until the weighted log-likelihood (5.4) decreases. An alternative would be to use the exact values of the gradient vector in order to make steps of variable length. An advantage of using the weighted log-likelihood in (5.4) is that the gradient in (5.17) depends on the target location $\boldsymbol{y}$. This is in contrast with a standard GMM-type likelihood (without the weight), which would not provide a gradient dependent on $\boldsymbol{y}$ and therefore the likelihood maximization with respect to it would not be feasible.

5.2.2 Mean shift-like update

Another approach for estimating the target's position after the computation of the GMM parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{\pi}$ would be to maximize (5.4) by setting its derivative (5.17) with respect to $\boldsymbol{y}$ equal to zero, thus obtaining:

$$\boldsymbol{y} = \frac{\sum_{n=1}^{N} \boldsymbol{x}_n g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right) L_n}{\sum_{n=1}^{N} g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right) L_n}, \tag{5.18}$$

which is a mean shift like update [30]. In (5.18), the log-likelihood L_n for the n-th pixel, which is obtained from (5.3), may have a negative or positive value. The negative values may yield erroneous estimations for the location of the target, as the mean could be shifted out of the convex hull of the pixels inside the ellipse. Moreover, in practice, positive values tend to be small in absolute value, while negative values may be of large amplitude. This results to abrupt changes in the mean location and the object can be lost. To overcome this drawback, L_n should have non negative values. This can be achieved by defining

$$L'_n = \ln\left(B \times \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) = \ln B + L_n \tag{5.19}$$

where B is a normalization factor such that

$$B \times \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \geq 1, \ \forall\, n \in \{1, \ldots, N\}, \tag{5.20}$$

thus the logarithm is always non negative. In our implementation, the normalization term B is set to a large number and we ignore pixels whose values of (5.20) are below 1. By following the same reasoning as before, we can end up in the same update formulas for the EM algorithm as the term in (5.19) is the sum $L'_n = \ln(B) + L_n$ and an update equation like (5.18) is obtained. Thus, in order to locate the object in an image, the tracking procedure can start from an initial position $\boldsymbol{y}_{old}$ (obtained from the object's position in the previous frame) and iteratively apply:

$$\boldsymbol{y}_{new} = \frac{\sum_{n=1}^{N} \boldsymbol{x}_n g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}_{old}, \boldsymbol{h}\right)\right) L'_n}{\sum_{n=1}^{N} g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}_{old}, \boldsymbol{h}\right)\right) L'_n}. \tag{5.21}$$

This procedure stops when the spatial distance between $\boldsymbol{y}_{old}$ and $\boldsymbol{y}_{new}$ is below a threshold which is expressed in pixels and may be relative to the target size. In our implementation we set this threshold in 3% of the target's diagonal. Otherwise, the center is moved to the next position $\boldsymbol{y}_{old} := \boldsymbol{y}_{new}$ and the procedure continues until convergence.

5.2.3 Scale adaptation

In order to scale the target, we could use the derivative of the weighted log-likelihood (5.4) with respect to the components of $\boldsymbol{h}$. For $h^{(1)}$, this would result to:

$$\frac{dL}{dh^{(1)}} = \sum_{n=1}^{N} 2g(f(x_n; \boldsymbol{y}, \boldsymbol{h})) \frac{(x_n^{(1)} - y_n^{(1)})^2}{(h^{(1)})^3} L_n. \tag{5.22}$$

In practice, this derivative is always negative. This results from the fact that $g(f(x_n; \boldsymbol{y}, \boldsymbol{h})) > 0$ because $g(x) = -\frac{dk(x)}{dx}$ and $\frac{dk(x)}{dx} < 0$ as we use a kernel with negative derivative in order to assign bigger weight to pixels near the center of the ellipse. Moreover, $\frac{(x_n^{(1)} - y_n^{(1)})^2}{(h^{(1)})^3} > 0$ as $(x_n^{(1)} - y_n^{(1)})^2 > 0$ and $h^{(1)} > 0$. Finally, in our experiment L_n was negative for the big majority of the pixels (over 99%), and the absolute value of the pixels having negative L_n was much greater than the absolute value of the pixels having positive L_n. Thus, the derivative $\frac{dL}{dh^{(1)}}$ was always negative in practice. Following this approach, in order to maximize (5.4) we had to shrink the ellipse every time, until it reaches 1 pixel.

One commonly used technique [30, 152] is to scale up and down the ellipse which represents the target by a scale factor and keep the scale which maximizes (5.4). However, due to the fact that the log-likelihood function (5.4) depends on the number of pixels N, we can not use (5.4) directly to evaluate the scale of the target. For example, if the size of the ellipse increases, which implies an increase in the number of pixels N, the likelihood in (5.4) will always decrease because it includes all of the terms of the previous (smaller) ellipse and new terms due to the larger size of the new ellipse. In practice, the new terms have negative L_n so the log-likelihood will be decreased if the ellipse gets bigger, or increased if the ellipse gets smaller. Therefore, we will have a likelihood that decreases proportionally to the size of the ellipse, so as in the previous case with the derivative, the ellipse that maximizes the log-likelihood is one pixel wide.

To overcome this drawback, the number of pixels N inside the ellipse, where the likelihood is evaluated, must be constant. To this end, we only consider pixels in a certain grid. The analysis below is done for the horizontal scale, but the procedure for the vertical scale adaptation is similar. The pixels in this grid exist in some columns of the ellipse, as shown in Fig. 5.1. The horizontal distance between neighboring pixels in this grid is d while the vertical distance between pixels in the same column is 1. When the horizontal size of the ellipse $h^{(1)}$ is increased (or decreased) by $\alpha\%$, the horizontal distance d between neighboring pixels in this grid is also increased (or decreased) by the same factor. Thus, the number of pixels N remain constant. This scale adaptation is performed independently in the horizontal and vertical directions and demands less computational resources compared to the computation of the position which necessitates the whole number of pixels inside the ellipse. Moreover, the weights $w_n(\boldsymbol{y})$ are evaluated only for the initial ellipse and are adapted accordingly. For example, in Fig. 5.1, the ellipse at the bottom is scaled up by $\alpha\%$. The weight for the pixels $\boldsymbol{P}$ and $\boldsymbol{P'}$ are equal due to the fact that the first terms in (5.2) are:

$$\begin{aligned} \left(\frac{P'^{(1)} - y^{(1)}}{(1+\alpha) * h^{(1)}}\right)^2 &= \left(\frac{(1+\alpha) * \left(P^{(1)} - y^{(1)}\right)}{(1+\alpha) * h^{(1)}}\right)^2 \\ &= \left(\frac{P^{(1)} - y^{(1)}}{h^{(1)}}\right)^2, \end{aligned} \tag{5.23}$$

while the second terms in (5.2) are equal because there is no scale in the vertical direction. Furthermore, smoothing by a 5×5 Gaussian filter is performed to avoid aliasing during the sampling procedure.

More specifically, in our implementation, for the horizontal adaptation procedure we use the pixels inside the current ellipse to construct a grid of pixels which have a constant horizontal distance (e.g. 10 pixels) with their neighboring points and we evaluate (5.4). Then, we increase and decrease the horizontal size of the ellipse by $\alpha = 10\%$ and we construct a new grid as described in Fig. 5.1. Now, we have three ellipses, the original, one smaller than the original (we will refer to it as *small ellipse*) and one bigger than the original (we will refer to it as *large ellipse*). If the log-likelihood of the original ellipse is greater than the log-likelihood of the other two ellipses, we stop. If the log-likelihood of the *large ellipse* is greater than the log-likelihood of the other two ellipses, we continue to increase the scale by $2\alpha, 3\alpha, \ldots$ until the log-likelihood is decreased or a maximum scale is reached. A similar approach is used if the *small ellipse* has greater log-likelihood than the other two ellipses. The same procedure is repeated for the vertical scale factor. The factor $\alpha = 10\%$ is selected as a tradeoff between the speed in which the ellipse changes (bigger α results to bigger scale changes, but results to more coarse estimation of the scale) and the computational speed (smaller α results to a more fine-grained estimation of the scale, but more increasing or decreasing iterations are needed).

An alternative approach to estimate both scale and rotation parameters would be to compute them directly by the moment of the pixels inside the ellipse [20].

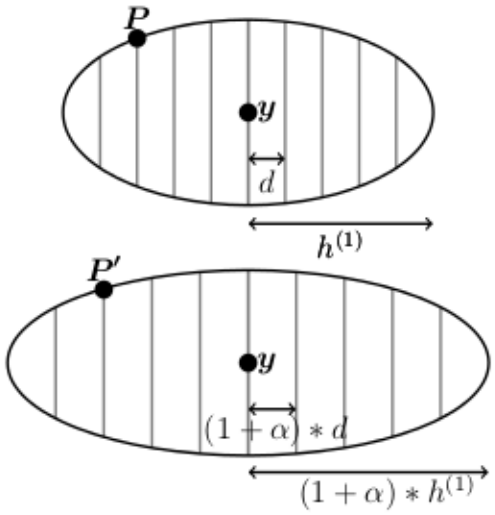


Figure 5.1: The original ellipse (top) and the horizontally scaled ellipse (bottom). The pixels that are used in (5.4) are represented by the gray columns. When the size of the ellipse increases by $\alpha\%$, the inter-column distance is also increased by the same amount. Thus, the number of pixels N is constant and $f(\boldsymbol{P}; \boldsymbol{y}, \boldsymbol{h}) = f(\boldsymbol{P'}; \boldsymbol{y}, \boldsymbol{h})$.

5.2.4 Target model update

The target's appearance (e.g. color) could change making the overall task more difficult. To overcome this difficulty, the main idea is to dynamically update the model of the target by

inserting new components to the GMM using pixels near the target which have small likelihood (under the current model assumptions). Also, if the importance π_k of a component becomes small enough, the component is eliminated from the GMM.

Initially, the weighted GMM is constructed using pixels inside the target ellipse. If a Gaussian component has an importance π_k below a threshold (e.g. bellow $0.1/K$), during the EM algorithm, then this component is removed from the GMM as it has a small contribution to the model. Furthermore, we remove the target's GMM components that are similar to components constructed from an area around and outside the ellipse in order to discriminate the object from the background, as the ellipse contains pixels belonging to the object and probably some pixels belonging to the background. In order to accomplish this, a GMM for the background is initialized using the parameters of the GMM of the target. The pixels from the area around the ellipse have $w_n(\boldsymbol{y}) = 1$, as they are all treated equally (this is equivalent to a standard GMM with no weights). During the EM algorithm for the background GMM, we remove components that have importances below a threshold. After convergence of the EM algorithm for the background GMM, the components that do not change their mean vectors significantly are removed from the target's GMM. In our implementation we removed components that have their center moved below 30 units (we used RGB images, having values $[0 - 255]$ in each component's range). The intuition behind this approach is that there will be similar pixels in the target and the background, resulting to approximately the same GMM component both in the GMM representing the object and in the GMM representing the background.

During the tracking procedure, to make the tracking algorithm more robust and to account for changes in the appearance of the target (i.e. a side of the target having a different color appears), a new component is also created into the GMM of the target at a certain frequency (e.g. every M frames, where M is application dependent and could be as low as 1). In this chapter, we used $M = 50$, which for 25 frames per second results in an update every two seconds. The new component is initialized with parameters computed by the lower quantile of the pixels likelihood. Finally, the EM algorithm is employed in order to estimate the correct center and covariance matrix of the new component. In this modified version of the EM algorithm, the centers and covariances of the current GMM components are not affected. Only their mixing proportions change due to the insertion of the new component. Furthermore, if the importance π_k of a component is below a threshold, the component is removed from the GMM.

Nevertheless, an ambiguity appears concerning whether this new component belongs to the target that changed its appearance or to the background. As a preliminary measure, we also construct a GMM for the background and we remove components from the object's GMM that are similar with the background's GMM. Furthermore, we track the target from the current position back in time by considering the last M frames and the respective positions of the target in these frames. The idea of backward tracking has also been proposed in [59] for tracking individual points, in [120] for scale estimation and in [68] in order to estimate the robustness of a tracker. Here we apply this idea to the target model. If the trajectory of the new weighted GMM is similar to the original trajectory, that is the average Euclidian distance between the centers of the ellipses and the sizes of the axis are below a threshold, then we assume that the

target has changed its appearance and the new component belongs to the target whose GMM is updated. Otherwise, the target model remains the same as these pixels are more probable to belong to the background. Let $\boldsymbol{y}_t$ and $\boldsymbol{h}_t$ be the center and the axis of the ellipse at time t estimated by the tracking algorithm while $\boldsymbol{y}'_t$ and $\boldsymbol{h}'_t$ be the center and the axis of the ellipse at time t estimated by tracking the target backward in time during the update procedure. Note that the sequence in which $\boldsymbol{y}'_t$ are estimated is $\boldsymbol{y}'_T$, $\boldsymbol{y}'_{T-1}$, ..., $\boldsymbol{y}'_{T-M}$ (the same applies to $\boldsymbol{h}'_t$). The average Euclidian distance between the centers and axis, respectively, is defined as:

$$Euc_y(\boldsymbol{y}, \boldsymbol{y}') = \frac{1}{M} \sum_{t=0}^{M} \sqrt{\sum_{j=1}^{2} \left(\frac{\boldsymbol{y}_{T-t}^{(j)} - \boldsymbol{y}_{T-t}^{\prime(j)}}{\frac{\boldsymbol{h}_{T-t}^{(j)} + \boldsymbol{h}_{T-t}^{\prime(j)}}{2}} \right)^2}, \tag{5.24}$$

$$Euc_h(\boldsymbol{h}, \boldsymbol{h}') = \frac{1}{M} \sum_{t=0}^{M} \sqrt{\sum_{j=1}^{2} \left(\frac{\boldsymbol{h}_{T-t}^{(j)} - \boldsymbol{h}_{T-t}^{\prime(j)}}{\boldsymbol{h}_{T-t}^{(j)}} \right)^2}, \tag{5.25}$$

where T is the current time. The distance $Euc_y(\boldsymbol{y}, \boldsymbol{y}')$ is normalized using the average of the axis size at the corresponding time. The GMM changes if both distances are below a threshold, i.e. $Euc_y(\boldsymbol{y}, \boldsymbol{y}') < Th_y$ and $Euc_h(\boldsymbol{h}, \boldsymbol{h}') < Th_h$, where $Th_y = Th_h = 0.1$.

The GMM update procedure is applied every M frames and it inserts at most one new component to the GMM which represents the target, while it can remove several components. After some calls of the update procedure, a different GMM (compared to the one constructed in the initial frame) may be constructed. So, we propose a technique in order to estimate if the new GMM that has been constructed can represent the target. By using the current position of the target and the position in previous frames, we examine if we can use the new GMM in order to track the target backwards in time accurately. The accuracy is estimated by comparing the respective positions of the backward tracking with the positions that have been estimated during the forward tracking procedure. Moreover, using this approach we do not need to predefine the number of components accurately. Indeed, if we choose a bigger number for the GMM components, the additional components will be removed as they will have small importance π_k. If the number of components is smaller, new components may be added during the update procedure. If the change in illumination or self-occlusion is gradual and not abrupt the proposed mechanism is expected to correctly update the model (e.g. Fi. 5.7). On the other hand, sudden changes in illumination or self-occlusions are more difficult to be handled by the proposed method.

In order to handle rotations, a heuristic method is employed which rotates the ellipse by small steps of 2° in the interval $[-45^\circ, +45^\circ]$ at each iteration and selects the angle providing the maximum value of the log-likelihood (5.4). In practice, only very small rotations between consecutive frames are observed.

The overall procedure describing the initialization and the tracking is presented in the weighted likelihood tracking (WLT) Algorithm 5. The update of the GMM parameters is described in Algorithm 6.

Algorithm 5 *WLT* algorithm

function WLT(Image sequence, M)
    Input: an image sequence consisting of T frames and the frequency M of updating the target model.
    Output: the ellipse center $\boldsymbol{y}$ at each frame.
    Initialization:
    Determine the initial position $\boldsymbol{y}_1$ and the size $\boldsymbol{h}_1$ of the target.
    Compute the parameters π_k, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ of the GMM describing the target using (5.10), (5.11) and (5.12).
    Tracking:
    for frame $t = 2, \ldots, T$ **do**
        $\boldsymbol{y}_t := \boldsymbol{y}_{t-1}$
        $\boldsymbol{h}_t := \boldsymbol{h}_{t-1}$
        while the likelihood in (5.4) increases **do**
            Move to $\boldsymbol{y}_t$ using (5.17).
        end while
        Estimate horizontal and vertical sizes of the target $\boldsymbol{h}_t = [h_t^{(1)}, h_t^{(2)}]^T$.
        Estimate the rotation R of the target.
        if mod$(t, M) == 0$ **then**
            Update the target model using Algorithm 6.
        end if
    end for
end function

5.3 Experimental results

The evaluation of the proposed tracking algorithm was performed using nine real datasets (Fig. 5.5). We used two variations of the proposed method, one based on the derivative (5.17), which will be referred as WLT, and one based on the mean shift-like formula (5.21), which will be referred as WLTMS. The image sequences *Real1* (449 frames), *Real2* (199 frames), *Real3* (299 frames) and *Real4* (309 frames) are taken from the PETS'01 database, the datasets *Real5* (129 frames), *Real6* (169 frames) and *Real7* (109 frames) are taken from PETS'06 database and the datasets *Real8* (71 frames) and *Real9* (121 frames) are taken from PETS'09 database. In all of these image sequences the targets change their position and size simultaneously. The ground truth for these image sequences was manually determined (both for the size and the position of the target). Note that although we show the ground truth delimited by rectangles, the WLT algorithm employs the inscribed ellipse in its computations. In our experimental evaluation we used $B = 10^6$, $M = 50$ frames and $Th_y = Th_h = 0.1$.

As each object is represented by an ellipse, in order to evaluate the performance of a tracking algorithm we use the center and the size of the ellipse axis. We employ the evaluation criteria that were used in [152]. The first criterion is the number of frames which the object is correctly

Algorithm 6 Target update

function TARGETUPDATE(targetGMM, M)
    Input: The GMM representing the target and the last M frames of the image sequence.
    Output: the new GMM representing the target.
    newGMM := targetGMM
    Create a new component for the newGMM (initialize using a pixel with a small likelihood and apply the EM only for the new component).
    Delete components with π_k below a threshold.
    Create a GMM for the background using an area around the target in the last frame and remove the components of newGMM whose mean vectors are close (Euclidian distance) to the components of the GMM of the background.
    for frame $t = M, \ldots, 1$ **do**
        Track the target in frame t using newGMM.
    end for
    if the trajectory created by tracking backwards the target using newGMM is close (Euclidian distance) to the trajectory of the target for the last M frames **then**
        return newGMM
    else
        return targetGMM
    end if
end function

tracked in. An object is considered to be correctly tracked in a frame if the estimated rectangle covers at least 25% of the area of the target in the ground truth. This is a coarse measure, and is only considered in order to roughly evaluate if the estimated object is near the ground truth object. The next two measures provide more details about the performance of the algorithms. The second criterion is the position error which is the Euclidian distance between the center of the object in the ground truth and the estimated target center, divided by the diagonal of the ground truth rectangle. The third criterion is the size error which is defined as the Euclidian distance between the ground truth and the estimated vectors (with components the width and the height of the ellipse), normalized by the ground truth length of the object diagonal. The division with the diagonal of the object eliminates the problems of different object sizes. Finally, three other criterions are the average precision:

$$p = \frac{1}{T}\sum_{i=1}^{T} p_i, \tag{5.26}$$

where

$$p_i = \frac{\text{number of correctly tracked pixels in frame } i}{\text{number of tracked pixels in frame } i}, \tag{5.27}$$

the average recall:

$$r = \frac{1}{T}\sum_{i=1}^{T} r_i, \tag{5.28}$$

where

$$r_i = \frac{\text{number of correctly tracked pixels in frame } i}{\text{number of target pixels in frame } i}, \tag{5.29}$$

and the average F-measure:

$$F = \frac{1}{T}\sum_{i=1}^{T}\frac{p_i \times r_i}{p_i + r_i}. \tag{5.30}$$

In our experiments we use a kernel with an exponential profile having $\sigma = 1$:

$$k(x) = \begin{cases} e^{(-x/\sigma)} & \text{if} \quad x \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{5.31}$$

Consecutively, the derivative of (5.17) becomes:

$$\frac{dL}{d\boldsymbol{y}} = \sum_{n=1}^{N} \boldsymbol{A}_n(\boldsymbol{y}) w_n(\boldsymbol{y}) L_n. \tag{5.32}$$

We compared our method with the OpenCV's implementation of Camshift algorithm [20, 19] which is a robust version of the mean shift algorithm [30] with scale adaptation and the FRAG tracker [3]. For Camshift, we used a 16 bin histogram for the hue component. Also, we did not take into account pixels with low or high brightness or low saturation (we apply thresholds equal to 10% of the maximum pixel value) as it is suggested in [20]. For comparison purposes, we did not search for the rotation of the target in Camshift in order to have a common baseline. For FRAG, we used the version provided by the authors which uses the grayscale information and is quantized to 16 bins.

In Tables 5.1-5.6 and Figures 5.2-5.4, the quantitative results of the compared methods are presented. The position and size errors are expressed in normalized coordinates. Thus, a position error of 0.5 means that the center of the estimated target is positioned in the middle of a ray of the ground truth ellipse. Similarly, a size error of 0.5 means that the estimated size is half the size of the ground truth ellipse. In *Real1* and *Real2*, where the targets are cars under different illumination conditions, all algorithms successfully track the objects with Camshift and WLTMS having a slightly better performance in terms of position error. In *Real3* and *Real4*, the target is a car viewed from the rear under different illumination conditions. In *Real3*, the color of the car is similar with the color of the road and Camshift did not estimate the position of the object accurately (the rectangle representing the target scaled up and included both the road and the car). Although we consider that Camshift tracked the target (the ground truth rectangle is inside the rectangle computed by Camshift), the position and size errors are large while the precision is small. In *Real4*, Camshift fails to track the object after the half of the image sequence due to the fact that the color of the target is similar with the color of the background mountains. In contrast, FRAG, WLT and WLMS successfully track the objects in *Real3* and *Real4* despite these difficulties with WLTMS having a slightly better performance in terms of position error. The image sequences *Real5*, *Real6* and *Real7* are taken inside a subway using cameras with different viewpoint angles and show persons walking. In *Real5* and *Real7*, a partial occlusion happens as another person walks between the camera and the target

and in *Real6* another person passes very close to the target. All approaches successfully track the objects, with WLT showing a significantly better performance in terms of position and size errors. In *Real8*, a woman is walking. In this dataset only Camshift and WLT successfully track the object with Camshift giving better results. On the other hand, FRAG and WLTMS lose the object from the early frames. They lose the object after a couple of frames, due to the fact that the object is close to the camera, and the difference in its position between consecutive frames is big. Finally, in *Real9*, a man in black clothes is walking among other people with dark colored clothes. FRAG loses the target in the early frames. Camshift follows the target in the majority of the frames, but loses the target in the end. In contrast, both WLT and WLTMS successfully track the target with WLT having better performance in terms of position error. Qualitative results for WLT are presented in Fig. 5.5. For each sequence, the left figure shows the first frame of the sequence, while the other frames are uniform samples in time. These examples show that WLT and WLTMS have comparable performance in terms of position and size error when the displacement of the object is small between consecutive frames. However, when the displacement is larger, e.g. in Real8, WLTMS may fail to localize the object correctly. This results from the fact that using (5.21), the new center may be significantly further with respect to the current center, and may not provide the maximum of the log-likelihood (5.4). On the other hand, WLT uses one pixel displacements in every iteration, after evaluating (5.17), and results to smoother position changes and estimated final locations which minimize the log-likelihood (5.4).

Table 5.1: Performance of camshift, FRAG, WLT and WLTMS in terms of correct target localization.

	Frames Tracked			
Seq.	Camshift	FRAG	WLT	WLTMS
Real1	**499/499**	**499/499**	**499/499**	**499/499**
Real2	**199/199**	**199/199**	**199/199**	**199/199**
Real3	**299/299**	**299/299**	**299/299**	**299/299**
Real4	165/309	**309/309**	**309/309**	**309/309**
Real5	**129/129**	**129/129**	**129/129**	**129/129**
Real6	**169/169**	**169/169**	**169/169**	**169/169**
Real7	**109/109**	**109/109**	**109/109**	**109/109**
Real8	**71/71**	3/71	**71/71**	2/71
Real9	116/121	6/121	**121/121**	**121/121**

Also, we evaluated the performance of the algorithm when the target rotates. In Fig. 5.9, representative frames of the image sequence that is used for testing are shown. The object performs a rotation of 130°, while moving during 62 frames. We used the WLT method for tracking, as the ellipse rotation procedure is the same for all of its variants. The algorithm successfully tracks the object, as the average error in the estimation of the rotation angle is 2.73° with standard deviation of 1.86.

Table 5.2: Performance of camshift, FRAG, WLT and WLTMS in terms of position error (mean $\pm$ std).

Seq.	Camshift	FRAG	WLT	WLTMS
Real1	**0.07**$\pm$0.04	0.15$\pm$0.05	0.10$\pm$0.05	**0.07**$\pm$0.05
Real2	0.08$\pm$0.04	0.19$\pm$0.09	0.14$\pm$0.03	**0.06**$\pm$0.02
Real3	2.36$\pm$0.82	0.26$\pm$0.27	**0.18**$\pm$0.06	**0.18**$\pm$0.04
Real4	3.00$\pm$1.89	0.27$\pm$0.16	0.12$\pm$0.06	**0.11**$\pm$0.05
Real5	0.26$\pm$0.12	0.16$\pm$0.02	**0.13**$\pm$0.04	0.20$\pm$0.48
Real6	0.26$\pm$0.18	0.30$\pm$0.08	**0.15**$\pm$0.05	0.22$\pm$0.05
Real7	0.28$\pm$0.27	**0.13**$\pm$0.07	0.20$\pm$0.09	0.25$\pm$0.06
Real8	**0.05**$\pm$0.03	**0.05**$\pm$0.02	0.07$\pm$0.05	0.08$\pm$0.01
Real9	0.43$\pm$0.08	**0.08**$\pm$0.02	0.12$\pm$0.09	0.34$\pm$0.10

Table 5.3: Performance of camshift, FRAG, WLT and WLTMS in terms of size error (mean $\pm$ std).

Seq.	Camshift	FRAG	WLT	WLTMS
Real1	0.23$\pm$0.21	**0.21**$\pm$0.11	0.23$\pm$0.09	0.24$\pm$0.09
Real2	0.23$\pm$0.07	0.28$\pm$0.10	0.32$\pm$0.05	**0.15**$\pm$0.05
Real3	8.26$\pm$2.99	0.60$\pm$0.29	0.21$\pm$0.12	**0.18**$\pm$0.08
Real4	3.32$\pm$1.61	1.52$\pm$1.27	**0.21**$\pm$0.10	0.30$\pm$0.11
Real5	0.45$\pm$0.15	**0.14**$\pm$0.03	0.34$\pm$0.07	0.31$\pm$0.06
Real6	0.42$\pm$0.44	0.28$\pm$0.10	**0.27**$\pm$0.08	0.38$\pm$0.08
Real7	0.34$\pm$0.33	**0.16**$\pm$0.10	0.28$\pm$0.12	0.35$\pm$0.12
Real8	**0.09**$\pm$0.10	0.11$\pm$0.02	0.12$\pm$0.04	0.21$\pm$0.03
Real9	0.96$\pm$0.18	0.75$\pm$0.10	**0.20**$\pm$0.15	0.45$\pm$0.14

Furthermore, to justify the use of a weighted likelihood, we compared the WLT algorithm with a tracking procedure using a standard GMM (referred by LT). The LT algorithm is the same as WLT, with two differences: a) the GMM which is constructed in the first frame is a standard GMM without location dependent weights and b) in order to move the center to one of the 8 neighboring pixels we evaluate the standard log-likelihood in each of these 8 pixels, considering them to be the center of the ellipse. This last distinction makes the LT algorithm about 8 times slower compared to the WLT algorithm.

Therefore, we compared WLT with LT in terms of the larger initial ellipse that makes the algorithm insensitive. More specifically, if the initial ellipse in the first frame, is erroneously larger than the ground truth ellipse, the algorithm will be trapped by the background elements included in the initial ellipse. In Table 5.7, the maximum initial target size is shown which does not affect correct tracking. As it can be observed in all cases, WLT accepts a larger initial

Table 5.4: Performance of camshift, FRAG, WLT and WLTMS in terms of precision (mean ± std).

Seq.	Camshift	FRAG	WLT	WLTMS
Real1	**0.95**±0.17	0.73±0.11	0.69±0.07	0.77±0.11
Real2	0.79±0.18	0.90±0.19	0.90±0.02	**0.93**±0.06
Real3	0.03±0.08	0.56±0.28	**0.71**±0.13	0.67±0.08
Real4	0.14±0.10	0.26±0.22	0.69±0.12	**0.71**±0.13
Real5	**0.96**±0.06	0.81±0.05	0.82±0.08	0.91±0.09
Real6	0.70±0.28	0.70±0.16	0.84±0.13	**0.90**±0.14
Real7	0.79±0.24	0.90±0.06	**0.91**±0.10	0.90±0.09
Real8	**0.92**±0.15	0.04±0.02	0.86±0.15	0.03±0.42
Real9	0.34±0.14	0.04±0.02	0.77±0.19	**0.80**±0.13

Table 5.5: Performance of camshift, FRAG, WLT and WLTMS in terms of recall (mean ± std).

Seq.	Camshift	FRAG	WLT	WLTMS
Real1	0.61±0.13	0.76±0.12	0.78±0.18	**0.84**±0.17
Real2	**0.82**±0.11	0.39±0.10	0.80±0.09	0.79±0.11
Real3	0.47±0.19	**0.87**±0.16	0.67±0.07	0.72±0.08
Real4	0.49±0.14	**0.98**±0.14	0.84±0.13	0.84±0.08
Real5	0.45±0.21	0.81±0.06	**0.89**±0.08	0.61±0.09
Real6	0.38±0.21	0.88±0.15	**0.90**±0.13	0.57±0.18
Real7	0.71±0.13	**0.72**±0.15	0.67±0.14	0.60±0.13
Real8	**0.77**±0.09	0.03±0.02	0.76±0.18	0.01±0.01
Real9	0.65±0.20	0.03±0.01	**0.84**±0.20	0.50±0.12

window by the user as it assigns smaller weights to pixels far from the window center. On the other hand, the standard GMM does not associate with small weights pixels that are far from the center and are more likely to belong to the background and therefore they affect the correct estimation of the GMM parameters. We also compared these approaches with respect to the size of the smaller initial ellipse, but in this case both algorithms provide similar accuracies, as the ellipse is small and all its pixels belong to the object. Hence, we do not present these results in Table 5.7.

In these image sequences, the rectangles which represent the targets have dimensions around 150×70 pixels. For these target sizes, our algorithm, which is developed using OpenCV, runs in real time, as the average time needed for each frame is at most 0.015 sec (or equivalently at least 65 fps) for both variations (both WLT and WLTMS). The computer used during the experimental evaluation is a dual core PC (even though in the implementations we did not use the second core) at 1.83GHz with 2GB RAM at 667 MHz.

Table 5.6: Performance of camshift, FRAG, WLT and WLTMS in terms of F-measure (mean ± std).

Seq.	Camshift	FRAG	WLT	WLTMS
Real1	0.73±0.09	0.73±0.09	0.72±0.10	**0.78**±0.09
Real2	0.75±0.07	0.53±0.06	0.82±0.09	**0.84**±0.06
Real3	0.04±0.14	0.61±0.16	**0.67**±0.07	**0.67**±0.04
Real4	0.18±0.05	0.36± 0.22	**0.74**±0.05	**0.74**±0.07
Real5	0.57±0.20	0.81±0.05	**0.85**±0.05	0.76±0.05
Real6	0.42±0.21	0.75±0.08	**0.86**±0.08	0.70±0.08
Real7	0.71±0.15	**0.79**± 0.11	0.76±0.09	0.70±0.08
Real8	**0.83**±0.09	0.03± 0.03	0.78±0.10	0.02±0.02
Real9	0.35±0.13	0.03± 0.01	**0.78**±0.10	0.60±0.11

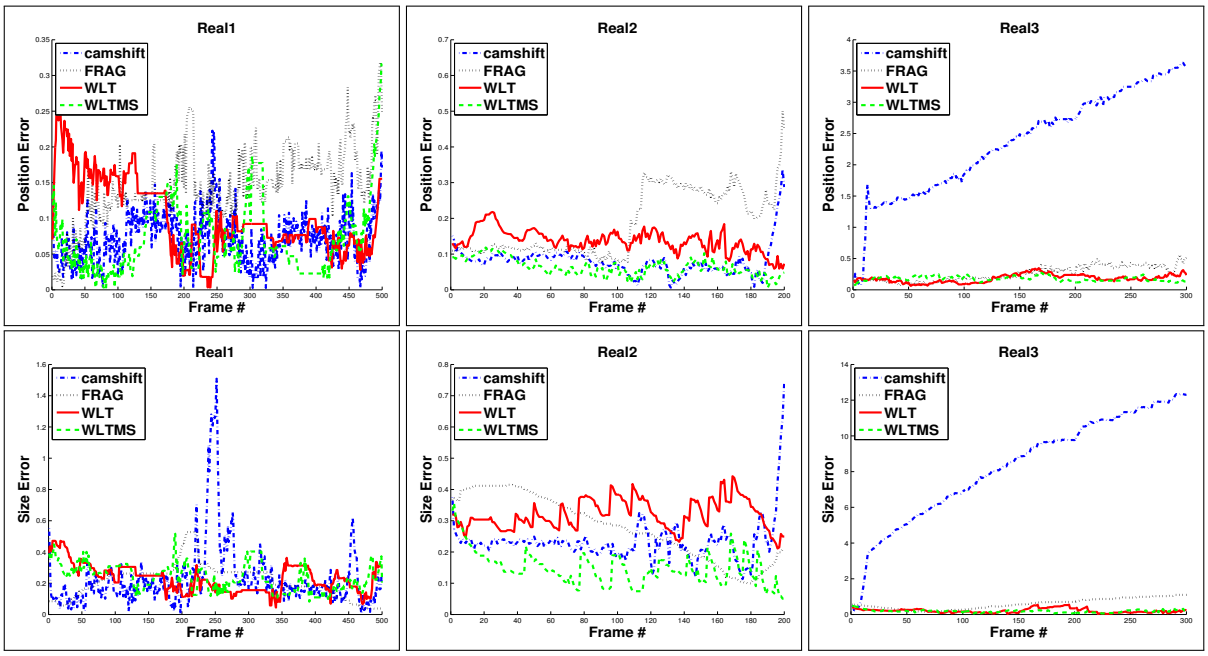


Figure 5.2: Performance of camshift, FRAG, WLT and WLTMS in terms of position and size error.

Finally, we present some qualitative results for the model update method using WLT. We used an image sequence of 71 frames showing a rotating chair. The front view of the chair has a purple color while the back view of the chair is black. The initialization is accomplished in the first frame (frame 0), where only the back view is visible. Afterwards, the chair moves from left to right while rotating twice around its axis (Fig. 5.7). We check for an update according to Algorithm 6 every 10 frames. In frame 10, where the back side of the chair is not visible, the tracking algorithm tracks a small black part of the chair. After frame 10, the model is updated and a component for the purple color of the front view of the chair is added. After the second rotation of the chair (frame 71), the tracking algorithm covers a large area of the purple area of the back of the chair. In Fig. 5.8, quantitative results are presented for the the position and size

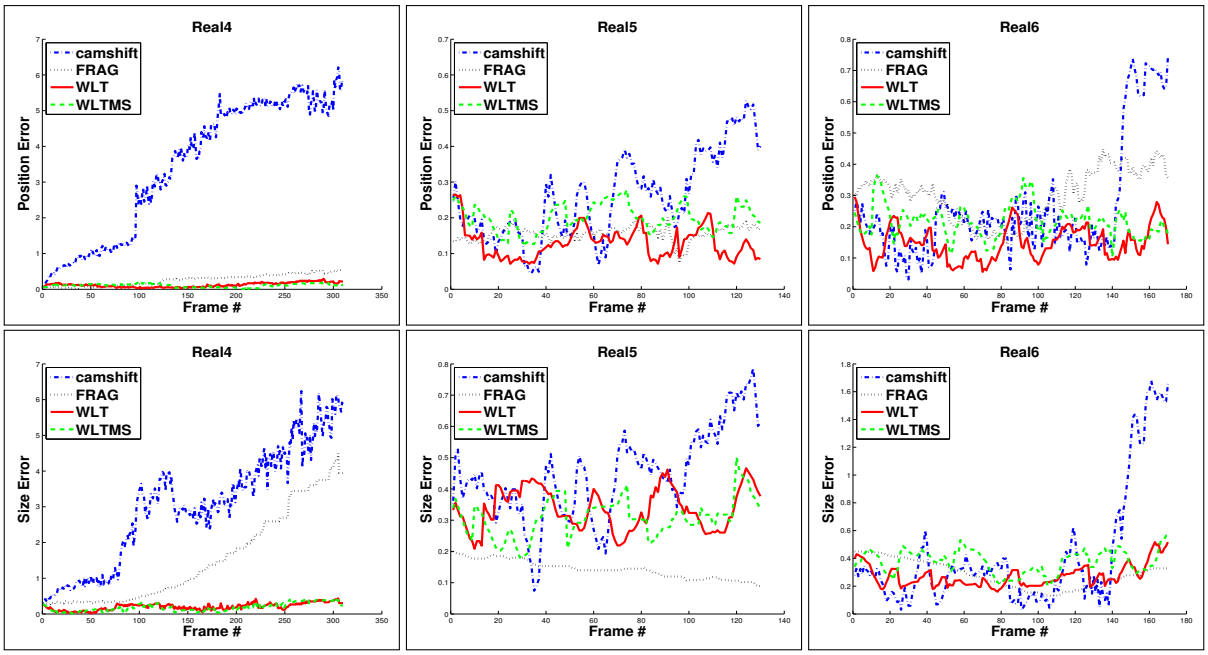


Figure 5.3: Performance of camshift, FRAG, WLT and WLTMS in terms of position and size error.

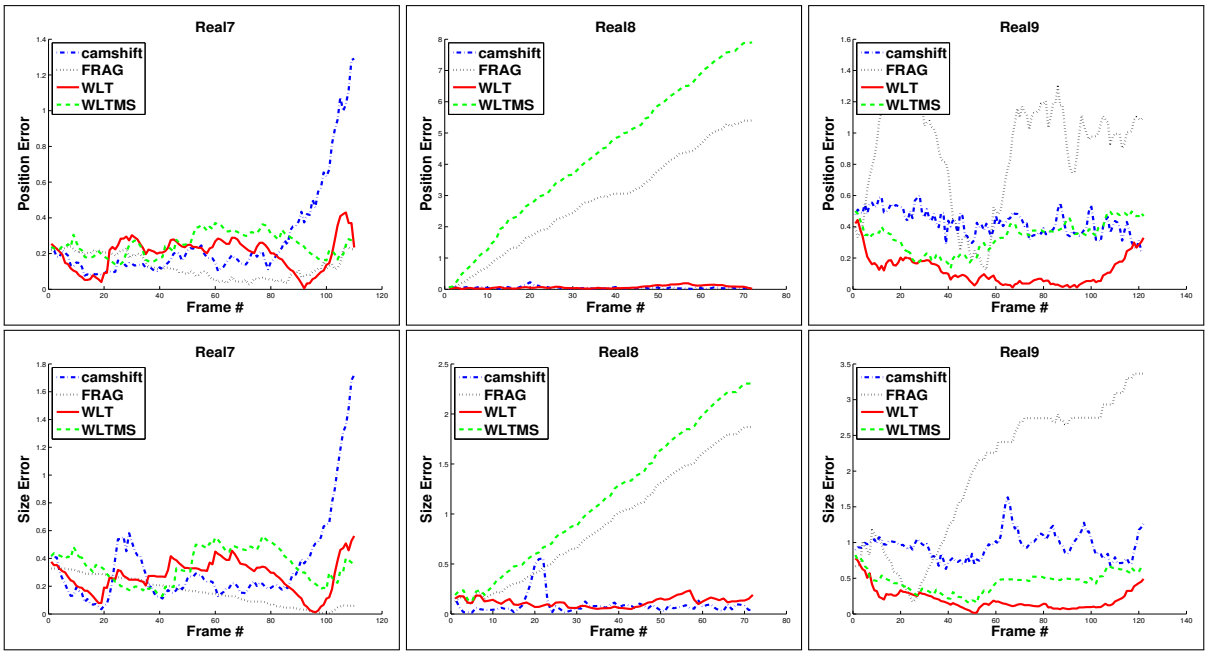


Figure 5.4: Performance of camshift, FRAG, WLT and WLTMS in terms of position and size error.

errors. We compared the performance of WLT without model update and with model update. The implementation, in which the initial model does not change, misses the object after some frames when the first rotation of the chair occurs. This is the reason why WLT without model update has a smaller size error (the target is missed). On the other hand, WLT with model update has to adapt its size in order to locate the object correctly.

Figure 5.5: Representative results on the real datasets used in the experiments *Real1*, *Real2*, *Real3* and *Real4*, *Real5*, *Real6*, *Real7*, *Real8* and *Real9* using WLT. Although the inscribed ellipse is used in the computations, the target is bounded by a green rectangle for visualization purposes.

frame 0 frame 20 frame 40 frame 61

Figure 5.6: Representative frames of the sequence used for the evaluation of the algorithm on rotations of the target.

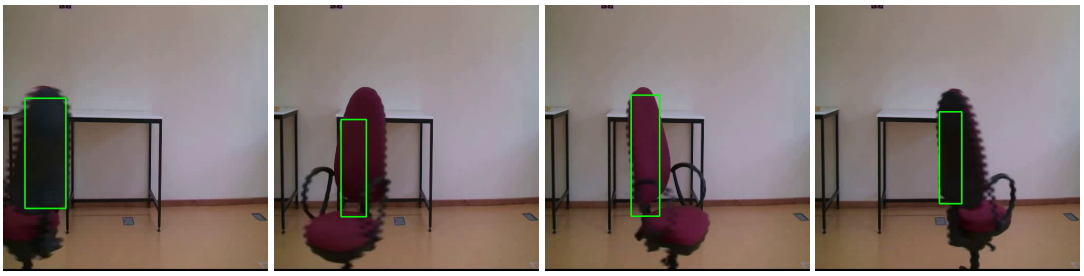
5.3.1 Experimental results on the VOT2014 dataset

We also evaluated the proposed method using the Visual Object Tracking (VOT) 2014 dataset (URL: http://votchallenge.net). A description of the dataset and the evaluation methodology can be found in [65]. VOT provides the toolset in order to evaluate a new tracker over the dataset as the performances of already tested trackers have been recorded. A comprehensive comparative report is the outcome of the toolset. This dataset consists of 25 video sequences including various visual phenomena like camera motion, illumination change, motion change, size change and occlusion. The selected objects in each sequence were manually annotated by bounding boxes. The report generated includes the results of 38 trackers which were evaluated by the authors of [65].

The evaluation indices used in order to estimate the performance of our tracker are the

Table 5.7: Comparison of WLT and LT in terms of the maximum allowable target initialization area.

	Max initial target size		Ratio
Seq.	WLT	LT	WLT/LT
Real1	74×32	48×22	2.242
Real2	130×27	128×25	1.096
Real3	112×62	114×60	1.015
Real4	211×162	146×128	1.837
Real5	60×170	50×130	1.569
Real6	50×162	50×154	1.051
Real7	42×150	36×150	1.166
Real8	30×250	29×218	1.186
Real9	100×400	82×400	1.219



frame 0 frame 10 frame 20 frame 30

frame 40 frame 50 frame 60 frame 71

Figure 5.7: Representative frames for the sequence that is used for the qualitative evaluation of the model update (the total number of frames that were used during the tracking procedure is 71). The chair rotates around its axis and moves from left to right. The model update procedure is applied every 10 frames. While in the initial frame only the black color is included in the target model, in the final frame (number 71) both the black and the purple colors are included in the model.

accuracy and the robustness. The accuracy measures how well the bounding box A^T estimated by the tracker overlaps with the ground truth bounding box A^G and is defined by:

$$acc = \frac{A^G \cap A^T}{A^G \cup A^T}. \tag{5.33}$$

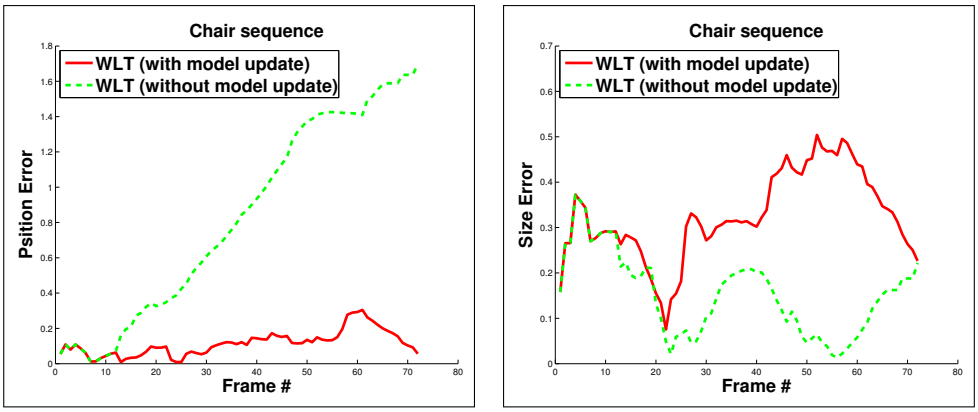


Figure 5.8: Performance of WLT without model update (green) and with model update (red).

The robustness is the number of times the tracker failed to locate the object correctly. A target is considered lost when the $A^G \cap A^T = \emptyset$, that is, there is no overlap between the estimated target and the ground truth. In this case, the tracker is reinitialized from the the ground truth in order to continue tracking and estimate the indices in the rest of the video sequence.

The evaluation procedure is as follows: the tracker runs on each sequence at most 15 times (3 times if it is deterministic, 15 if not deterministic) and the average accuracy and robustness of the sequences is indicated. Moreover, two set of experiments are performed: (i) the baseline experiments, in which the initial position of the tracker is exactly the ground truth in the first frame and (ii) the region noise experiments, in which the initial position of the tracker, whenever it is initialized, is the ground truth perturbed by some noise, which uniformly affects the position and the size of the target by $\pm 10\%$ pixels and the orientation of the ground truth bounding box by ± 0.1 radians.

The performance of our WLTMS method is summarized in Table 5.8. There are two evaluations: a qualitative estimation which gives the performance of the proposed method with respect to the mean of the rest of the already tested trackers and a quantitative index showing the ordering of our method with respect to the rest of the trackers. It is worth noting that the 38 other trackers constitute the state of the art in the framework of the VOT2014 dataset [65]. Moreover, as it is stated in [65], none of the examined algorithms outperforms all the others in all test.

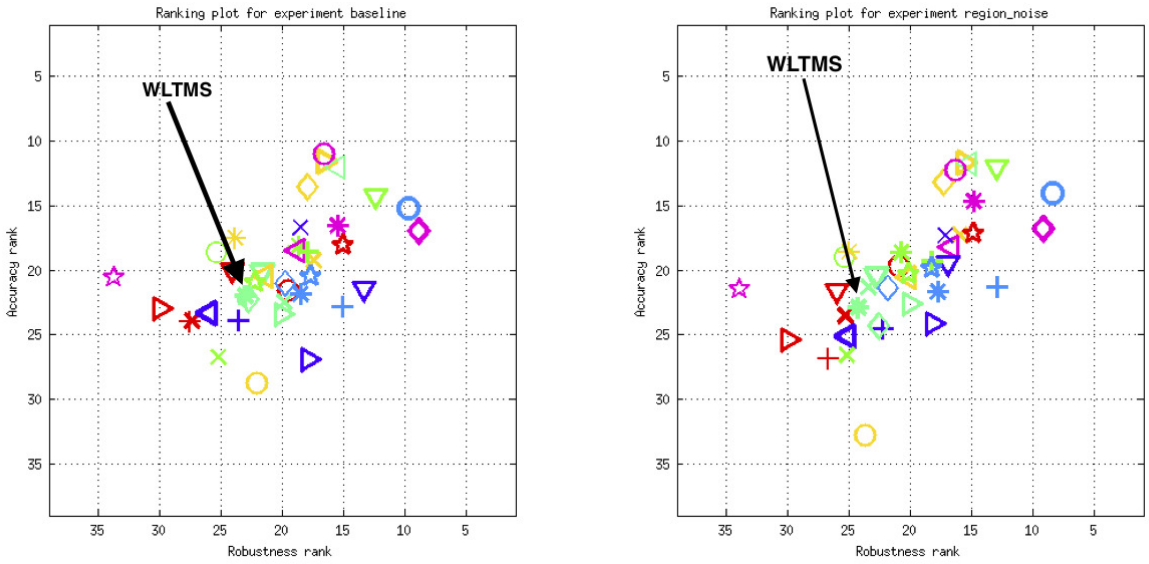
In the majority of the sequences, our method has average performance with respect to the rest of the algorithms. In some cases it has below average performance and in a few cases it exhibits a performance above average. The best performance for our method is achieved for the *fish2* sequence, where our method is ranked second in the baseline experiment concerning the accuracy index. In the sequences *diving* and *gymnastics*, our algorithm has a performance which is above the average in terms of robustness with respect to the other algorithms in the dataset. Especially for the sequence *gymnastics*, the performance in terms of accuracy for region noise is drastically increased with respect to the baseline. These sequences have rotated targets, and our method, which is based on kernel tracking may perform better due to the fact that no exact matching of the target region is needed in contrast to template matching algorithms in the VOT2014 dataset. The worst performance is achieved for sequences where the target is or

Table 5.8: Performance of the proposed WLTMS method over the VOT2014 dataset. The labels Average, Below and Above indicate the performance of the tracker with respect to the the mean of the state-of-the-art algorithms considered in the evaluation. The ordering of the algorithm's performance is also indicated for each video sequence.

	Baseline experiments		Region noise experiments	
Seq.	Accuracy	Robustness	Accuracy	Robustness
ball	Average (18/39))	Average (31/39)	Average (13/39)	Average (24/39)
basketball	Average (9/39)	Average (18/39)	Above (5/39)	Average (14/39)
bicycle	Average (24/39)	Below (37/39)	Below (36/39)	Below (38/39)
bolt	Average (13/39)	Average (19/39)	Average (15/39)	Average (19/39)
car	Below (39/39)	Below (37/39)	Below (39/39)	Below (32/39)
david	Below (34/39)	Below (36/39)	Average (33/39)	Below (35/39)
diving	Below (35/39)	Above (11/39)	Average (21/39)	Above (6/39)
drunk	Below (32/39)	Below (37/39)	Below (31/39)	Below (34/39)
fernando	Average (19/39)	Average (26/39)	Below (37/39)	Below (34/39)
fish1	Average (8/39)	Average (26/39)	Average (22/39)	Average (20/39)
fish2	Above (2/39)	Average (13/39)	Above (2/39)	Above (9/39)
gymnastics	Average (30/39)	Above (8/39)	Average (7/39)	Above (8/39)
hand1	Average (11/39)	Average (14/39)	Above (12/39)	Average (12/39)
hand2	Average (13/39)	Above (12/39)	Average (21/39)	Average (15/39)
jogging	Below (34/39)	Average (27/39)	Average (28/39)	Below (37/39)
motocross	Average (27/39)	Below (37/39)	Below (35/39)	Below (37/39)
polarbear	Average (25/39)	Average (39/39)	Average (19/39)	Average (38/39)
skating	Below (31/39)	Below (31/39)	Below (36/39)	Below (33/39)
sphere	Below (32/39)	Below (35/39)	Below (32/39)	Below (35/39)
sunshade	Average (20/39)	Average (24/39)	Average (27/39)	Average (17/39)
surfing	Average (30/39)	Average (37/39)	Average (26/39)	Average (35/39)
torus	Below (38/39)	Below (36/39)	Below (38/39)	Below (34/39)
trellis	Average (32/39)	Above (15/39)	Average (30/39)	Average (23/39)
tunnel	Above (11/39)	Below (39/39)	Average (21/39)	Below (39/39)
woman	Average (20/39)	Below (36/39)	Average (22/39)	Below (33/39)

becomes very small in the image sequence. For example, in the *tunnel* sequence, the target is the jacket of a man riding a motorbike which moves inside a tunnel. In many frames, the target occupies a rectangular area of 20×30 pixels, which cannot be correctly tracked by our algorithm as the number of pixels is low to be successfully handled. More specifically, the number of pixels inside the ellipse is small for the initialization of the GMM (the estimation of the GMM parameters fails). Moreover, when the motion is large (due to the fact that the camera moves quickly) and no overlapping section exists between the target in two consecutive frames, our algorithm also fails as it starts from the initial position of the previous frame and performs a

local optimization procedure. This drawback may be eliminated if some sort of particle filtering is employed.



(a) (b)

Figure 5.9: Comparative evaluation of the proposed WLTMS (green square indicated by the arrow) with respect to state-of-the-art algorithms over all the video sequences of the VOT2014 data set. The plot is generated by the VOT 2014 toolset. (a) Baseline experiments and (b) region noise experiment.

A graphical representation of the performance of the proposed method with respect to the state of the art is shown in Fig.5.9 which is generated by the VOT toolset. The horizontal axis represents the robustness and the vertical axis shows the accuracy. Better performance is from bottom to up and from left to right. As we can observe, our method is situated very close to the average performance of the state-of-the-art methods, which makes it competitive.

5.4 Conclusions

From the point of view of the target modeling and localization, the proposed algorithm belongs to the same family as the histogram based methods [19, 20, 30, 152]. These methods minimize the distance between the probability distribution of the model and the distribution of the pixels at a candidate location in an image frame. The mean shift family of methods [20, 30] minimizes the Bhattacharyya distance while in [152] the earth mover's distance is involved. The WLT method proposed herein, maximizes the weighted log-likelihood of the model without creating a second distribution in the image frame under consideration. The key issue in estimating the target's position is the weight term depending on the location of the target. Concerning the two versions of our algorithm (WLT and WLTMS), WLTMS shows in general a slightly better performance and it is favored due to its faster convergence. More specifically, in each iteration WLT moves the center of the ellipse by exactly one pixel, while WLTMS may move the center

of the ellipse by a larger step and consequently it may converge faster.

CHAPTER 6

REAL TIME VISUAL TRACKING USING A SPATIALLY WEIGHTED VON MISES MIXTURE MODEL

6.1 Introduction

In this chapter, we use the hue component of the HSV color space for visual object tracking and we employ a weighted von Mises mixture model in order to overcome drawbacks caused by shifting histogram values. The von Mises distribution is the circular analog of the normal distribution on a line, and it can be used in order to model circular data. The values of the hue component that is periodic with period 2π, can be represented as points in the two dimensional unit circle. Thus, the terms periodic random variable and circular data will be considered interchanged in this chapter. Moreover, we propose the weighted von Mises mixture to model the distribution of the hue value when a single von Mises distribution is not flexible enough to describe the target. Moreover, the proposed weighted von Mises mixture employs the spatial weights that are provided by the kernel. The von Mises distribution has been employed in order to model sensor noise [98], the direction of the movement [24, 64, 108], and the pose of an object [56]. In [104], the background hue component is modeled through a single wrapped

Gaussian distribution which is also designed for circular data. Here, we use a weighted von Mises mixture in order to model the model the appearance and track the target efficiently.

In the remaining of the chapter, section 6.2 reviews the von Mises distribution and presents the proposed weighted von Mises mixture model, section 6.3 integrates the proposed weighted von Mises mixture model in the visual tracking framework. Experimental results are presented in section 6.4 and conclusions are drawn is section 6.5.

6.2 Weighted von Mises mixture model

6.2.1 Introduction to the von Mises distribution

There are cases in image processing and analysis where the measured quantity is periodic and modeling it by a periodic variable may be an advantage (e.g. the hue component of an image in the HSV color space). In what follows, we assume that the period is 2π and the periodic variable is defined in the interval $[0, 2\pi)$. If the variable is defined in another interval, we may map this interval to $[0, 2\pi)$. We will also refer to the observations (e.g. hue values) as angles accordingly.

The main drawback of circular data is that we can not directly apply a conventional distribution (e.g. Gaussian) as there is a dependence on the choice of the origin. For example, if we have two angles one at 0 and one at π, then if we select 0 as the origin then the mean of these angles is $\pi/2$. However, if we select $\pi/2$ as the origin, that is the interval is $[\pi/2, 5\pi/2)$, the mean is $3\pi/2$ due to the fact that the angle 0 is mapped to the angle 2π. In order to overcome these drawbacks, the von Mises distribution has been proposed. For a complete reference to its properties, the reader is referred to [14]. Here we summarize the key points.

The von Mises probability density function for an angle a is given by:

$$\mathcal{M}(a; \theta, m) = \frac{1}{2\pi I_0(m)} e^{m\cos(a-\theta)}, \tag{6.1}$$

where θ is the mean, m is the concentration (analogous to the inverse variance), $I_0(m)$ is the zeroth-order Bessel function of the first kind [2], which is defined as $I_0(m) = \int_0^{2\pi} e^{m\cos(t)} dt$. For large values of m the distribution becomes Gaussian and for $m = 0$ it becomes uniform.

In order to estimate the parameters θ and m having some observed angles $\boldsymbol{A} = \{a_n\}_{n=1,\ldots,N}$, we can use the maximum likelihood estimation. The log-likelihood of the model is given by:

$$\begin{aligned} \ln p(\boldsymbol{A}; \theta, m) &= \ln \prod_{n=1}^{N} \mathcal{M}(a_n; \theta, m) \\ &= -N \ln(2\pi) - N \ln(I_0(m)) + m \sum_{n=1}^{N} \cos(a_n - \theta). \end{aligned} \tag{6.2}$$

By maximizing (6.2) with respect to θ we obtain:

$$\theta = \tan^{-1}\left(\frac{\sum_{n=1}^{N} sin(a_n)}{\sum_{n=1}^{N} cos(a_n)} \right). \tag{6.3}$$

By maximizing (6.2) with respect to m we obtain the equation:

$$\frac{I_1(m)}{I_0(m)} = \frac{1}{N}\sum_{n=1}^{N}\cos(a_n - \theta), \tag{6.4}$$

which can be numerically solved, where $I_1(m) = I_0'(m) = \int_0^{2\pi} e^{m\cos(t)}\cos(t)dt$.

6.2.2 Von Mises mixture model

If the von Mises distribution is not flexible enough in order to model the observations, then we can use the von Mises mixture model as a linear superposition of von Mises components. The probability density function of an angle a for a von Mises mixture model can be defined as:

$$\mathcal{L}(a; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi}) = \sum_{k=1}^{K}\pi_k\mathcal{M}(a; \theta_k, m_k), \tag{6.5}$$

where K is the number of components, $\boldsymbol{\theta} = \{\theta_k\}_{k=1,\dots,K}$ are the means of the components, $\boldsymbol{m} = \{m_k\}_{k=1,\dots,K}$ are the concentrations of the components and $\boldsymbol{\pi} = \{\pi_k\}_{k=1,\dots,K}$ are the importances (weights) of the components.

In order to estimate the parameters we have to maximize the log-likelihood function with respect to these parameters, which can be achieved using the Expectation-Maximization algorithm [14]. We assume that we have observer N angles $\boldsymbol{A} = \{a_n\}_{n=1,\dots,N}$ and we want to estimate the parameters $\boldsymbol{\theta}$, $\boldsymbol{m}$ and $\boldsymbol{\pi}$ of the von Mises mixture model. The log-likelihood function is defined as:

$$\ln L(\boldsymbol{A}; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi}) = \sum_{n=1}^{N}\ln\left(\mathcal{L}(a_n; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi})\right) \tag{6.6}$$

We can define a set of random variables $\boldsymbol{Z} = \{\boldsymbol{z}_n\}_{n=1,\dots,N}$, where $\boldsymbol{z}_n$ is a K-dimensional binary random variable which has $z_{nk} = 1$ if the n-th angle is produced from the k-th component, and $z_{nj} = 0$ for $j \neq k$. Thus $\boldsymbol{z}_n$ can reveal from which component the observation a_n has been generated. In practice, the values of the variables $\boldsymbol{Z}$ are not known, so they are called latent variables. If the value of the corresponding latent variable $\boldsymbol{z}_n$ is known for each observation a_n, then the set $\{\boldsymbol{A}, \boldsymbol{Z}\}$ is called the complete data set. The complete data log-likelihood function is given by:

$$\begin{aligned}
&\ln L(\boldsymbol{A}, \boldsymbol{Z}; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi}) \\
&= \ln\left(\prod_{n=1}^{N}\prod_{k=1}^{K}\left(\pi_k\mathcal{M}(a_n; \theta_k, m_k)\right)^{z_{nk}}\right) \\
&= \sum_{n=1}^{N}\sum_{k=1}^{K} z_{nk}\left(\ln\pi_k + \mathcal{M}(a_n; \theta_k, m_k)\right).
\end{aligned} \tag{6.7}$$

Due to the fact that the latent variables $\boldsymbol{Z}$ are not know, we can only use their posterior distribution:

$$p(\boldsymbol{Z};\boldsymbol{A},\boldsymbol{\theta},\boldsymbol{m},\boldsymbol{\pi}) = \frac{p(\boldsymbol{A};\boldsymbol{Z},\boldsymbol{\theta},\boldsymbol{m},\boldsymbol{\pi})p(\boldsymbol{Z};\boldsymbol{\theta},\boldsymbol{m},\boldsymbol{\pi})}{p(\boldsymbol{A};\boldsymbol{\theta},\boldsymbol{m},\boldsymbol{\pi})} \propto \prod_{n=1}^{N}\prod_{k=1}^{K}\left(\pi_k \mathcal{M}(a_n;\theta_k,m_k)\right)^{z_{nk}}. \tag{6.8}$$

The expectation of the complete data log-likelihood is given by:

$$\begin{aligned} Q &= \sum_{\boldsymbol{Z}} p(\boldsymbol{Z};\boldsymbol{A},\boldsymbol{\theta},\boldsymbol{m},\boldsymbol{\pi}) \ln L(\boldsymbol{A},\boldsymbol{Z};\boldsymbol{\theta},\boldsymbol{m},\boldsymbol{\pi}) \\ &= \sum_{n=1}^{N}\sum_{k=1}^{K} r(z_{nk})\left(\ln \pi_k + \ln \mathcal{M}(a_n;\theta_k,m_k)\right), \end{aligned} \tag{6.9}$$

where the $r(z_{nk})$ is the expectation of the latent variable z_{nk}:

$$\begin{aligned} r(z_{nk}) = E[z_{nk}] &= \frac{\sum_{z_{nk}} z_{nk} p(z_{nk};a_n,\theta_k,m_k,\pi_k)}{\sum_{z_{nj}} p(z_{nj};a_n,\theta_j,m_j,\pi_j)} \\ &= \frac{\pi_k \mathcal{M}(a_n;\theta_k,m_k)}{\sum_{j=1}^{K}\pi_j \mathcal{M}(a_n;\theta_j,m_j)}. \end{aligned} \tag{6.10}$$

Now, the maximization of (6.9) with respect to $\boldsymbol{\theta},\boldsymbol{m},\boldsymbol{\pi}$ can be easily achieved.

Thus, in order to evaluate the parameters $\boldsymbol{\theta},\boldsymbol{m},\boldsymbol{\pi}$ of the von Mises mixture model, we initialize these parameters to some values and repeatedly apply the E-step and M-step.

E-step:

$$r(z_{nk}) = \frac{\pi_k \mathcal{M}(a_n;\theta_k,m_k)}{\sum_{j=1}^{K}\pi_j \mathcal{M}(a_n;\theta_j,m_j)}. \tag{6.11}$$

M-step:

$$N_k = \sum_{n=1}^{N} r(z_{nk}), \tag{6.12}$$

$$\pi_k = \frac{N_k}{N}, \tag{6.13}$$

$$\theta_k = \tan^{-1}\frac{\sum_{n=1}^{N} r(z_{nk})\sin(a_n)}{\sum_{n=1}^{N} r(z_{nk})\cos(a_n)}, \tag{6.14}$$

$$\frac{I_1(m)}{I_0(m)} = \frac{1}{N_k}\sum_{n=1}^{N} r(z_{nk})\cos(a_n-\theta_k). \tag{6.15}$$

Note that (6.15) is not in closed form but can be numerically solved with respect to the parameter m.

6.2.3 Weighted von Mises mixture model

In some applications, some observations may affect more the log-likelihood (6.6) with respect to others. For example, if we are confident that a pixel is more important than the others, then the log likelihood becomes:

$$\ln L(\boldsymbol{A}, \boldsymbol{w}; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi}) = \sum_{n=1}^{N} w_n \ln \left(\mathcal{L}(a_n; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi}) \right), \tag{6.16}$$

where $\boldsymbol{w} = \{w_n\}_{n=1,\ldots,N}$ are the weights of the observations. The intuition behind this approach is that if we use a pixel multiple times in (6.6), then we aggregate its appearances to w_n in (6.16).

By using the same approach the E-step equation (6.11) remains the same. On the other hand, the M-step equations (6.13), (6.14) and (6.15) change accordingly:

$$N_k = \sum_{n=1}^{N} w_n r(z_{nk}), \tag{6.17}$$

$$\pi_k = \frac{N_k}{\sum_{n=1}^{N} w_n}, \tag{6.18}$$

$$\theta_k = \tan^{-1} \frac{\sum_{n=1}^{N} w_n r(z_{nk}) \sin(a_n)}{\sum_{n=1}^{N} w_n r(z_{nk}) \cos(a_n)}, \tag{6.19}$$

$$\frac{I_1(m)}{I_0(m)} = \frac{1}{N_k} \sum_{n=1}^{N} w_n r(z_{nk}) \cos(a_n - \theta_k). \tag{6.20}$$

6.3 Tracking using the weighted von Mises mixture model

In this chapter, we assume that the images employ the HSV color model and we use only the hue component, that is, each pixel is represented by a single value in the interval $[0, 2\pi)$. Moreover, we assume that the object to be tracked can be represented by an ellipse. The ellipse has a center denoted by $\boldsymbol{y} = [y^{(1)}, y^{(2)}]^T$, where $y^{(1)}$ is the horizontal coordinate and $y^{(2)}$ is the vertical coordinate of the center in the image coordinates system, and a vector $\boldsymbol{h} = [h^{(1)}, h^{(2)}]^T$, where $h^{(1)}$ is the length of the horizontal semi-axis and $h^{(2)}$ is the length of the vertical semi-axis of the ellipse.

Having set the parameters $\boldsymbol{y}$ and $\boldsymbol{h}$, we can assign a weight to every pixel of the image by using a spatial kernel $k(t)$ which will assign greater weights to pixels near the center of the ellipse. More specifically, we use a kernel with exponential profile:

$$k(t) = \begin{cases} e^{(-t/\sigma)} & \text{if} \quad t \leq 1 \\ 0 & \text{otherwise} \end{cases}. \tag{6.21}$$

Using this kernel, the weight $w_n(\boldsymbol{y})$ of the n-th pixel with spatial coordinates $\boldsymbol{x}_n = [x_n^{(1)}, x_n^{(2)}]^T$ is given by:

$$w_n(\boldsymbol{y}) = k(M(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h})), \tag{6.22}$$

where

$$M\left(\boldsymbol{x}_n;\boldsymbol{y},\boldsymbol{h}\right) = \left(\frac{x_n^{(1)}-y^{(1)}}{h^{(1)}}\right)^2+\left(\frac{x_n^{(2)}-y^{(2)}}{h^{(2)}}\right)^2 = (\boldsymbol{x}_n-\boldsymbol{y})^T\boldsymbol{H}^{-1}(\boldsymbol{x}_n-\boldsymbol{y}), \tag{6.23}$$

is the squared Mahalanobis distance between $\boldsymbol{x}_n$ and $\boldsymbol{y}$ with diagonal covariance matrix $\boldsymbol{H} = diag(h^{(1)}, h^{(2)})$. By using the function M in (6.22) the drawback of the difference in axis lengths is overcome because the normalized pixel coordinates, for pixels inside the ellipse, are now in the interval $[0, 1]$. Thus, the weights $w_n(\boldsymbol{y})$ for pixels inside the ellipse are greater than zero, while pixels outside the ellipse have weights equal to zero.

6.3.1 First frame

We assume that the position of the ellipse is known in the first frame of the sequence. Thus, the objective here is to estimate the von Mises mixture model using the hue component of the pixels. The image consists of N pixels (with some given order, e.g. row-by-row), each pixel's weight $w_n(\boldsymbol{y})$ is given by (6.22) and each pixel's hue component is denoted by a_n. We can now estimate the von Mises mixture model parameters $\boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi}$ using equations (6.11), (6.18), (6.19) and (6.20) of the EM algorithm.

6.3.2 Tracking in consecutive frames

In every frame of the video (except for the first), we know: (i) the center $\boldsymbol{y}$ and the size $\boldsymbol{h}$ of the ellipse which represents the target in the imediatelly previous frame and (ii) the parameters $\boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi}$ of the von Mises mixture model which models the distribution of the hue component of the object's pixels. In order to estimate the center of the ellipse in the current frame, a gradient based technique will be used.

We seek to estimate the position $\boldsymbol{y}$ which maximizes the log-likelihood:

$$\ln L(\boldsymbol{A}, \boldsymbol{w}(\boldsymbol{y}); \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi}) = \sum_{n=1}^{N} w_n(\boldsymbol{y}) \ln\left(\mathcal{L}(a_n; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi})\right). \tag{6.24}$$

This can be achieved by taking the derivative of (6.24) and setting it to zero. The derivative of (6.24) is defined as:

$$\frac{dL}{d\boldsymbol{y}} = \left[\frac{dL}{dy^{(1)}}, \frac{dL}{dy^{(2)}}\right]^T, \tag{6.25}$$

where:

$$\frac{dL}{dy^{(j)}} = \sum_{n=1}^{N} \frac{dw_n(\boldsymbol{y})}{dy^{(j)}} \mathcal{L}(a_n; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi}). \tag{6.26}$$

The only term that depends on $\boldsymbol{y}$ is $w_n(\boldsymbol{y})$. By defining the negative derivative of the kernel function as $g(t) = -\frac{dk(t)}{dt}$, we have:

$$\frac{dk\left(M\left(\boldsymbol{x}_n;\boldsymbol{y},\boldsymbol{h}\right)\right)}{dy^{(j)}} \propto g\left(M\left(\boldsymbol{x}_n;\boldsymbol{y},\boldsymbol{h}\right)\right) \frac{x_n^{(j)}-y^{(j)}}{h^{(j)^2}}. \tag{6.27}$$

By substituting (6.27) into (6.26) we have:

$$\frac{dL}{dy^{(j)}} \propto \sum_{n=1}^{N} g\left(M\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right) \frac{x_n^{(j)} - y^{(j)}}{{h^{(j)}}^2} \mathcal{L}(a_n; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi}). \tag{6.28}$$

By setting (6.28) equal to zero, we get the update formula (in vector form):

$$\boldsymbol{y} = \frac{\sum_{n=1}^{N} \boldsymbol{x}_n g\left(M\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right) \mathcal{L}(a_n; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi})}{\sum_{n=1}^{N} g\left(M\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right) \mathcal{L}(a_n; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi})}. \tag{6.29}$$

Thus, in every frame, starting from $\boldsymbol{y}$ estimated at the previous frame, we iteratively apply equation (6.29) in order to move the center $\boldsymbol{y}$ to a new position, until (6.24) decreases. In (6.29), the value of $\boldsymbol{y}$ in the right side of the equation is the new center while the value of $\boldsymbol{y}$ in the left side of the equation is the old center. Scale estimation can be performed by increasing and decreasing the ellipse size by a percentage (for example 10%) and choose the ellipse with the bigger average likelihood.

6.3.3 Implementation details

The execution time of the proposed algorithm can be improved as the values of the hue component are integers in the interval $[0, 359]$.

First, in (6.16), the term $\mathcal{L}(a_n; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi})$ depends only on the hue value of the pixel. Thus, we can aggregate the weight w_n of the pixels that have the same hue value to a new weight W_n. This is equivalent to creating a new image with 360 pixels having values from 0 to 359 and assign to each pixel the corresponding weight $W_i = \sum_{n=1}^{N} w_n \delta(i - a_n)$. The delta function is zero everywhere except the $\delta(0) = 1$. Using this approach, the number of the pixels used by the EM algorithm is constant and this makes also the time needed for the initialization on the first frame relatively constant.

Second, in (6.29), the term $\mathcal{L}(a_n; \boldsymbol{\theta}, \boldsymbol{m}, \boldsymbol{\pi})$ can be pre-calculated for all the values of a_n. The parameters $\boldsymbol{\theta}$, $\boldsymbol{m}$ and $\boldsymbol{\pi}$ are determined for the first frame and are keep constant in the subsequent frames. Thus, we can have an array of 360 values which can be computed after the estimation of the parameters $\boldsymbol{\theta}$, $\boldsymbol{m}$ and $\boldsymbol{\pi}$. During the tracking procedure, we can use this array instead of the equations (6.16) and (6.1).

6.4 Experimental results

In order to evaluate the proposed method we used the Visual Object Tracking (VOT) 2014 dataset (URL: http://votchallenge.net). A detailed description on the dataset and the evaluation methodology can be found in [65]. Here we will provide a quick overview of the dataset and the toolkit.

The VOT dataset consist of 25 color image sequences with one moving object in each sequence. These videos include various visual phenomena (also called frame attributes) like camera motion, illumination change, motion change, size change and occlusion. In every image of

every image sequence, the ground trough of the target has been manually annotated by bounding boxes. The information provided for the initialization of the tracker is the bounding box in the first frame. If the tracker loses the object, then it is reinitialized in a subsequent frame.

The toolkit evaluates the performance of a tracker in terms of accuracy and robustness. The evaluation is performed N_{rep} times for each video sequence, which allows dealing with potential variance in the performance. The accuracy is associated with the average overlap per repetition per frame between the target's ground truth bounding box and the bounding box which was estimated by the tracker. The robustness index is associated with the average number of times the tracker failed per repetition. A target is considered to have lost the object when there is no overlap between the estimated target and the ground truth.

For fair comparison, a ranking-based methodology is used [65]. Thus, the average accuracy and the average robustness are correlated with the accuracy rank and the robustness rank respectively. For each tracker, a group of equivalent trackers is determined and a rank is then calculated. The group of equivalent trackers is found per tracker, as the concept of equivalent trackers is not transitive. For example, tracker T_1 can perform indistinguishably from trackers T_2 and T_3, but trackers T_2 and T_3 may not be indistinguishably between themselves. In the end, the rank is the mean of the ranks in the equivalent trackers group. For visual comparison, the AR-rank plot is created, where the axes correspond to the accuracy and robustness rank. The best performing trackers appear at the top-right corner of the AR-rank plot, while the less efficient trackers appear at the bottom-left corner.

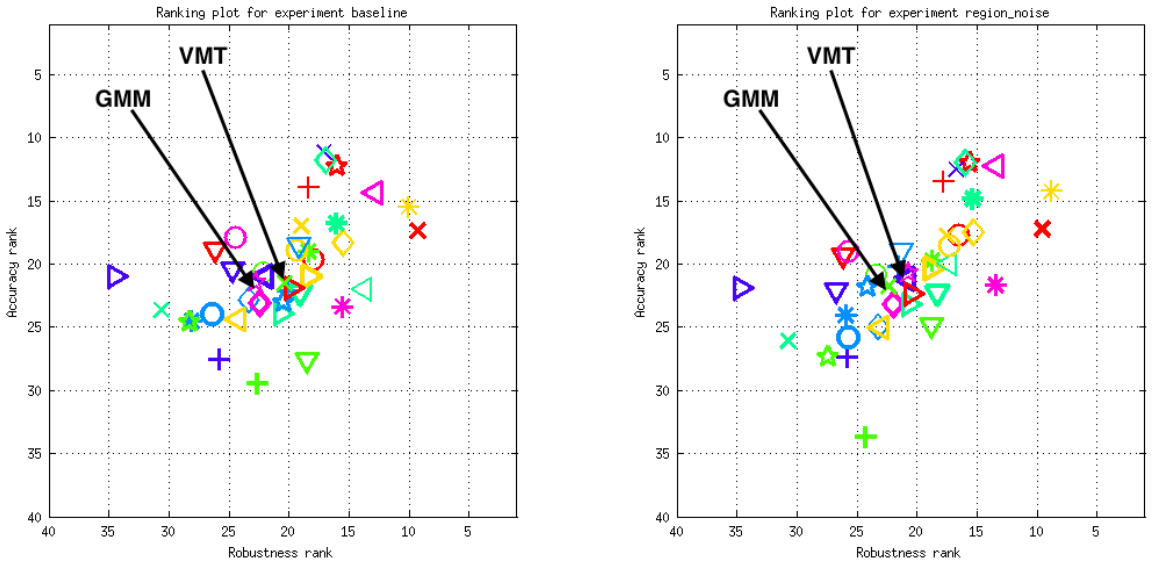
The performance of the tracker is evaluated in two sets of experiments. In the first set, which is called Baseline, the initialization of the target is done using the exact ground truth bounding box. In the second set, which is called Region Noise, a noisy initialization is done. More specifically, the ground truth bounding box position and size are perturbed by drawing uniformly from $[0 - 10\%]$ of the bounding box size.

Moreover, the VOT toolkit provides the tools that are needed in order to evaluate the performance of a tracker over the VOT dataset. It provides the performance of 38 trackers that have been already tested by the authors [65] and the tools needed to compare a new tracker with these state-of-the-art algorithms.

The performance of the proposed method (denoted by VMT) using VOT 2014 is presented in Fig. 6.1 and Tables 6.1-6.4. The number of components K is set a priori, but its estimation is not a guess. The components of the von Mises mixture model, roughly represent the number of colors of the object. In our experiments, we noticed that if K is set to a large value, some components will have $\pi_k = 0$.

In Fig. 6.1, the AR-rank plots for the Baseline and Region Noise experiments are presented. The horizontal and vertical axis denote the robustness rank and accuracy rank respectively. The proposed tracker, which is highlighted, is placed near the center of the plot, thus it has average performance in both measures with respect to the other algorithms. It is worth noting that the 38 other trackers constitute the state of the art in the framework of the VOT2014 dataset [65]. Moreover, the performance of a similar tracker, (denoted by GMM) that used the Gaussian distribution instead of the von Mises distribution is presented. Due to the fact that the Gaussian

distribution can not model circular data in the beginning of the axis accurately, it exhibits an inferior performance than the von Mises distribution.



(a) (b)

Figure 6.1: Comparative evaluation of the proposed VMT (green star indicated by the arrow) with respect to state-of-the-art algorithms over all the video sequences of the VOT2014 data set. The plot is generated by the VOT 2014 toolset. (a) Baseline experiments and (b) Region Noise experiments.

In Tables 6.1-6.4, the average performance of the proposed method for each image sequence is presented. The name of the sequence, the performance of the proposed method (denoted by VMT), the variance of the tracker that uses the Gaussian distribution (GMM) and the minimum, maximum, mean and standard deviation of the other 38 trackers are shown. The last row of each table shows the average for all sequences for the tracker. Tables 6.1-6.2 correspond to the Baseline experiments, while Tables 6.3-6.4 correspond to the Region Noise experiments. For each set of experiments (Baseline and Region Noise) there are two tables which represent: a) the accuracy rank (Tables 6.1 and 6.3) and b) the robustness rank (Tables 6.2 and 6.4). The accuracy rank is correlated with the overlap and the robustness rank is correlated with the number of failures. Thus, if a target performs well, for example, in accuracy with respect to the other trackers, it will also perform well in terms of overlap. In the majority of the cases, both VMT and GMM have comparable performances. These are the cases where the histogram bins are not located in the beginning of the axis, so the Gaussian and von Mises distribution can model the data. In some cases where the data are located in the beginning or the end of the axis (e.g. sphere sequence) the VMT performs better than the GMM tracker).

In Table 6.1, the average performance in terms of accuracy rank is provided for the Baseline experiment. In the majority of the sequences, VMT performance lies near the average performance of the 38 trackers. In 7 sequences (ball, diving, fish1, fish2, hand1, polarbear, sphere) it performs better than the mean while in 4 sequences (bicycle, car, skating, torus) its performance is inferior to the mean value. The last row which indicates the average performance between

the sequences confirms that our algorithm has performance near the mean performance of the 38 trackers. As it is stated in [65], none of the examined algorithms outperforms all the others in all tests.

Table 6.1: Performance of VMT in terms of accuracy rank for the Baseline experiments (less is better).

Seq.	VMT	GMM	Other Trackers		
			min	max	mean±std
ball	9.50	9.50	1.50	39.00	20.26 ± 11.31
basketball	14.00	16.00	4.00	38.50	20.15 ± 10.33
bicycle	30.88	30.50	1.50	32.00	19.72 ± 9.32
bolt	17.50	18.03	3.00	36.40	20.09 ± 5.72
car	32.00	32.50	2.50	37.50	19.70 ± 10.94
david	21.00	21.50	4.50	35.00	19.97 ± 8.87
diving	11.00	10.55	4.62	32.58	20.11 ± 8.05
drunk	23.65	23.65	1.50	39.00	19.90 ± 10.55
fernando	24.00	22.40	1.00	36.75	19.91 ± 7.49
fish1	4.00	12.06	1.50	39.00	20.40 ± 7.99
fish2	12.50	13.00	1.00	32.42	20.16 ± 7.05
gymnastics	21.87	22.21	15.50	39.00	19.85 ± 4.77
hand1	14.00	14.00	4.00	38.00	20.05 ± 10.52
hand2	24.56	27.00	2.00	36.20	19.76 ± 9.94
jogging	28.00	26.50	7.71	39.00	19.78 ± 8.61
motocross	25.80	29.56	1.50	35.20	19.51 ± 9.78
polarbear	16.50	16.50	2.50	38.50	20.09 ± 10.56
skating	32.50	33.45	3.50	36.17	19.68 ± 8.68
sphere	16.00	25.60	1.50	39.00	20.18 ± 11.25
sunshade	28.57	27.00	9.00	38.50	19.76 ± 10.15
surfing	24.00	25.50	5.00	39.00	19.89 ± 10.33
torus	36.33	36.80	2.50	39.00	19.56 ± 10.97
trellis	22.00	22.00	2.00	37.25	19.94 ± 9.97
tunnel	23.00	16.30	1.00	37.50	19.92 ± 10.63
woman	22.50	33.00	6.58	39.00	19.93 ± 9.86
Average	21.42	22.61	10.97	28.72	19.93 ± 4.20

In Table 6.2, the average performance in terms of robustness rank is provided for the Baseline experiment. In 11 sequences (ball, car, david, diving, gymnastics, polarbear, skating, sphere, sunshade, surfing, trellis), VMT has exactly the same performance as the best of the 38 trackers (these are the sequences that the target is never lost), in 7 sequences (basketball, bolt, fernando, fish2, hand1, hand2, jogging), VMT has performance near the average and in 7 sequences (bicycle, drunk, fish1, motocross, torus, tunnel, woman), its performance is inferior to the average performance.

Table 6.2: Performance of VMT in terms of robustness rank for the Baseline experiments (less is better).

Seq.	VMT	GMM	Other Trackers		
			min	max	mean±std
ball	10.50	10.50	10.50	38.50	20.25 ± 10.51
basketball	23.50	23.50	6.50	39.00	19.91 ± 11.29
bicycle	36.00	38.50	11.00	39.00	19.58 ± 10.16
bolt	11.00	11.00	3.00	39.00	20.25 ± 11.38
car	14.50	14.50	14.50	39.00	20.14 ± 9.04
david	12.00	12.00	12.00	39.00	20.21 ± 10.11
diving	4.00	4.00	4.00	34.00	20.48 ± 10.34
drunk	34.00	34.00	15.50	39.00	19.63 ± 8.16
fernando	20.50	29.00	3.50	39.00	20.11 ± 11.09
fish1	30.00	30.00	3.50	39.00	19.73 ± 11.27
fish2	4.00	4.00	2.00	39.00	20.42 ± 11.11
gymnastics	4.50	4.50	4.50	38.50	20.41 ± 11.03
hand1	12.67	12.50	4.50	39.00	20.17 ± 11.17
hand2	29.50	26.00	1.50	39.00	19.72 ± 11.37
jogging	15.50	15.50	2.00	39.00	20.12 ± 10.01
motocross	35.00	38.00	2.00	38.50	19.61 ± 11.04
polarbear	20.00	20.00	20.00	20.00	20.00 ± 0.00
skating	3.50	31.00	3.50	39.00	20.49 ± 11.04
sphere	16.50	34.00	16.50	39.00	20.09 ± 7.70
sunshade	13.00	13.00	13.00	39.00	20.18 ± 9.79
surfing	19.00	19.00	19.00	38.50	20.03 ± 4.41
torus	37.50	38.50	10.00	39.00	19.54 ± 10.43
trellis	8.00	8.00	8.00	38.00	20.32 ± 10.98
tunnel	34.00	35.50	11.00	39.00	19.63 ± 10.33
woman	36.00	38.00	5.50	39.00	19.58 ± 10.98
Average	19.38	21.78	9.04	33.66	20.02 ± 5.26

In Table 6.3, the average performance in terms of accuracy rank is provided for the Region Noise experiment. In the majority of the sequences, VMT lies near the average performance of the 38 trackers. In 8 sequences (ball, basketball - not in Baseline, diving, fish1, fish2, hand1, polarbear, sphere) it performs better than the mean while in 6 sequences (bicycle, car, motocross - not in Baseline, skating, sunshade - not in Baseline, torus) its performs worst than the mean. The last row, which indicates the average performance between the sequences shows that our algorithm has performance near the mean performance of the 38 trackers. The average performance in terms of accuracy rank of VMT in the presence of noise in the initialization is slightly inferior to the performance of the same tracker in the Baseline experiment (Baseline=21.42, Region Noise=21.94.

Table 6.3: Performance of VMT in terms of accuracy rank for the Region Noise experiments (less is better).

Seq.	VMT	GMM	Other Trackers		
			min	max	mean±std
ball	9.50	7.50	1.50	38.00	20.28 ± 11.31
basketball	12.50	17.00	4.50	38.00	20.20 ± 9.87
bicycle	35.00	38.50	2.50	39.00	19.60 ± 8.87
bolt	18.50	18.00	4.00	35.50	20.05 ± 6.91
car	32.00	31.50	2.00	38.50	19.59 ± 10.85
david	20.00	20.50	6.50	34.00	20.00 ± 7.59
diving	8.50	9.50	5.50	33.50	20.32 ± 7.56
drunk	22.00	22.00	1.00	38.50	19.95 ± 10.28
fernando	26.00	23.00	1.00	39.00	19.84 ± 8.12
fish1	8.00	9.00	1.00	35.50	20.32 ± 8.44
fish2	11.00	12.00	1.50	32.00	20.24 ± 7.93
gymnastics	20.00	19.00	5.50	38.00	20.00 ± 5.83
hand1	13.00	13.00	3.50	39.00	20.18 ± 10.60
hand2	25.50	23.67	1.50	36.50	19.84 ± 10.39
jogging	26.50	23.59	12.00	39.00	19.82 ± 7.30
motocross	32.50	33.00	1.50	37.50	19.70 ± 10.55
polarbear	16.50	16.00	2.00	37.50	20.09 ± 10.50
skating	34.00	35.50	9.00	37.50	19.63 ± 7.41
sphere	13.00	27.50	2.50	39.00	20.30 ± 11.13
sunshade	29.50	28.50	8.00	38.50	19.75 ± 9.43
surfing	24.00	29.50	8.00	38.00	19.89 ± 9.28
torus	38.50	38.50	1.00	38.00	19.53 ± 10.97
trellis	21.00	21.50	2.00	37.00	19.97 ± 10.07
tunnel	27.00	27.00	1.50	37.00	19.82 ± 10.67
woman	24.50	23.00	5.00	38.50	19.88 ± 8.85
Average	21.94	22.69	11.74	32.89	19.95 ± 4.71

In Table 6.4, the average performance in terms of robustness rank is provided for the Region Noise experiment. In 11 sequences (ball, car, david, diving, fish2, gymnastics, polarbear, sphere, sunshade, surfing, trellis), VMT has the best or close to the best performance among the 38 trackers. In 7 sequences (basketball, bolt, fernando, fish1, hand1, jogging, skating), VMT has average performance and in 7 sequences (bicycle, drunk, hand2, motocross, torus, tunnel, woman), it is inferior. Compared to the Baseline results, fish1 has improved its accuracy and fish2 has moved from the average performance closer to the best performance. The average performance in terms of robustness rank of VMT in the presence of noise in the initialization is slightly inferior to the performance of the same tracker in the Baseline experiment (Baseline=19.38, Region Noise=19.94).

Table 6.4: Performance of VMT in terms of robustness rank for the Region Noise experiments (less is better).

Seq.	VMT	GMM	Other Trackers		
			min	max	mean±std
ball	13.00	10.00	10.00	38.00	20.20 ± 10.31
basketball	26.00	28.67	2.50	39.00	19.84 ± 10.85
bicycle	37.00	37.50	7.08	39.00	19.56 ± 10.10
bolt	17.00	18.62	1.50	39.00	20.12 ± 11.25
car	15.50	15.50	14.50	39.00	20.12 ± 8.99
david	11.00	13.00	11.00	39.00	20.20 ± 10.11
diving	3.50	3.50	3.50	36.50	20.50 ± 10.84
drunk	32.00	32.00	13.00	39.00	19.69 ± 8.95
fernando	17.44	20.00	1.00	39.00	20.13 ± 10.11
fish1	23.00	22.50	2.33	38.50	19.98 ± 11.18
fish2	4.00	4.75	1.50	38.50	20.51 ± 10.71
gymnastics	4.00	4.00	4.00	38.50	20.41 ± 10.69
hand1	14.00	14.00	4.50	39.00	20.17 ± 11.12
hand2	30.11	28.00	1.50	39.00	19.63 ± 11.15
jogging	26.70	16.00	2.75	39.00	19.85 ± 10.32
motocross	33.38	31.00	2.00	38.50	19.76 ± 10.42
polarbear	19.50	19.50	19.50	38.00	20.01 ± 3.00
skating	12.00	26.60	2.75	39.00	20.44 ± 10.87
sphere	17.50	32.00	17.50	38.50	20.07 ± 6.70
sunshade	9.00	13.00	9.00	38.00	20.29 ± 10.41
surfing	18.50	$33,50$	18.00	39.00	20.04 ± 5.89
torus	38.00	38.00	7.00	38.00	19.58 ± 10.75
trellis	7.00	7.00	7.00	38.00	20.34 ± 10.86
tunnel	36.00	33.00	9.72	38.50	19.81 ± 9.88
woman	33.50	32.00	5.00	39.00	19.70 ± 10.98
Average	19.94	22.27	8.65	33.84	20.04 ± 5.38

Some representative frames from the david, sphere and sunshade image sequences along with their corresponding histograms are presented in Fig. 6.2 - 6.4. In these figures, the first column shows some frames of the corresponding image sequence. The second column shows the corresponding histogram bins (computed from pixels inside the target) and the weighted von Mises mixture (estimated in the first frame and not changing along time) which is indicated by a continuous black line. For demonstration purposes, the histogram bins are normalized to $[0-1]$. In Fig. 6.2, the object to be tracked is in a very low illumination environment at the beginning of the video and moves to a better illuminated place. The initial histogram has one major bin at 0 and some smaller bins for the other values of hue. The mixture model that is formed has two components with positive weights. The first component has its center around 0, so its left

tail gets to the right side of the histogram. As the video proceeds, the major histogram bin at 0 decreases its importance, while other bins with smaller weights become more important.

In Fig. 6.3, the target has a dominant red color, which in the beginning of the sequence is located mainly at the right side of the histogram while at the end the bins are shifted to the right and circularly appear at the left side of the histogram (due to the fact that the Hue component is periodic). Even in these cases, the proposed algorithm successfully tracks the object due to the fact that the von Mises distribution is periodic. More specifically, it assigns a likelihood to the pixels whose colors belong to the right side of the histogram which is sufficient to distinguish the object from the background. Finally, in Fig. 6.4, the object moves from the sun to the sunshade and back. Although during this procedure some histogram bins change their importances significantly, the tracker successfully locates the object.

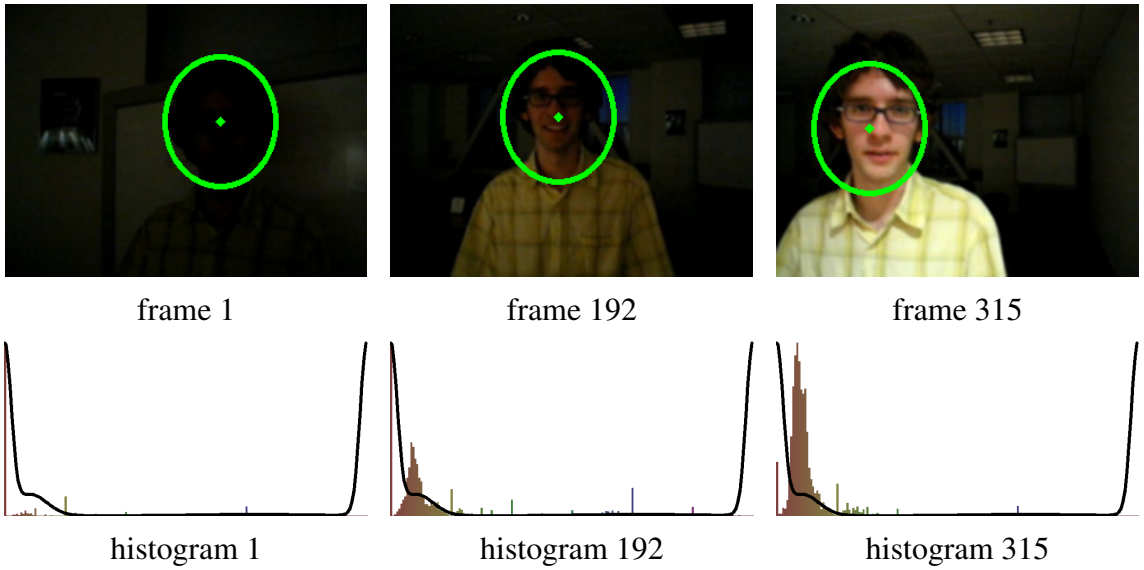


Figure 6.2: Representative frames from the david image sequence (left) and the corresponding histograms with the estimated von Mises mixture superimposed on it (right).

In Table 6.5, the performance of the different initialization strategies is presented. The first column represents the size of the target. The second column is the optimized implementation that is proposed in section 6.3.3 which uses only 360 pixels in order to estimate the parameters of the model using the EM algorithm. The third column is the standard implementation that uses all of the image in order to estimate the same parameters. The time needed by the optimized implementation is relatively constant and around 1 millisecond. This time also includes the time needed in order to create the image with the 360 pixels and its aggregated weights. On the other hand, the time needed by the standard implementation increases as the number of pixels increases.

In Table 6.6, the performance in terms of time for the estimation of each pixel's likelihood is presented. The first column represents the size of the target. The fourth column (indicated by GMM) is an implementation which uses a Gaussian mixture model. Finally, the fifth column (indicated by Hist) is an approach that uses histograms in order to estimate the likelihood for each pixel as described in [30] and employed by the mean shift tracker. The proposed optimized

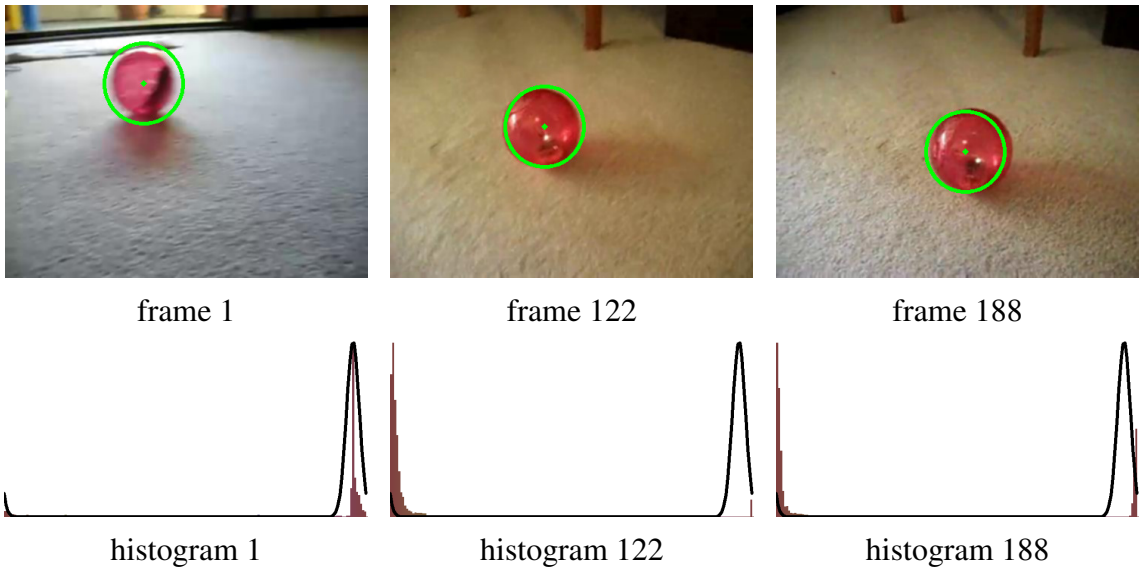


Figure 6.3: Representative frames from the sphere image sequence (left) and the corresponding histograms with the estimated von Mises mixture superimposed on it (right).

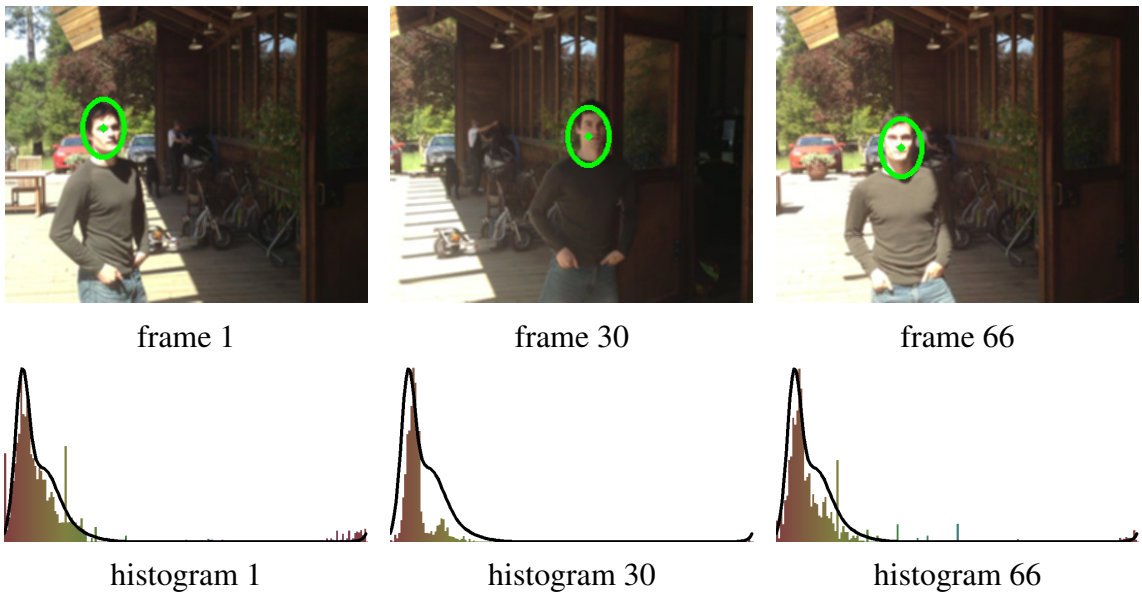


Figure 6.4: Representative frames from the sunshade image sequence (left) and the corresponding histograms with the estimated von Mises mixture superimposed on it (right).

method is around 200 times faster than the straightforward approach that evaluates the exponential and cosine functions in every pixel and 20 times than the approach that uses histograms.

In the experiments above, all mixtures have the same number of components and produce the same results. The evaluation has been performed 10000 times and we present here the mean values. The machine that was used is a laptop with a dual core CPU at 2.26 GHz .

From these experiments, we can underpin some properties of the algorithm: It performs well when the target follows the following assumptions: a) the target may be represented by an ellipsoidal shape and b) the pixels near the center of the ellipse are more important because they

Table 6.5: Comparison of the different initialization approaches that are presented in section 6.3.3. All times are in microseconds (10^{-6} second).

Size	Optimized	Standard	Gain
50×50	742	22004	29
50×100	898	52170	58
50×150	883	76103	86
50×200	746	120294	161
50×250	902	160631	178
50×300	900	190094	211
100×100	904	113634	125
100×150	909	180908	199
100×200	771	277454	359
100×250	771	322151	417
100×300	937	408177	435
150×150	910	291927	320
150×200	793	443965	559
150×250	799	517091	647
150×300	960	614066	639
200×200	803	664858	827
200×250	826	780319	944
200×300	984	825003	838
250×250	837	860095	1027
250×300	1013	1164215	1149
300×300	1042	1311570	1258

are more likely to belong to the target. For example, the torus sequence contains a target with a hole in its center, where background pixels are present. In this case, the weighted von Mises mixture model is trained using mainly pixels from the background, so the tracker can not locate the object. In the motocross sequence, the target contains a rider with its motorbike performing various stunts and it can not be modeled by an ellipse as the area of the target contains many pixels from the background.

Also, when the tracked object is a sphere or the body of a human, it shows better performance in terms of robustness with respect to the other trackers but average performance in terms of accuracy. This means that it does not miss the object, but it can not locate exactly its position, as some interference with the background may affect the result near the boundaries of the ellipse. Moreover, the performance of VMT if not affected significantly when the initialization in the first frame of the image sequence does not contain exactly the target. This can be confirmed by the results of the Baseline and the Region Noise experiments, where the performance both in terms of accuracy and robustness remain nearly the same. Finally, the tracker continues to perform well when the histogram of the color is shifted, like for example in the sphere sequence (Fig. 6.3).

Table 6.6: Comparison of different likelihood estimation approaches presented in section 6.3.3. GMM indicates a Gaussian mixture model and Hist a histogram approach employed in the mean shift algorithm. All times are in microseconds (10^{-6} second).

Size	Optimized	Standard	Gain	GMM	Hist
50×50	3	656	218	544	51
50×100	6	1303	217	1090	98
50×150	9	1953	217	1631	144
50×200	11	2596	236	2174	191
50×250	13	3228	248	2718	237
50×300	16	3888	243	3262	283
100×100	11	2542	231	2013	201
100×150	17	3819	224	3017	296
100×200	22	5083	231	4021	391
100×250	27	6358	235	5028	486
100×300	33	7624	231	6034	581
150×150	25	5834	233	5179	448
150×200	33	7767	235	6903	593
150×250	41	9699	236	8627	736
150×300	48	11624	242	10338	879
200×200	44	10272	233	10223	782
200×250	53	12850	242	12787	973
200×300	64	15415	240	15329	1165
250×250	67	16131	240	15211	1223
250×300	80	19344	241	18250	1461
300×300	96	23144	241	21444	1760

6.5 Conclusions

The proposed algorithm eliminates drawbacks in kernel-based tracking which usually appear in standard applications and are due to periodic shift of the histogram bins of the target. Although some approaches have been proposed to handle this issue for linear spaces [69, 152], these methods can not be directly applied for circular data as the determination of the origin of the axis affect the distance between two points. The VMT method proposed herein, employs the weighted von Mises mixture in order to estimate the target position within a maximum likelihood framework using a gradient based approach. As the von Mises is a continuous distribution, the likelihood is not affected by shifts in the histogram bins of the hue value. Moreover, as the hue values are integers in the interval $[0, 2\pi)$, the pre-calculation of key quantities of the likelihood of the mixture model, both in terms of computational time and memory. Although VMT uses the hue values, it could be used with other circular data, like the angle of the image gradient.

CHAPTER 7

MOTION SEGMENTATION AND TRACKING BY CLUSTERING INCOMPLETE TRAJECTORIES

7.1 Introduction

In this chapter, we present a novel framework for visual target tracking based on model-based clustering trajectories of key points (Fig. 7.1). The proposed method creates trajectories of image features (e.g *Harris corner* [48]). However, key point tracking introduces an additional difficulty since the resulting feature trajectories have a short duration, as the key points disappear and reappear due to occlusion, illumination and viewpoint changes. Therefore, we are dealing with time-series of variable length. Motion segmentation is then converted into a clustering problem of these input trajectories, in a sense of grouping together feature trajectories that belong to the same object. For this purpose, we use an efficient regression mixture model, which has three significant features: a) Sparse properties over the regression coefficients, b) it is allowed to be translated in measurement space and c) its noise covariance matrix is diagonal and not spherical as it is commonly used. The above properties are incorporated through

a Bayesian regression modeling framework, where the EM algorithm can be applied for estimating the model parameters. Special care is given for initializing the EM algorithm where an interpolating scheme is proposed based on executing successively the k-means algorithm over the duration of trajectories. Experiments show the robustness of our method to occlusions and highlight its ability to discover better the objects motion in comparison with other approaches and the Hopkins 155 dataset [115].

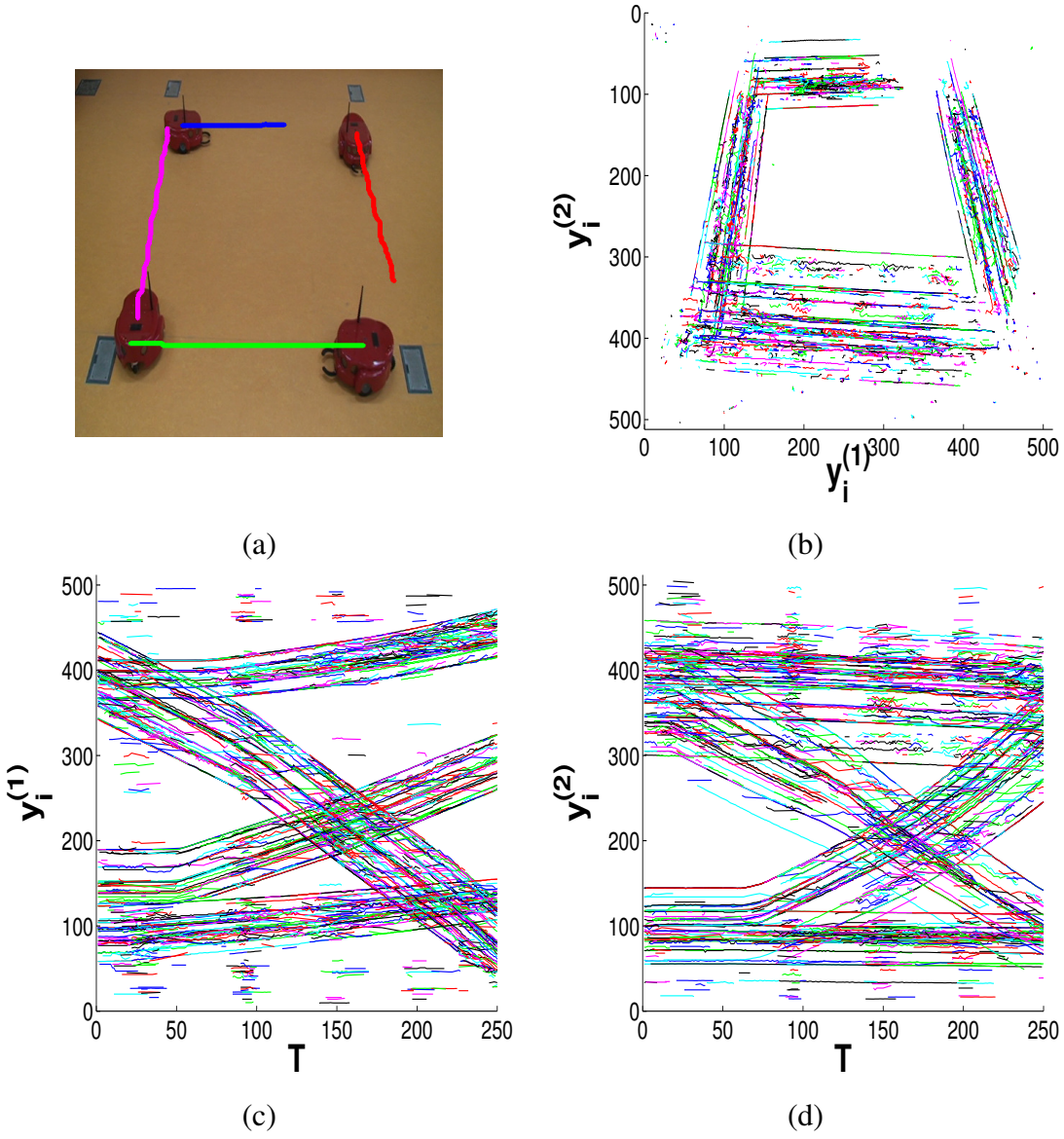


Figure 7.1: Trajectories extracted from an image sequence. (a) The first frame of the image sequence showing four robots and their mean trajectories. The group of robots perform a square-like movement. (b) The trajectories of the features extracted from the image sequence. The two axes represent the horizontal and vertical coordinates. (c) The horizontal trajectories along time. (d) The vertical trajectories along time. Notice that there is a large number of short and incomplete trajectories because the features disappear and reappear during the image sequence due to illumination changes and the distance of the object from the camera.

The rest of the chapter is organized as follows: the procedure of feature extraction and tracking in order to create the trajectories is presented in section 7.2. The trajectories clustering algorithm is presented in section 7.3, experimental results are shown in section 7.4 and conclusions is drawn in section 7.5.

7.2 Extracting trajectories

Trajectories are constructed by tracking points in each frame of the image sequence. The main idea is to extract some salient points from a given image and associate them with points from previous images. To this end, we employ the so called *Harris corners* [48] and standard optical flow for the data association step [78]. Let us notice that any other scale or affine invariant features [77, 86] would also be appropriate. In this section, we use Harris corner features due to their simplicity, as they rely on the eigenvalues of the matrix of the second order derivatives of the image intensities.

Let T be the number of image frames and $\boldsymbol{Y} = \{\boldsymbol{y}_i\}_{i=1,\ldots,N}$ be a list of trajectories with N being unknown beforehand. Each individual trajectory $\boldsymbol{y}_i$ consists of a set of 2D points, the time of appearance of its corner point into the trajectory, (i.e. the number of the image frame) and the optical flow vector of the last point in the list.

Initially, the list $\boldsymbol{Y}$ is empty. In every image frame, Harris corners are detected and the optical flow vector at each corner is estimated [105]. Then, each corner found in the current image frame is attributed to a trajectory that already exists, or a new trajectory is created having only one element, the corner under consideration. According to this scheme, three cases are possible:

- If any key point of the previous frame has an optical flow vector pointing out the key point under consideration, then the current corner is attributed to an existing trajectory. In this case, a trajectory follows the optical flow displacement vector, meaning that the corner is apparent in consecutive frames.

- If there is no such key point in the previous frame, we apply a window around the last corner which is similar to the current corner. If there are more than one similar corners then we choose the closest one.

- Otherwise, a new trajectory is constructed containing only the corner under consideration.

In Fig. 7.2, an intuitive example is presented where three corners are considered for demonstration purposes and five time instances are depicted. In the first frame, three corners are detected and three trajectories are created. In the second frame, the same corners are detected and associated with existing trajectories due to optical flow constraint. Next, one corner is detected and attributed to an existing trajectory due to optical flow constraints while the other two key points are occluded. During the fourth frame, the key point that was not occluded is also detected and attributed to an existing trajectory. One of the other two corner points that reappear

is attributed to a trajectory due to local window matching. The other corner is not associated with any existing trajectory, so a new trajectory is created. In the last frame, three corners are detected and associated with existing trajectories due to optical flow. Thus, four trajectories have been created: two trajectories corresponding to the same key point and two additional trajectories involving two distinct key points.

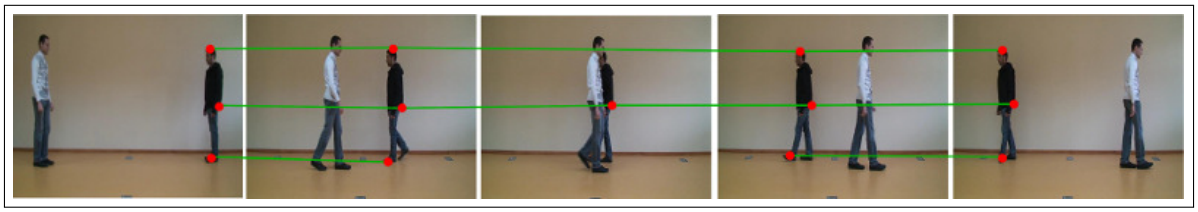


Figure 7.2: Example of trajectories construction. The red dots represent the image key points and the green lines represent their trajectories. The figure is better seen in color.

Once the list $\boldsymbol{Y}$ is created, the trajectories of the corner points that belong to the background are eliminated. This is achieved by removing the trajectories having a small variance, according to a predefined-threshold value, as well as the trajectories of small length (e.g. 1% of the number of frames). The complete procedure is described in Algorithm 7.

Algorithm 7 Trajectories construction algorithm

function CREATETRAJECTORIES($\boldsymbol{Im}$)
  Input: an image sequence $\boldsymbol{Im}$.
  Output: a list of trajectories $\boldsymbol{Y}$.
  $\boldsymbol{Y} = \emptyset$.
  for every image $\{im^{(t)}\}_{t=1,...,T}$ **do**
    Detect corners $\{c_l^{(t)}\}_{l=1,...,L^{(t)}}$ and estimate optical flow $\{f_l^{(t)}\}_{l=1,...,L^{(t)}}$ in each one.
    for every corner $\{c_l^{(t)}\}_{l=1,...,L^{(t)}}$ detected in $im^{(t)}$ **do**
      if $\boldsymbol{y}_i$ has its last corner c_i^{last} in the image $t-1$ and its optical flow f_i^{last} points to the current corner, i.e. $c_i^{last} + f_i^{last} \approx c_l^{(t)}$ **then**
        Insert $c_l^{(t)}$ into $\boldsymbol{y}_i$.
      else if $\boldsymbol{y}_i$ has its the window around its last corner c_i^{last} similar to the window around thw current corner $c_l^{(t)}$ **then**
        Insert $c_l^{(t)}$ into $\boldsymbol{y}_i$.
      else
        Insert a new trajectory $\boldsymbol{y}_i$ with only $c_l^{(t)}$ into $\boldsymbol{Y}$.
      end if
    end for
  end for
  Eliminate trajectory $\boldsymbol{y}_i$ with small variation in its corners coordinates.
end function

7.3 Clustering trajectories of variable length

Suppose we have a set of trajectories of N tracked feature points over T frames obtained from the previous procedure. The aim of tracking is to detect K independently moving objects in the scene and simultaneously estimate their characteristic motion. This can be seen as a clustering problem.

A 2D trajectory $\boldsymbol{y}_i = (\boldsymbol{y}_i^{(1)}, \boldsymbol{y}_i^{(2)})$ consists of two directions: (1) horizontal and (2) vertical. It is defined by a set of T_i points $\{(y_{i1}^{(1)}, y_{i1}^{(2)}), \ldots, (y_{iT_i}^{(1)}, y_{iT_i}^{(2)})\}$, corresponding to T_i image positions $(t_{i1}, \ldots, t_{iT_i})$ of the image sequence. It is important to note that we deal with trajectories of variable length T_i since occlusions or illumination changes may block the view of the objects in certain image frames.

Linear regression model constitutes a powerful platform for modeling sequential data that can be adopted in our case. In particular, we assume that a trajectory $\boldsymbol{y}_i^{(j)}$ of any direction $j = \{1, 2\}$ can be modeled through the following functional form:

$$\boldsymbol{y}_i^{(j)} = \Phi_i \boldsymbol{w}^{(j)} + d_i^{(j)} + \boldsymbol{e}_i^{(j)} \,, \tag{7.1}$$

where Φ_i is the design kernel matrix (common for both directions) of size $T_i \times T$, and $\boldsymbol{w} = (w_1, \ldots, w_T)$ is the vector of the T unknown regression coefficients.

At first, we assume that the input space consists of the time instances $(t_1, \ldots, t_T)$ which correspond to the T frames of the image sequence. Having that in mind, equation (1) is the standard linear regression model [14] with a global translation term added, where $\boldsymbol{y}_i^{(j)}$ is a vector of length T_i containing the values of the i-th observation of the horizontal ($j = 1$) and vertical ($j = 2$) directions. The i-th design matrix Φ_i is generated by keeping the T_i rows of the global design matrix Φ that correspond to time instances $(t_{i1}, \ldots, t_{iT_i})$, where the i-th key point exists. The global matrix Φ is a kernel matrix of size $T \times T$ with elements calculated by a kernel function, i.e. $\Phi(k, l) = k(t_k, t_l)$, where $t_k, t_l \in \{t_1, \ldots, t_T\}$. In this chapter, we considered the mexican hat wavelet kernel $k(t_k, t_l) = \frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}} \left(1 - \frac{(t_l - t_k)^2}{\sigma^2}\right) e^{\frac{-(t_l - t_k)^2}{2\sigma^2}}$, though any other kernel function could be used (e.g. Gaussian).

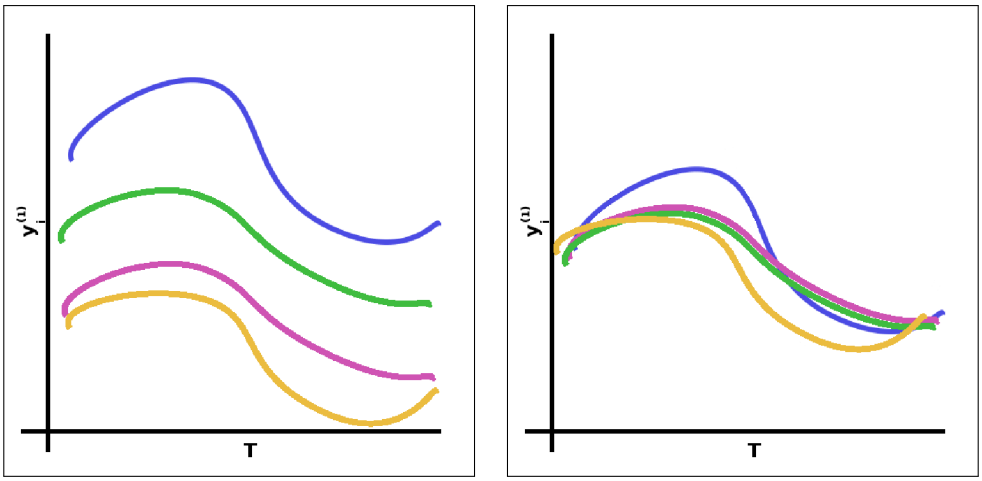
Also, in the above equation, we assume a translation scalar term $d_i^{(j)}$ that allows for the entire trajectory to be translated globally [43], see Fig. 7.3. Incorporating such term results in a regression model that allows for arbitrary translations in measurement space. In our case, the features are distributed around the edges of the objects and the trajectories of the key points are translated in order to be aligned with the trajectory of the center of gravity of the object. Finally, the error term $\boldsymbol{e}_i^{(j)}$ in the above formulation is a T_i-dimensional vector that is assumed to be zero-mean Gaussian and independent over time:

$$e_i \sim \mathcal{N}(0, \Sigma_i), \tag{7.2}$$

with a diagonal covariance matrix $\Sigma_i^{(j)} = diag(\sigma_{t_{i1}}^{2(j)}, \ldots, \sigma_{t_{iT_i}}^{2(j)})$. More specifically, Σ_i is a block matrix of a $T \times T$ diagonal covariance matrix that corresponds to the noise variance of T frames.

Under these assumptions, the conditional density of a trajectory is also Gaussian:

$$p(\boldsymbol{y}_i^{(j)}; \boldsymbol{w}^{(j)}, \Sigma_i^{(j)}, d_i^{(j)}) = \mathcal{N}(\Phi_i \boldsymbol{w}^{(j)} + d_i^{(j)}, \Sigma_i^{(j)}) \,. \tag{7.3}$$



(a) (b)

Figure 7.3: The effect of the translation parameter. (a) A set of trajectories. (b) Alignment of the trajectories.

Moreover, we consider the scalar $d_i^{(j)}$ to be a zero-mean Gaussian variable with variance $u^{(j)}$:

$$d_i^{(j)} \sim \mathcal{N}(0, u^{(j)}). \tag{7.4}$$

Thus, we can further integrate out d_i to obtain the marginal density for $\boldsymbol{y}_i^{(j)}$ which is also Gaussian:

$$p(\boldsymbol{y}_i^{(j)}; \theta) = \int p(\boldsymbol{y}_i^{(j)}; \boldsymbol{w}^{(j)}, \Sigma_i^{(j)}, d_i^{(j)}) p(d_i^{(j)}) dd_i^{(j)} = \mathcal{N}(\Phi_i \boldsymbol{w}^{(j)}, \Sigma_i^{(j)} + u^{(j)} \mathbb{1}) \,, \tag{7.5}$$

where $\mathbb{1}$ is a matrix of 1's of size $T_i \times T_i$. The marginal density is implicitly conditioned on the parameters $\boldsymbol{\theta} = \{\boldsymbol{w}, u, \Sigma\}$.

In our study we consider that every object k can be described by a unique functional regression model, as given by the set parameters $\boldsymbol{\theta}_k = \{\boldsymbol{\theta}_k^{(1)}, \boldsymbol{\theta}_k^{(2)}\}$, where $\boldsymbol{\theta}_k^{(j)} = \{\boldsymbol{w}_k^{(j)}, u_k^{(j)}, \Sigma_k^{(j)}\}$, that fits to all trajectories belong to this object. Therefore, the objective is to dividing the set of N trajectories into K clusters. This can be described by the following regression mixture model:

$$p(\boldsymbol{y}_i; \boldsymbol{\Theta}) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{y}_i; \boldsymbol{\theta}_k) = \sum_{k=1}^{K} \pi_k p(\boldsymbol{y}_i^{(1)}; \boldsymbol{\theta}_k^{(1)}) p(\boldsymbol{y}_i^{(2)}; \boldsymbol{\theta}_k^{(2)}) \,, \tag{7.6}$$

where we have assumed independence between that trajectories of both directions $(\boldsymbol{y}_i^{(1)}, \boldsymbol{y}_i^{(2)})$. In addition, π_k are the mixing weights satisfying the constraints $\pi_k \geq 0$ and $\sum_{k=1}^{K} \pi_k = 1$, while $\boldsymbol{\Theta}$ is the set of all the unknown mixture parameters, $\boldsymbol{\Theta} = \{\pi_k, \boldsymbol{\theta}_k\}_{k=1}^{K}$.

An important issue, when dealing with regression models is how to determine their order. Models of small order can lead to under-fitting, while larger order lead to curve over-fitting. Both cases may cause to serious deterioration of the clustering performance. Sparse modeling [114] offers a significant solution to this problem by employing models having initially many degrees of freedom than could uniquely be adapted given data. Sparse Bayesian regression can be achieved through a hierarchical prior definition over regression coefficients

$\boldsymbol{w}_k^{(j)} = (w_{k1}^{(j)}, \ldots, w_{kT}^{(j)})^T$. In particular, we assume first that coefficients follows a zero-mean Gaussian distribution:

$$p(\boldsymbol{w}_k^{(j)}; \boldsymbol{\alpha}_k^{(j)}) = \mathcal{N}(\boldsymbol{w}_k^{(j)}; \mathbf{0}, \boldsymbol{A}_k^{-1^{(j)}}) = \prod_{l=1}^{T} \mathcal{N}(w_{kl}^{(j)}; 0, \alpha_{kl}^{-1^{(j)}}) \tag{7.7}$$

where $\boldsymbol{A}_k^{(j)}$ is a diagonal matrix containing the T elements of precisions $\boldsymbol{\alpha}_k^{(j)} = (\alpha_{k1}^{(j)}, \ldots, \alpha_{kT}^{(j)})^T$. We impose next a Gamma prior on these hyperparameters:

$$p(\boldsymbol{\alpha}_k^{(j)}) = \prod_{l=1}^{T} Gamma(\alpha_{kl}^{(j)}|a, b) \propto \prod_{l=1}^{T} {\alpha_{kl}^{(j)}}^{a-1} \exp(-b\alpha_{kl}^{(j)}) \,, \tag{7.8}$$

where a and b denote parameters that are *a priori* set to values near zero. The above two-stage hierarchical prior is actually a Student's t-distribution [114]. This is a heavy tailed prior distribution that enforces most of the values $\alpha_{kl}^{(j)}$ to be large, thus the corresponding $w_{kl}^{(j)}$ are set to zero and eliminated from the model. In this way, the complexity of the regression model is controlled in an automatic and elegant way and over-fitting is avoided.

Now, the clustering procedure has been converted into a maximum *a posteriori* (MAP) estimation approach, in a sense of estimating the mixture model parameters that maximize the MAP log-likelihood function given by:

$$L(\boldsymbol{\Theta}) \;=\; \sum_{i=1}^{N} \log\{\sum_{k=1}^{K} \pi_k p(\boldsymbol{y}_i; \boldsymbol{\theta}_k)\} + \sum_{k=1}^{K} \sum_{j=1}^{2} \{\log p(\boldsymbol{w}_k^{(j)}; \boldsymbol{\alpha}_k^{(j)}) + \log p(\boldsymbol{\alpha}_k^{(j)})\} \,. \tag{7.9}$$

The model can be trained using the Expectation - Maximization (EM) algorithm [33] that iteratively performs two main stages: The *E-step* where the expected values of the hidden variables are calculated. In our case, this includes the cluster labels of trajectories as given by the posterior probabilities:

$$z_{ik} = P(k; \boldsymbol{y}_i) = \frac{\pi_k p(\boldsymbol{y}_i; \boldsymbol{\theta}_k)}{\sum_{k'} \pi_{k'} p(\boldsymbol{y}_i; \boldsymbol{\theta}_{k'})} \,, \tag{7.10}$$

as well as the mean value of the translations $d_{ik}^{(j)}$ at any direction. The latter is obtained by using the fact that the posterior density of translations is also Gaussian:

$$p(d_{ik}^{(j)}; \boldsymbol{y}_i^{(j)}, k) \propto p(\boldsymbol{y}_i^{(j)}; \boldsymbol{\theta}_k^{(j)}) p(d_{ik}^{(j)}) = \mathcal{N}(\hat{d}_{ik}^{(j)}, V_{ik}^{(j)}), \tag{7.11}$$

where

$$\hat{d}_{ik}^{(j)} = V_{ik}^{(j)} \left(\boldsymbol{y}_i^{(j)} - \Phi_i \boldsymbol{w}_k^{(j)}\right)^T \boldsymbol{\Sigma}_{ik}^{-1^{(j)}} \mathbf{1}_i \;\text{ and }\; V_{ik}^{(j)} = \left(\mathbf{1}_i^T \boldsymbol{\Sigma}_{ik}^{-1^{(j)}} \mathbf{1}_i + \frac{1}{u_k^{(j)}}\right)^{-1} \,, \tag{7.12}$$

where $\mathbf{1}_i$ is a T_i-length vector of 1's.

During the *M-step*, the maximization of the expected value of the complete log-likelihood function (referred as Q-function in the machine learning bibliography [43]) is performed:

$$\begin{aligned} Q(\theta^t, \theta^{t-1}) &= \sum_{i=1}^{N}\sum_{k=1}^{K} z_{ik} \left\{ \log \pi_k + \sum_{j=1}^{2} -\frac{1}{2}\log|\Sigma_k^{(j)}| - \frac{\left(\boldsymbol{y}_i^{(j)} - \boldsymbol{\mu}_k^{(j)}\right)^T \Sigma_k^{(j)} \left(\boldsymbol{y}_i^{(j)} - \boldsymbol{\mu}_k^{(j)}\right)}{2} \right\} \\ &+ \sum_{k=1}^{K}\sum_{j=1}^{2} -\frac{1}{2}\log|A_k^{(j)}| - \frac{\boldsymbol{w}_k^{(j)^T} A_k^{(j)} \boldsymbol{w}_k^{(j)}}{2} + \sum_{k=1}^{K}\sum_{j=1}^{2}\sum_{l=1}^{T} \log \alpha_{kl}^{(j)^{a-1}} - b\alpha_{kl}^{(j)} \end{aligned} \quad (7.13)$$

Maximizing (7.13) with respect to the model parameters leads to the following update rules [43, 17]:

$$\hat{\pi}_k = \frac{\sum_{i=1}^{N} z_{ik}}{N}\,, \quad (7.14)$$

$$\hat{\boldsymbol{w}}_k^{(j)} = \left[\sum_{i=1}^{N} z_{ik}\Phi_i^T \boldsymbol{\Sigma}_{ik}^{-1^{(j)}} \Phi_i + \boldsymbol{A}_k^{(j)}\right]^{-1} \sum_{i=1}^{N} z_{ik}\Phi_i^T \boldsymbol{\Sigma}_{ik}^{-1^{(j)}} (\boldsymbol{y}_i^{(j)} - \hat{d}_{ik}^{(j)})\,, \quad (7.15)$$

$$\alpha_{kl}^{(j)} = \frac{1+2a}{\hat{w}_{kl}^{2(j)} + 2b},\ \forall\, l = 1, \ldots, T\,, \quad (7.16)$$

$$\hat{\sigma}_{kl}^{2(j)} = \frac{\sum_{i=1}^{N} z_{ik} \left\{ \left(\boldsymbol{y}_{il}^{(j)} - [\Phi_i \hat{\boldsymbol{w}}_k^{(j)}]_l - \hat{d}_{ik}^{(j)}\right)^2 + V_{ik}^{(j)} \right\}}{\sum_{i=1}^{N} z_{ik}}\,,\ \forall\, l = 1, \ldots, T\,. \quad (7.17)$$

$$\hat{u}_k^{(j)} = \frac{\sum_{i=1}^{N} z_{ik} \left(\hat{d}_{ik}^{2(j)} + V_{ik}^{(j)}\right)}{\sum_{i=1}^{N} z_{ik}}\,. \quad (7.18)$$

where $[.]_l$ indicates the l-th component of the T_i-dimensional vector that corresponds to time location t_{il}.

After convergence of the EM algorithm, two kinds of information are available: At first, the cluster labels of the trajectories are obtained according to the maximum posterior probability value (Eq. 7.10). Moreover, the motion of objects are obtained from the predicted mean trajectories per cluster, i.e. $\boldsymbol{\mu}_k = (\boldsymbol{\mu}_k^{(1)}, \boldsymbol{\mu}_k^{(2)}) = (\Phi \boldsymbol{w}_k^{(1)}, \Phi \boldsymbol{w}_k^{(2)})$.

7.3.1 Initialization Strategy

A fundamental issue when applying the EM algorithm, is its strong dependence on the initialization of the model parameters due to its local nature. Improper initialization may lead to reaching poor local maxima of the log-likelihood, a fact that may affect significantly the performance of the method. A natural way for initialization is by randomly selecting K samples from the set of input trajectories, one for each cluster. Then, we can obtain the least-squares solution for initializing the regression coefficients. In addition, the noise variance Σ_k is initialized by a small percentage of the total variance of all trajectories equally for each clusters, while we set the mixing weights π_k equal to $1/K$. Finally, one step of the EM algorithm is executed for further refining these parameters and calculating the MAP log-likelihood function. Several

such different trials are executed and the solution with the maximum MAP likelihood function is selected for initializing model parameters.

However, the above scheme cannot be easily applied to our approach due to the large variability in length (T_i) of the input trajectories which brings a practical difficulty in obtaining the least-squared solution. For this reason, we have followed a more robust initialization scheme based on the interpolation among successive time steps. More specifically, starting from the first time step we perform periodically (e.g. at every $5T\%$ frames) the k-means clustering over the points $(y_{it}^{(1)}, y_{it}^{(2)})$. Then, the resulting K centers are associated with those of the previous time according to the minimum distance criterion. Finally, a linear interpolation (per cluster) is performed and thus the resulting K curves are used for initializing the parameters of the K clusters. It must be noted that in cases where there is a large number of features representing the background, the initialization may diverge from the desired solution since the existence of a significant amount of outliers affects the k-means solution. Even if during our experiments we have not faced with any such problem, treating with this situation still remains a future plan of study. An example of the proposed process is given in Fig. 7.4 adopted from an experimental data set, where both the initial interpolated trajectories (Fig. 7.4c) and the final clustering solution are shown (Fig. 7.4d).

7.4 Experimental results

We have evaluated the performance of our approach using both simulated and real examples. Demonstration videos are available at http://www.cs.uoi.gr/ ∼vkaravas/motion_segmentation_and _tracking. Some implementation details of our method are the following: At first, we have normalized spatial and temporal coordinates into the interval $[0, 1]$ while the extracted trajectories either with length less than 1% of the number of frames T, or with variance less than 0.01 were not taken into account. We have selected the mexican hat wavelet kernel $k(t_k, t_l) = \frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}}\left(1 - \frac{(t_l - t_k)^2}{\sigma^2}\right) e^{\frac{-(t_l - t_k)^2}{2\sigma^2}}$ for the design kernel matrix Φ, where the scalar parameter took the value of $\sigma = 0.3$. Experiments have shown that the performance of our approach is not very sensitive to this parameter, since we took similar results for values in the range of $[0.1, 0.5]$.

Comparison has been made with the mean shift algorithm [30], the camshift algorithm [20] and the Kalman filter [32], which are established algorithms in visual tracking. For the mean shift algorithm, the images were represented in the RGB color space using histograms of 16 bins for each component. For the camshift algorithm, the hue component of HSV color space was used to construct a 16-bin histogram. For the Kalman filter, camshift was used in order to provide measurements (the position and the size of the object). For initializing all these algorithms, we have manually selected the position and the size of each object in the first frame of the image sequence. Then, the objects are tracked, using a distinct tracker per each target. The centers of the ellipses surrounding the targets are used to construct the mean trajectory of each object. This comparison favors these algorithms in cases where the features are not

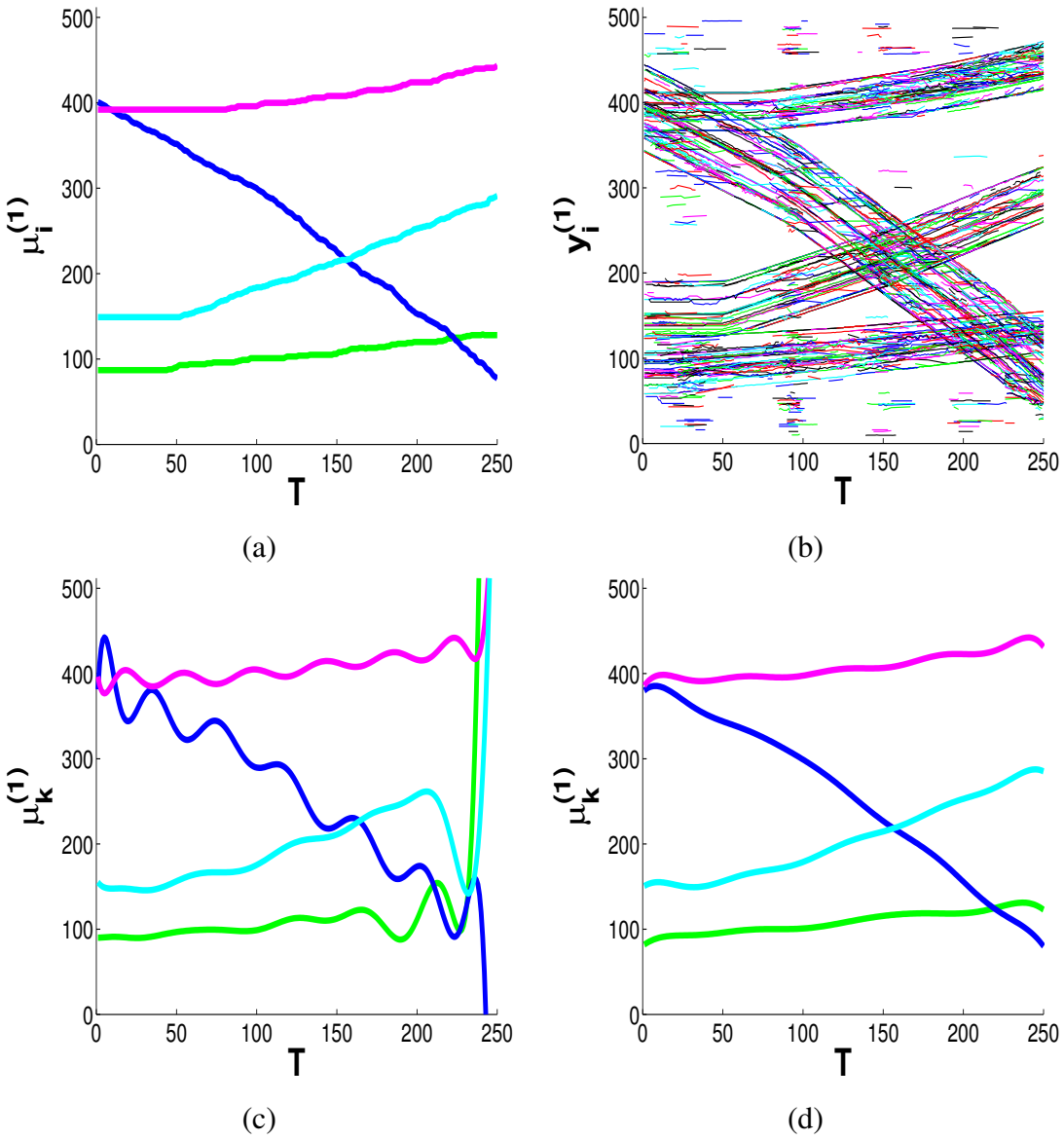


Figure 7.4: The overall progress of our method applied to an experimental image sequence of 250 images with $k = 4$ objects with different motions. (a) Real trajectories, (b) input trajectories, (c) initial estimation of mean trajectories using the proposed technique, (d) the estimated trajectories after EM convergence.

uniformly distributed around the object, as the center estimated by the features may vary from the geometric center of the object (Fig. 7.5).

7.4.1 Experiments with simulated data sets

The first series of experiments involves seven (7) simulated image sequences with spheres moving in predefined directions as shown in Fig. 7.6 and Fig. 7.7. Each image sequence contains 130 frames of size 512×512. The value of N varies from 1500 to 2000 trajectories per case, with average length around $T = 60$ frames each trajectory. In the first five problems, no occlu-

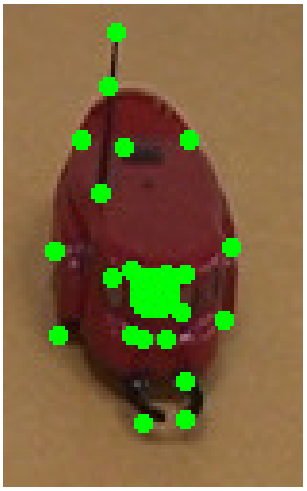


Figure 7.5: Features are not uniformly distributed over the object and the center of gravity of the key points does not coincide with the center of gravity of the object. The small dots represent the features and the big dot represents their barycenter. The figure is better visualized in color.

sions are simulated (all objects are visible in every frame). In the rest two problems, occlusion is happened, where in *Sim6* a sphere disappears while in *Sim7* a sphere disappears and reappears.

Since we are aware of the ground truth, the performance of all tracking approaches was evaluated using two criteria:

- The mean squared error (MSE) measured in pixels, between the ground truth $\boldsymbol{r}$ and the estimated mean trajectories $\boldsymbol{\mu}$ as given by

$$MSE = \frac{1}{K \cdot T} \sum_{k=1}^{K} \sum_{j=1}^{2} ||r_k^{(j)} - \mu_k^{(j)}||^2 \, .$$

- The accuracy (ACC) that counts the percentage of correctly classified trajectories. It must be noted that the input trajectories created by our method have been also chosen to evaluate the mean shift algorithm, the camshift algorithm and the Kalman filter tracker. In particular, we assign every input trajectory to an object according to the smallest distance with the predicted mean trajectory.

The depicted results are presented in Table 7.1. As it is obvious, we took comparable results in the first thee approaches *Sim1*, *Sim2* and *Sim3*. However, in the fourth problem (*Sim4*), our approach and mean shift successfully track the objects, while camshift and the Kalman filter are failed. This is happened in the case of camshift due to the fact that the objects are overlapping in the initial frame, and after some iterations the result is an ellipse with its center located at the center of the image and whose size is increased in order to contain all the objects inside it. Moreover, Kalman filter fails because it uses camshift to obtain the measurements. On the other hand, mean shift tracks the objects due to the fact that in this implementation we decided not to integrate a scale change, as in camshift. In problem *Sim5*, camshift and Kalman filter fail again, because two objects (the one with the green trajectory and the one with the blue trajectory in

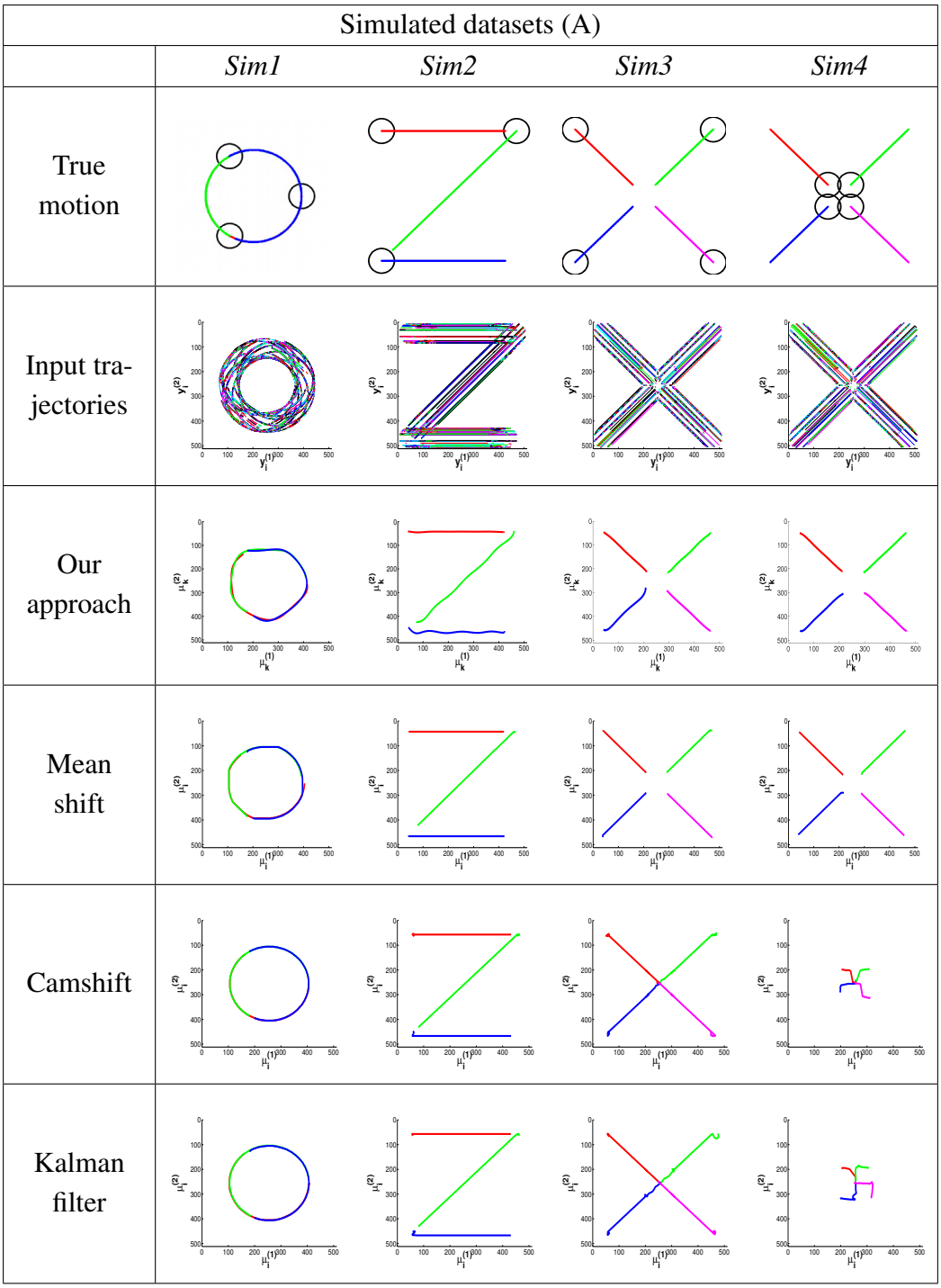


Figure 7.6: Comparative results with four artificial datasets. For each problem we give the true objects motion, the created input trajectories and the estimated motion by all approaches.

Fig. 7.7) pass near each other and camshift makes the target ellipse bigger in order to include both objects. In problem *Sim6* one object suddenly disappears (the one with the red trajectory in Fig. 7.7). Finally, in problem *Sim7* the object disappears and reappears in another position after some time. Mean shift fails to track the sphere that disappears and reappears and tracks the object only as long as it is visible. Camshift and Kalman filter, as they take into account

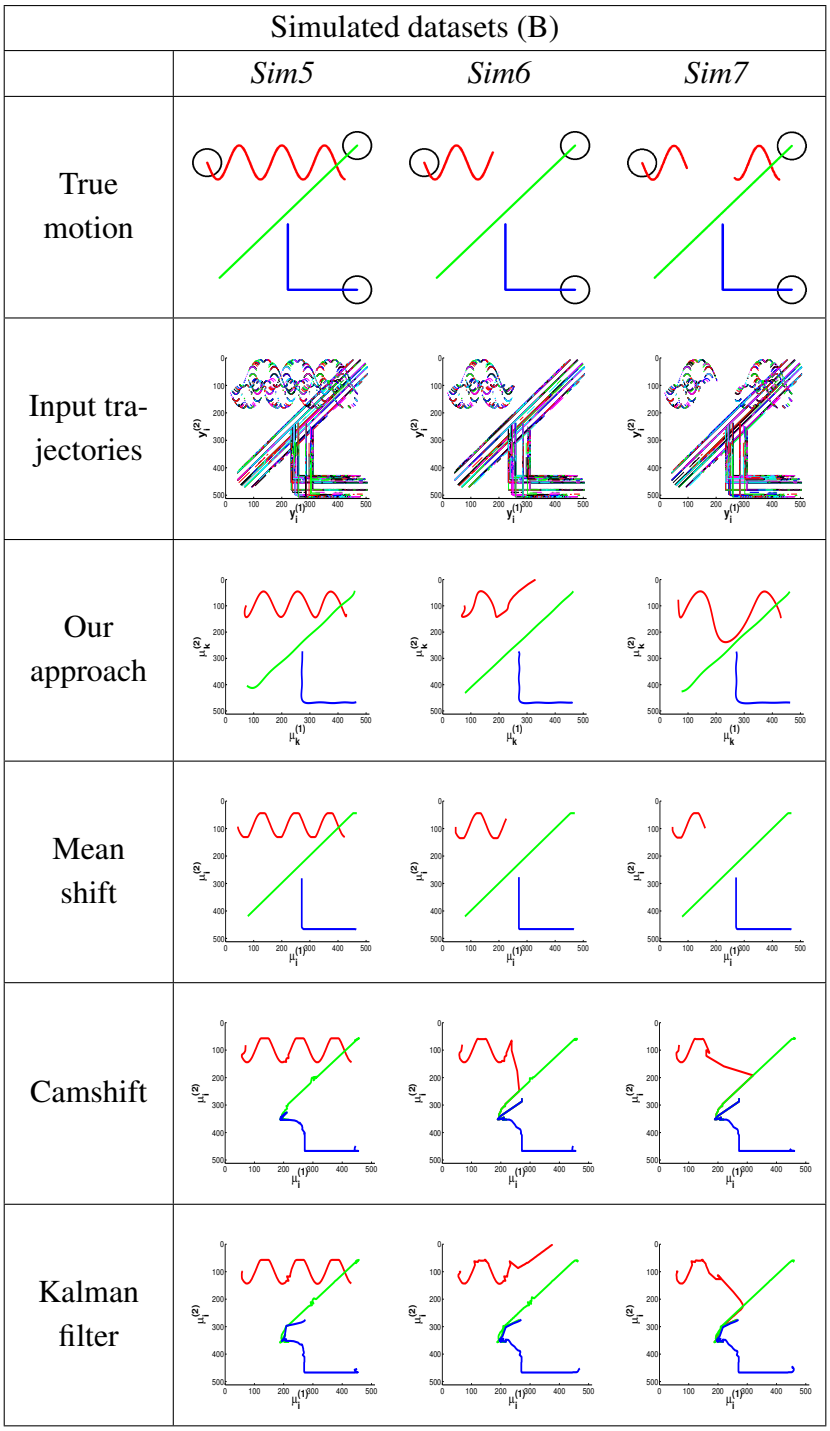


Figure 7.7: Comparative results with three artificial datasets. For each problem we give the true objects motion, the created input trajectories and the estimated motion by all approaches.

scale changes, increase the size of the ellipse and track a nearby object. On the other hand, the proposed method correctly associates the two separated trajectories of the sphere. Finally, it must be noted that in the case of both problems *Sim6* and *Sim7*, the frames in which a sphere is not visible are not taken into account for computing the MSE criterion.

Table 7.1: The performance of the compared methods in terms of classification accuracy (ACC) and mean squared error (MSE).

Problem	Our approach		Mean shift		Camshift		Kalman	
	MSE	ACC	MSE	ACC	MSE	ACC	MSE	ACC
Sim1	69	**100%**	121	**100%**	4	96%	**0**	**100%**
Sim2	**10**	99%	114	**100%**	55	**100%**	54	**100%**
Sim3	**10**	96%	114	**99%**	202	95%	224	95%
Sim4	**15**	97%	130	**99%**	lost	lost	lost	lost
Sim5	**20**	**100%**	118	**100%**	lost	lost	lost	lost
Sim6	**29**	**100%**	74	**100%**	lost	lost	lost	lost
Sim7	**41**	**99%**	lost	lost	lost	lost	lost	lost

7.4.2 Experiments with real data sets

We have also evaluated our motion segmentation approach using five (5) real datasets shown in Fig. 7.8 containing images of size 512×512. All of these videos were created in our laboratory and they may be downloaded at http://www.cs.uoi.gr/ ∼vkaravas/motion_segmentation_and _tracking. The first three of them (*Real1-3*) show mobile robots moving in various directions: In *Real1* ($T = 250$), the robots are moving towards the borders of the image forming the vertices of a square. In *Real2* ($T = 680$), the robots are moving around the center of the image forming a circle, and finally in *Real3* ($T = 500$), the robots are moving forward and backward. The rest two datasets (*Real4-5*) show two persons walking, and so occlusion take place as one person gets behind the other. In particular, in *Real4* ($T = 485$) two persons are moving from one side of the scene to the other and backwards, while in *Real5* ($T = 635$) the persons additionally move forward and backward in the scene.

As ground truth is not provided in these cases we have evaluated our approach only visually. In problems *Real1-3* all algorithms produce approximately the same trajectories. On the contrary, in the cases of *Real4* and *Real5* problems, where we deal with articulated objects and occlusion, mean shift and camshift fail to properly track the persons and one of them is lost. Moreover, the trajectory of the center of the ellipse (which represents the object in mean shift and camshift) is not smooth. This is due to the change in the appearance of the target. When the person walks, there are frames where both arms and legs of a person are visible and instances where only one of them is present. In these cases, mean shift and camshift may produce abrupt changes in motion estimation because the person in the frame has change its appearance with respect to its appearance in the first frame (which is used to initialize the model representing the object). On the other hand, our method is more accurate and produces a smooth trajectory.

Looking in more detail the problem *Real4*, we can see the person in black disappears (because he gets behind the other person) twice during this image sequence: at first, when he is moving from right to left, and then, as he is moving from left to right. Mean shift successfully

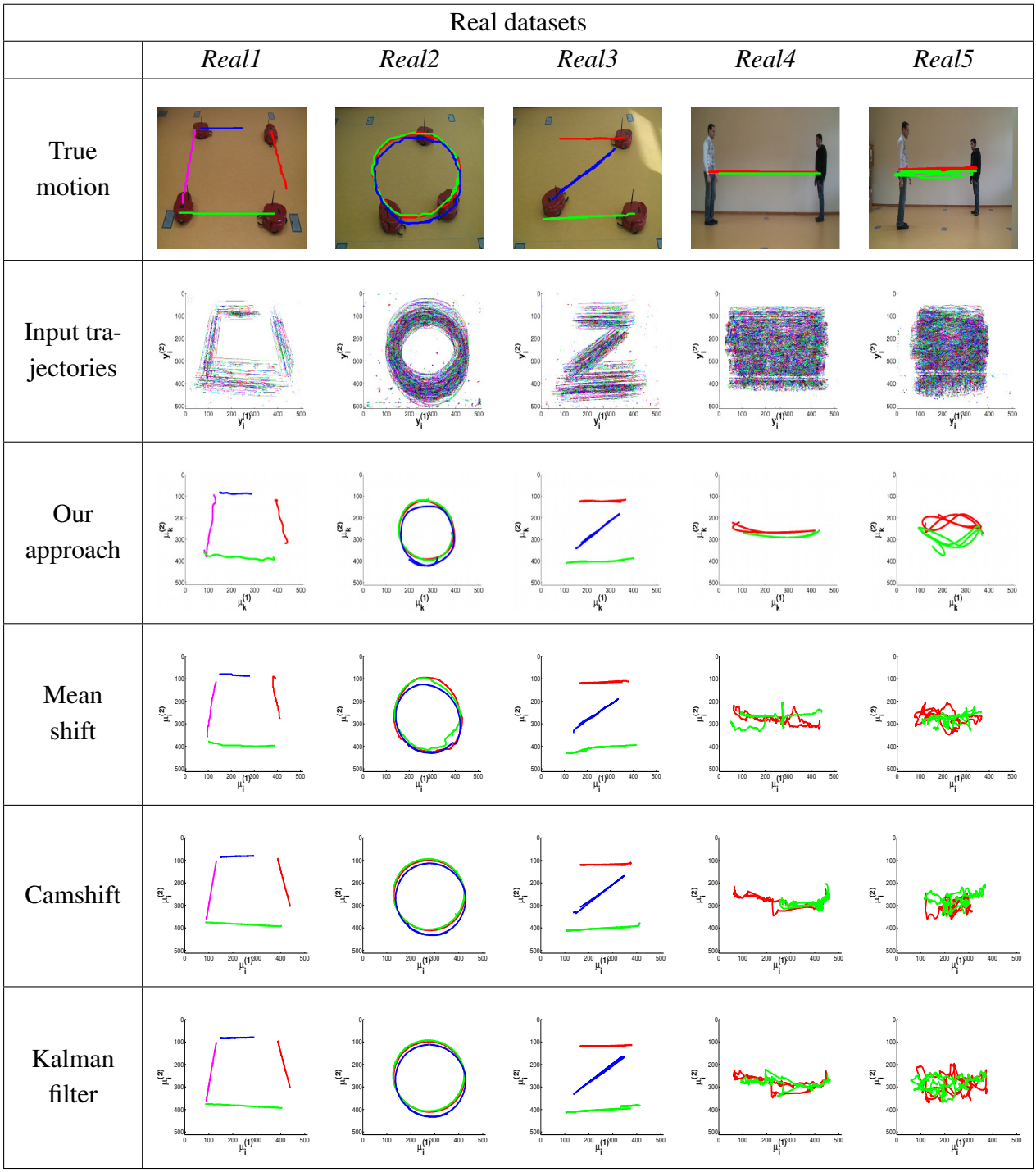


Figure 7.8: Comparative results with five real datasets. For each problem we give the true objects motion (chosen manually), the created input trajectories and the estimated motion by all approaches.

follows the person before and after the first occlusion, but it fails to track it when the second one takes place. Camshift fails to track the person after the first occlusion, but it follows it after the second occlusion when the person turns back. The Kalman filter successfully follows the person. This is better depicted in Fig. 7.9 where the predicted trajectory, corresponding to the person in black, is shown in green. The part of the trajectory where the person is lost is high-

lighted by an ellipse. Mean shift (Fig. 7.9(b)) follows the correct person from right to left and from left until the middle of the image when the second occlusion takes place. Then it follows the other person which moves from the middle of the image to the left. Camshift (Fig. 7.9(c)) follows the person until the first occlusion at the middle of the image. Then, it follows the other person which moves from the middle to the right and from the right to the middle. After the second occlusion is occurred, it finds the correct person again which moves from the middle to the right. On the other hand, the proposed method successfully tracks the object in all frames (Fig. 7.9(a)).

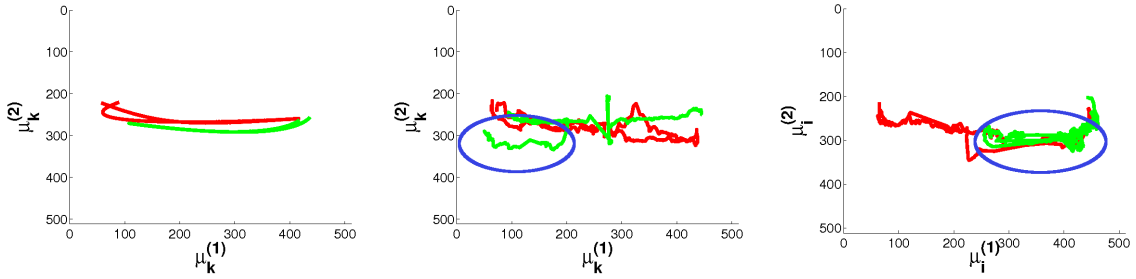


Figure 7.9: Estimated trajectories for the dataset *Real4*. (a) Our method, (b) mean shift, (c) camshift. The green (printed in light gray in black and white) trajectory in (b) and (c) corresponds to the person in black moving from the right side of the image to the left and backwards. The ellipse highlights the part of the trajectory where the person is lost, because mean shift or camshift fails to track the object due to occlusion. The figure is better visualized in color.

7.4.3 Experiments using the Hopkins 155 dataset

We have also used the Hopkins 155 dataset [115] to evaluate our approach, where we have selected the traffic subset that consists of 31 scenarios with two motions and 7 scenarios with three motions. For each video, the extracted trajectories and the ground truth are provided. Here we must highlight that these trajectories have been manually corrected or filled by the researchers that constructed the dataset [115], so each trajectory has a value for every frame. Some representative frames of the traffic subset are presented in Fig. 7.10.

Figure 7.10: Representative frames of the Hopkins 155 dataset. The feature points are marked using different colors in order to denote the cluster they belong to.

In Tables 7.2 and 7.3, the performance of our algorithm is presented in terms of the classification error. For comparison purposes, we also summarize the results obtained by 15 other

methods as they are presented in the web page of the dataset (http://www.vision.jhu.edu/ motion.php#results). More specifically, the average value of the classification error as calculated by our method over the corresponding dataset is shown in Table 7.2. For the rest of the compared methods, we present the minimum, the maximum, the mean and the median values for the average classification error over all the 15 state of art methods. This means that we have included the average classification error of each of the compared methods in the computation of each statistic (min, max, mean, median). As it may be observed, our method provides satisfactory results compared to all other methods. This is also confirmed by the ranking information, shown in the last column of Table 7.2. In the case of two motions, our method is ranked first among the 15 compared algorithms. Let us notice that the method sparse subspace clustering (SSC) [37] provides a similar average error. This is indicated in the last column of the Table 7.2, showing how many methods provide the same error in average. In the case of three motions, the performance of our method is ranked in the second place behind SSC, whose average error is 0.58%.

Similar in spirit statistics are shown in Table 7.3 concerning the median classification error. In that case, our method accurately classifies more than half of the time series which is also the case for 7 out of 15 methods for the problems involving two motions. For the three motions problems, where the complexity increases, our algorithm is also ranked at the first place along with other three methods, namely multi stage learning (MSL) [111], local linear manifold clustering with projection to a space of dimension 5 (LLMC 5) [45] and SSC [37].

Table 7.2: Statistics on the average of classification error for the traffic subset of the Hopkins 155 dataset.

	Our approach	Other approaches				Our Rank	In tie
		Min	Max	Mean	Median		
Two motions	0.02%	0.02%	5.74%	2.63%	2.23%	1/15	1
Three Motions	0.98%	0.58%	27.02%	9.53%	8.00%	2/15	-

Table 7.3: Statistics on the median of classification error for the traffic subset of the Hopkins 155 dataset.

	Our approach	Other approaches				Our Rank	In tie
		Min	Max	Mean	Median		
Two motions	0.00%	0.00%	1.55%	0.51%	0.21%	1/15	7
Three Motions	0.00%	0.00%	34.01%	7.22%	2.06%	1/15	3

7.4.4 Experiments using other key point descriptors

In section 7.2 we described how to create trajectories from an image sequence and as an example we used *Harris corners* [48] in order to detect salient image features and optical flow [78] to associate them between images. Apart from *Harris corners*, numerous interest-point detectors have been proposed in the computer vision literature, such as the scale invariant feature transform difference of Gaussian key points (SIFT DoG or SIFT for simplicity) [77], the maximally stable extremal regions (MSER) [84], the Hessian matrix-based affine features [86] and the speeded-up robust features (SURF) [11]. They mainly differ to the level of the trade-off between repeatability and complexity [25, 86]. The SIFT features for example are highly repeatable but require a large computational cost. This is why SURF key point detectors have gained increasing interest, as they are faster (they are based on box filters and integral images [119]).

In order to study the consistency of the proposed method we have evaluated it in terms of the technique used for generating trajectories. Table 7.4 summarize the comparative results on the seven simulated datasets using *Harris corners*, SIFT and SURF features. For every case we give also the number of the generated trajectories. It may be observed that when the trajectories are computed using *Harris corners* a smaller MSE is obtained while trajectories computed by SIFT and SURF detectors have approximately similar errors but in any case higher than *Harris corners*. In terms of accuracy, all of the methods exhibit, in average, comparable rates which indicates a larger number of trajectories, provided for example by SURF, does not necessary ensure a better result.

Table 7.4: The performance of the different key point extraction methods in terms of classification accuracy (ACC) and mean squared error (MSE).

Problem	*Harris corners*			SIFT			SURF		
	#Traject.	MSE	ACC	#Traject.	MSE	ACC	#Traject.	MSE	ACC
Sim1	3820	**69**	**100%**	973	130	**100%**	6352	107	99%
Sim2	1516	**10**	**99%**	1146	138	98%	3453	139	**99%**
Sim3	2348	**10**	96%	2296	172	98%	7708	104	**99%**
Sim4	2346	15	97%	2319	122	98%	7758	32	100%
Sim5	1954	**20**	**100%**	811	110	99%	4848	130	**100%**
Sim6	1351	**29**	**100%**	646	142	99%	3693	191	99%
Sim7	1485	**41**	99%	689	168	**100%**	4244	155	99%

7.5 Conclusions

In this chapter, we have presented a compact methodology for objects tracking based on model-based clustering trajectories of Harris corners extracted from an image sequence. Clustering is

achieved through an efficient sparse regression mixture model that embodies efficient characteristics in order to handle trajectories of variable length, and to be translated in measurement space. Experiments have shown the abilities of our approach to automatically detect the motion of objects without any human interaction and also demonstrated its robustness to occlusion and feature misdetection.

The main advantage of our method with respect to Kalman filter is that the former handles both tracking and motion segmentation while the latter only tracks the target. Also, Kalman filter should be provided with the motion model while the method proposed herein needs as input only the number of the objects to be tracked. Moreover, our method may be applied without the knowledge of the full trajectories by using only time series data up to the current time instant if this is imposed by the application. Finally, linear regression model can be applied in order to predict the next state [14].

CHAPTER 8

CONCLUSIONS AND FUTURE WORK

8.1 Conclusions

In this work, visual tracking algorithms using clustering methods were proposed in order to provide solutions to drawbacks and limitations of the existing methods. A first issue arrises when scene illumination changes the color of the object to be tracked. In this case, the values of the histogram representing the object's color shift, making tracking approaches unstable. A possible solution is to smooth the target's candidate histogram using a a GMM that has the form of the target's model histogram. However, this approach does not eliminate the problem of the curse of dimensionality of the target's representation. To this end, two methods using mixtures of continuous distributions were proposed. The first approach uses weighted Gaussian mixture models in order to represent the distribution of the object's features. During the tracking procedure, the GMM of the target candidate and the target model are compared using EMD and a gradient based approach is employed in order to minimize their distance. EMD is robust to variations of the compared distributions, but it needs the estimation of two mixture models. The second approach uses only one weighted GMM, whose parameters are estimated during the initialization step. In the tracking step, the position of the target that maximizes this specific weighted GMM is estimated using a closed form update. A further improvement in terms of execution speed can be achieved by using only the hue component instead of the complete RGB feature vector. However, the hue is periodic and the Gaussian distribution can not model it accurately. Thus, a weighted von Mises mixture model is employed, which results to an approach both robust to illumination changes and computationally efficient.

Color variations are not only a result from scene illumination change, but also from actual color change of the object (e.g. due to 3D rotation of the object). We developed a solution to this

limitation in order to update the appearance model when a previously unseen face of the target appears. The key idea is to track the new model (including its new color) backward in time and compare this trajectory with the trajectory that was produced during the standard tracking procedure. This framework is not tightly combined with any specific tracking algorithm so it can be used by any similar method.

Moreover, occlusions may decrease the performance of the tracking algorithms. In order to handle this problem, we proposed a method based on Kalman filtering which can be combined with other tracking approaches. The Kalman filter predicts the position of the target in the next frame, which can be used if parts of the target are occluded. Finally, a novel framework was proposed for tracking multiple objects with partial occlusions by clustering the trajectories of various key points. The trajectories can be of varying length and may have missing values, due to the fact that the key points may disappear and reappear during the tracking procedure. Tracking is accomplished by estimating a sparse model for the mean trajectory of each object's key points. The correspondence between the key points and the objects as well as the initial position of the objects are not known a priori.

8.2 Future Work

In the following, we present some promising directions for future research that elaborate on a number of open issues related to the problems that we have tackled in this thesis so far. The following topics are of interest for more detailed investigation.

The methodology of Chapter 4 can applied to other approaches that use histograms in order to represent the object. Using this approach, the standard algorithm does not change, for example by using cross bin metrics. The only difference would be an intermediate smoothing of its after the histogram estimation and before employing it at the next steps of the algorithm.

The methodology that is proposed in Chapter 5 in order to update the appearance model of the target can be incorporated in other tracking approaches, as not require any special property from the tracking algorithms, except for the history of the target's positions at the corresponding frames.

The algorithm of Chapter 6 can be used with other circular features, for example the angle of the image gradient, which is also robust to illumination changes.

The approach of the weighted likelihood which was presented in Chapters 4 and 5 can be used in other applications, when some samples of the data are considered more important compared to others. In the application of visual tracking using spatial kernels, the pixels near the center of the object are considered more important, due to the fact that they are more probable to belong to the object.

We also examined the application of Gaussian mixture models and von Mises mixture models. However, the appearance model of the color could also be modeled with mixtures of other distributions. For example, the Student's t-distribution is robust to outliers. Moreover, an interesting approach would be to model different components with different distributions. For

instance, in the HSV color space, the hue component is periodic, while the saturation and value components are linear. Thus, the von Mises distribution can be used to model circular spaces and the Gaussian distribution for linear spaces.

BIBLIOGRAPHY

[1] V. Ablavsky and S. Sclaroff. Layered graphical models for tracking partially occluded objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(8):1758–1775, 2011.

[2] M. Abramowitz and I. Stegun. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables*. Dover Publications, 1965.

[3] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 1, pages 798–805, 2006.

[4] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering clusters in motion time-series data. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 375–381, 2003.

[5] A. A. Argyros and M. I. A. Lourakis. Real-time tracking of multiple skin-colored object with a possibly moving camera. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 368–379, 2004.

[6] S. Arulampalam, S. Maskell, and N. Gordon. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[7] S. Avidar. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(8):1064–1072, 2004.

[8] R. V. Babu, P. Pérez, and P. Bouthemy. Robust tracking with motion estimation and local kernel-based color modeling. *Image and Vision Computing*, 25(8):1205–1216, 2007.

[9] S.-H. Bae and K.-J. Yoon. Robust online multi-object tracking based on tracklet confidence and online discriminative appearance learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14)*, pages 1218–1225, 2014.

[10] C. Bailer, A. Pagani, and D. Stricker. A superior tracking approach: Building a strong tracker through fusion. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 8695, pages 170–185, 2014.

[11] H. Bay, T. Tuytelaars, and L. V. Gool. SURF: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision (ECCV'06)*, pages 404–417, 2006.

[12] D. Beymer and K. Konolige. Real-time tracking of multiple people using continuous detection. In *Proceedings of International Conference on Computer Vision (ICCV'99)*, 1999.

[13] C. Bibby and I. Reid. Real-time tracking of multiple occluding objects using level sets. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10)*, pages 1307 –1314, 2010.

[14] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[15] M. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[16] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26:63–84, 1998.

[17] K. Blekas, C. Nikou, N. Galatsanos, and N. Tsekos. A regression mixture model with spatial constraints for clustering spatiotemporal data. *Intern. Journal on Artificial Intelligence Tools*, 17(5):1023–1041, 2008.

[18] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[19] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reily, 2008.

[20] G. R. Bradski. Computer vision face tracking for use in a perceptual user interface. *Intel Technology Journal*, Q2, 1998.

[21] R. P. Browne, P. D. McNicholas, and M. Sparling. Model-based learning using a mixture of mixtures of gaussian and uniform distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):814–817, 2012.

[22] A. Bugeau and P. Perez. Track and cut: Simultaneous tracking and segmentation of multiple objects with graph cuts. *EURASIP Journal on Image and Video Processing*, 2008(ID:317278), 2008.

[23] D. Buzan, S. Sclaroff, and G. Kollios. Extraction and clustering of motion trajectories in video. In *International Conference on Pattern Recognition*, pages 521–524, 2004.

[24] S. Calderara, A. Prati, and R. Cucchiara. Mixtures of von Mises distributions for people trajectory shape analysis. *IEEE Transactions on Circuits and Systems for Video Technology*, 21(4):457–471, 2011.

[25] V. Chandrasekhar, D. Chen, A. Lin, G. Takacs, S. Tsai, N.-M. Cheung, Y. Reznik, R. Grzeszczuk, and B. Girod. Comparison of local feature descriptors for mobile visual search. *IEEE Signal Processing Magazine*, 28(4):61–76, 2011.

[26] Z. Chen and T. Ellis. A self-adaptive gaussian mixture model. *Computer Vision and Image Understanding*, 122(0):35–46, 2014.

[27] H. S. Choi, I. S. Kim, and J. Y. Choi. Combining histogram-wise and pixel-wise matchings for kernel tracking through constrained optimization. *Computer Vision and Image Understanding*, 118(0):61–70, 2014.

[28] E. K. P. Chong and S. H. Zak. *An Introduction to Optimization*. Wiley-Interscience Series in Discrete Mathematics and Optimization, 3rd edition, 2008.

[29] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[30] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(5):564–577, 2003.

[31] D. Cremers. Dynamical statistical shape priors for level set based tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(8):1262–1273, 2006.

[32] E. Cuevas, D. Zaldivar, and R. Rojas. Kalman filter for vision tracking. Technical Report B 05-12, Freier Universitat Berlin, Institut fur Informatik, 2005.

[33] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Roy. Statist. Soc. B*, 39:1–38, 1977.

[34] S. Duffner and C. Garcia. Pixeltrack: A fast adaptive algorithm for tracking non-rigid objects. In *Proceedings of International Conference on Computer Vision (ICCV'13)*, pages 2480–2487, 2013.

[35] A. M. Elgammal, R. Duraiswami, and L. S. Davis. Probabilistic tracking in joint feature-spatial spaces. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03)*, pages 781–788, 2003.

[36] T. Elguebaly and N. Bouguila. Finite asymmetric generalized gaussian mixture models learning for infrared object detection. *Computer Vision and Image Understanding*, 117(12):1659–1671, 2013.

[37] E. Elhamifar and R. Vidal. Sparse subspace clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09)*, pages 2790–2797, 2009.

[38] L. Ellis, N. Dowson, J. Matas, and R. Bowden. Linear regression and adaptive appearance models for fast simultaneous modelling and tracking. *International Journal of Computer Vision*, 95(2):154–179, 2011.

[39] Z. Fan, M. Yang, and Y. Wu. Multiple collaborative kernel tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1268–1273, 2007.

[40] M. Fergie and A. Galata. Mixtures of gaussian process models for human pose estimation. *Image and Vision Computing*, 31(12):949–957, 2013.

[41] M. A. T. Figueiredo and A. K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:381–396, 2000.

[42] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[43] S. Gaffney. *Probabilistic curve-aligned clustering and prediction with regression mixture models*. PhD thesis, Department of Computer Science, University of California, Irvine, 2004.

[44] S. Gammeter, A. Ess, T. Jaeggli, K. Schindler, B. Leibe, and L. van Gool. Articulated multibody tracking under egomotion. In *European Conference on Computer Vision (ECCV'08)*, LNCS, pages 816–830. Springer, 2008.

[45] A. Goh and R. Vidal. Segmenting motions of different types by unsupervised manifold clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, 2007.

[46] G. D. Hager, M. Dewan, and C. V. Stewart. Multiple kernel tracking with SSD. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04)*, volume 1, pages 790–797, 2004.

[47] B. Han, S.-W. Joo, and L. S. Davis. Probabilistic fusion tracking using mixture kernel-based Bayesian filtering. In *Proceedings of International Conference on Computer Vision (ICCV'07)*, pages 1–8, 2007.

[48] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[49] S. He, Q. Yang, R. W. Lau, J. Wang, and M.-H. Yang. Visual tracking via locality sensitive histograms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13)*, pages 2427–2434, 2013.

[50] Z. Hong, Z. Chen, C. Chen, X. Mei, D. Prokhorov, and D. Tao. MUlti-store tracker (MUSTer): A cognitive psychology inspired approach to object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 749–758, 2015.

[51] E. Horbert, K. Rematas, and B. Leibe. Level-set person segmentation and tracking with multi-region appearance models and top-down shape information. In *IEEE International Conference on Computer Vision (ICCV'11)*, pages 1–8, 2011.

[52] Y. Hua, K. Alahari, and C. Schmid. Occlusion and motion reasoning for long-term tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 8694, pages 172–187, 2014.

[53] J. S. S. III and D. Ramanan. Self-paced learning for long-term tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13)*, pages 2379–2387, 2013.

[54] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29:5–28, 1998.

[55] M. Isard and A. Blake. Icondensation - unifying low-level and high-level tracking in a stochastic framework. In *Proceedings of the 5th European Conference on Computer Vision (ECCV)*, pages 893–908, 1998.

[56] O. Javed, M. Shah, and D. Comaniciu. A probabilistic framework for object recognition in video. In *International Conference on Image Processing, 2004 (ICIP '04)*, volume 4, pages 2713–2716, 2004.

[57] C. Julià, A. Sappa, F. Lumbreras, J. Serrat, and A. López. Motion segmentation from feature trajectories with missing data. In *Proceedings of the 3rd Iberian conference on Pattern Recognition and Image Analysis, Part I*, IbPRIA '07, pages 483–490, 2007.

[58] T. Kailath. The divergence and Bhattacharyya distance measures in signal selection. *IEEE Transactions on Communications*, 15(1):52–60, 1967.

[59] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *Proceedings of 20th International Conference on Pattern Recognition (ICPR'10)*, pages 2756–2759, 2010.

[60] A. T. Kamal, J. A. Farrell, and A. K. Roy-Chowdhury. Information consensus for distributed multi-target tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13)*, pages 2403–2410, 2013.

[61] S. Kay. *Fundamentals of statistical signal processing: Estimation Theory*. Prentice Hall, 1993.

[62] J. Kim, Z. Lin, and I. S. Kweon. Rao-Blackwellized particle filtering with Gaussian mixture models for robust visual tracking. *Computer Vision and Image Understanding*, 125(1):128–137, 2014.

[63] J.-H. Kim and L. Agapito. Motion segmentation using the Hadamard product and spectral clustering. In *Proceedings of the 2009 international conference on Motion and video computing*, WMVC'09, pages 126–133. IEEE Computer Society, 2009.

[64] L. Kratz and K. Nishino. Going with the flow: Pedestrian efficiency in crowded scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 558–572, 2012.

[65] M. Kristan, R. Pflugfelder, A. Leonardis, J. Matas, L. Cehovin, G. Nebehay, T. Vojir, G. Fernandez, A. Lukezic, A. Dimitriev, A. Petrosino, A. Saffari, B. Li, B. Han, C. Heng, C. Garcia, D. Pangersic, G. Hager, F. S. Khan, F. Oven, H. Possegger, H. Bischof, H. Nam, J. Zhu, J. Li, J. Y. Choi, J.-W. Choi, J. F. Henriques, J. van de Weijer, J. Batista, K. Lebeda, K. Ofjall, K. M. Yi, L. Qin, L. Wen, M. E. Maresca, M. Danelljan, M. Felsberg, M.-M. Cheng, P. Torr, Q. Huang, R. Bowden, S. Hare, S. Y. Lim, S. Hong, S. Liao, S. Hadfield, S. Z. Li, S. Duffner, S. Golodetz, T. Mauthner, V. Vineet, W. Lin, Y. Li, Y. Qi, Z. Lei, and Z. Niu. The visual object tracking vot2014 challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV) Visual Object Tracking Challenge Workshop*, pages 98–111, Zurich, Switzerland, September 2014.

[66] J. Kwon, J. Roh, K. Lee, and L. V. Gool. Robust visual tracking with double bounding box model. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 8689, pages 377–392, 2014.

[67] F. Lauer and C. Schnorr. Spectral clustering of linear subspaces for motion segmentation. In *Proc. of the 12th IEEE Int. Conf. on Computer Vision*, ICCV'09, 2009.

[68] D.-Y. Lee, J.-Y. Sim, and C.-S. Kim. Multihypothesis trajectory analysis for robust visual tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 5088–5096, 2015.

[69] I. Leichter. Mean shift trackers with cross-bin metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):695–706, 2012.

[70] I. Leichter, M. Lindenbaum, and E. Rivlin. Mean shift tracking with multiple reference color histograms. *Computer Vision and Image Understanding*, 114(3):400–408, 2010.

[71] S. Li, O. Wu, C. Zhu, and H. Chang. Visual object tracking using spatial context information and global tracking skills. *Computer Vision and Image Understanding*, 125:1–15, 2014.

[72] S.-X. Li, H.-X. Chang, and C.-F. Zhu. Adaptive pyramid mean shift for global real-time visual tracking. *Image and Vision Computing*, 28(3):424–437, 2010.

[73] Y. Li, J. Zhu, and S. C. Hoi. Reliable patch trackers: Robust visual tracking by exploiting reliable patches. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 353–361, 2015.

[74] B. Liu, J. Huang, C. Kulikowski, and L. Yang. Robust visual tracking using local sparse appearance model and k-selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2968–2981, 2013.

[75] S. Liwicki, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. Efficient online subspace learning with an indefinite kernel for visual tracking and recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 23:1624–1636, 2012.

[76] S. Liwicki, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. Euler principal component analysis. *International Journal of Computer Vision*, 101(3):498–518, 2013.

[77] D. G. Lowe. Distinctive image features from scale-invariant keypoint. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[78] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI '81)*, pages 674–679, 1981.

[79] X. Luo, Y. Wan, X. He, J. Yang, and K. Mori. Diversity-enhanced condensation algorithm and its application for robust and accurate endoscope three-dimensional motion tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14)*, pages 1250–1257, 2014.

[80] D. R. Magee. Tracking multiple vehicles using foreground, background and motion models. *Image and Vision Computing*, 22(2):143–155, 2004.

[81] A. Mansouri. Region tracking via level set pdes without motion computation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24:947–961, 2002.

[82] I. Marras, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. Online learning and fusion of orientation appearance models for robust rigid object tracking. *Image and Vision Computing*, 32(10):707–727, 2014.

[83] O. Masoud and N. P. Papanikolopoulos. A novel method for tracking and counting pedestrians in real-time using a single camera. *IEEE Transactions on Vehicular Technology*, 50(5):1267–1278, 2001.

[84] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. In *In British Machine Vision Conference (BMVC'02)*, volume 1, pages 384–393, 2002.

[85] X. Mei and H. Ling. Robust visual tracking and vehicle classification via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(11):2259–2272, 2011.

[86] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. V. Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1/2):43–72, 2005.

[87] A. Milan, K. Schindler, and S. Roth. Detection- and trajectory-level exclusion in multiple object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13)*, pages 3682–3689, 2013.

[88] F. Moreno-Noguer, A. Sanfeliu, and D. Samaras. Dependent multiple cue integration for robust tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(4):670–685, 2008.

[89] D. Mukherjee, Q. M. J. Wu, and T. M. Nguyen. Multiresolution based gaussian mixture model for background suppressio. *IEEE Transactions on Image Processing*, 22(12):5022–5035, 2013.

[90] D. P. Mukherjee, S. Member, N. Ray, S. T. Acton, and S. Member. Level set analysis for leukocyte detection and tracking. *IEEE Transactions on Image Processing*, 13:562–572, 2004.

[91] G. Nebehay and R. Pflugfelder. Clustering of static-adaptive correspondences for deformable object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 2791–2784, 2015.

[92] T. M. Nguyen and Q. J. Wu. Dirichlet gaussian mixture model: Application to image segmentation. *Image and Vision Computing*, 29(12):818–828, 2011.

[93] S. Oron, A. Bar-Hillel, and S. Avidan. Extended lucas kanade tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 8693, pages 142–156, 2014.

[94] N. Papadakis and A. Bugeau. Tracking with occlusions via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):144–157, 2011.

[95] N. Paragios and R. Deriche. Geodesic active contours and level sets for the detection and tracking of moving objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:266–280, 2000.

[96] K. Pauwels, L. Rubio, J. Diaz, and E. Ros. Real-time model-based rigid object pose estimation and tracking combining dense and sparse visual cues. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13)*, pages 2347–2354, 2013.

[97] F. Poiesi, R. Mazzon, and A. Cavallaro. Multi-target tracking on confidence maps: An application to people tracking. *Computer Vision and Image Understanding*, 117(10):1077–1272, 2013.

[98] G. Pons-Moll, A. Baak, J. Gall, L. Leal-Taixe, M. Muller, H. Seidel, and B. Rosenhahn. Outdoor human motion capture using inverse kinematics and von Mises-Fisher sampling. In *Proceedings of International Conference on Computer Vision (ICCV'11)*, pages 1243–1250, 2011.

[99] H. Possegger, T. Mauthner, and H. Bischof. In defense of color-based model-free tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 2113–2120, 2015.

[100] S. J. Pundlik and S. T. Birchfield. Real-time motion segmentation of sparse feature points at any speed. *IEEE Transactions on Systems, Man, and Cybernetics*, 38:731–742, 2008.

[101] Y. Rathi, N. Vaswani, A. Tannenbaum, and A. Yezzi. Particle filtering for geometric active contours with application to tracking moving and deforming objects. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2005 (CVPR'05)*, volume 2, pages 2–9, 2005.

[102] Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover's distance as a metric for image retrieval. *International Journal of Computer Vision*, 40(2):99–121, 2000.

[103] K. Sato and J. K. Aggarwal. Temporal spatio-velocity transform and its application to tracking and interaction. *Computer Vision and Image Understanding*, 96(2):100–128, 2004.

[104] F. Seitner and A. Hanbury. Fast pedestrian tracking based on spatial features and color. In *Proceedings of the Computer Vision Winter Workshop*, pages 105–110, 2006.

[105] J. Shi and C. Tomasi. Good features to track. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94)*, pages 593–600, 1994.

[106] D. Simon. *Optimal state estimation: Kalman, H_∞ and non linear approaches*. Wiley–Interscience, 2006.

[107] A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1442–1468, 2014.

[108] B. Song, T.-Y. Jeng, E. Staudt, and A. K. Roy-Chowdhury. A stochastic graph evolution framework for robust multi-target tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 605–619, 2010.

[109] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing: Analysis and Machine Vision*. Thomson-Engineering, 1998.

[110] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):747–757, 2000.

[111] Y. Sugaya and K. Kanatani. Multi-stage optimization for multi-body motion segmentation. *IEICE Transactions on Information and Systemsg*, E87-D(7):1935–1942, 2004.

[112] S. Tang, M. Andriluka, A. Milan, K. Schindler, S. Roth, and B. Schiele. Learning people detectors for tracking in crowded scenes. In *Proceedings of International Conference on Computer Vision (ICCV'13)*, pages 1049–1056, 2013.

[113] H. Tao, H. S. Sawhney, and R. Kumar. Object tracking with Bayesian estimation of dynamic layer representations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):75–89, 2002.

[114] M. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.

[115] R. Tron and R. Vidal. A benchmark for the comparison of 3d motion segmentation algorithms. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'07)*, volume 1, 2007.

[116] J. Tu, H. Tao, and T. Huang. Online updating appearance generative mixture model for mean-shift tracking. *Machine Vision and Applications*, 20(3):163–173, 2009.

[117] G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. Sparse representations of image gradient orientations for visual recognition and tracking. In *Proceedings of Workshop on CVPR for Human Behaviour Analysis (CVPR-W'11)*, pages 26–33, 2011.

[118] C. J. Veenman, M. J. T. Reinders, and E.Backer. Resolving motion correspondence for densely moving points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(1):54–72, 2001.

[119] P. Viola and M. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57:137–154, 2004.

[120] T. Vojir, J. Noskova, and J. Matas. Robust scale-adaptive mean-shift for tracking. In *Proc. of the 18th Scandinavian Conf. on Image Analysis (SCIA)*, pages 652–663, 2013.

[121] D. Wang, H. Lu, and M.-H. Yang. Online object tracking with sparse prototypes. *IEEE Transactions on Image Processing*, 22(1):314–325, 2013.

[122] H. Wang, D. Suter, and K. Schindler. Effective appearance model and similarity measure for particle filtering and visual tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 606–618, 2006.

[123] N. Wang, J. Wang, and D.-Y. Yeung. Online robust non-negative dictionary learning for visual tracking. In *Proceedings of International Conference on Computer Vision (ICCV'13)*, pages 657–664, 2013.

[124] X. Wang, E. Turetken, F. Fleuret, and P. Fua. Tracking interacting objects optimally using integer programming. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 8689, pages 17–32, 2014.

[125] Z. Wang, M. B. Salah, H. Salah, and N. Ra. Shape based appearance model for kernel tracking. *Image and Vision Computing*, 30(4–5):332–344, 2012.

[126] Z. Wang, X. Yang, Y. Xu, and S. Yu. Camshift guided particle filter for visual tracking. *Pattern Recognition Letters*, 30(4):407–413, 2009.

[127] K. Y. Wong and M. E. Spetsakis. Motion segmentation by EM clustering of good features. In *Computer Vision and Pattern Recognition Workshop (CVPRW'04)*, volume 11, pages 1–8, 2004.

[128] B. Wu, S. Lyu, B.-G. Hu, and Q. Ji. Simultaneous clustering and tracklet linking for multi-face tracking in videos. In *Proceedings of International Conference on Computer Vision (ICCV'13)*, pages 2856–2863, 2013.

[129] Y. Wu and T. S. Huang. Robust visual tracking by integrating multiple cues based on co-inference learning. *International Journal of Computer Vision*, 58(1):55–71, 2004.

[130] Y. Wu, J. Lim, and M.-H. Yang. Object tracking benchmark. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP(99):1–1, 2015.

[131] J. Xiao, R. Stolkin, and A. Leonardis. Single target tracking using adaptive clustered decision trees and dynamic multi-level appearance models. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 4978–4987, 2015.

[132] J. Xing, J. Gao, B. Li, W. Hu, and S. Yan. Robust object tracking with online multi-lifespan dictionary learning. In *Proceedings of International Conference on Computer Vision (ICCV'13)*, pages 665–672, 2013.

[133] J. Yan and M. Pollefeys. A general framework for motion segmentation: Independent, articulated, rigid, non-rigid, degenerate and non-degenerate. In *Proceedings of the European Conference on Computer Vision*, ECCV'06, 2006.

[134] X. Yan, I. A. Kakadiaris, and S. K. Shah. What do I see? modeling human visual perception for multi-person tracking. In *Proceedings of the European Conference on Computer Vision (ECCV)*, volume 8690, pages 314–329, 2014.

[135] C. Yang, R. Duraiswami, and L. Davis. Efficient mean-shift tracking via a new similarity measure. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 176–183, 2005.

[136] F. Yang, H. Lu, and M.-H. Yang. Robust superpixel tracking. *IEEE Transactions on Image Processing*, 23(4):1639–1651, 2014.

[137] M. Yang, Y. Wu, and G. Hua. Context-aware visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1195–1209, 2009.

[138] R. Yao, Q. Shi, C. Shen, Y. Zhang, and A. van den Hengel. Part-based visual tracking with online latent structural learning. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'13)*, pages 2363–2370, 2013.

[139] K. M. Yi, H. Jeong, B. Heo, H. J. Chang, and J. Y. Choi. Initialization-insensitive visual tracking through voting with salient local features. In *Proceedings of International Conference on Computer Vision (ICCV'13)*, pages 2912–2919, 2013.

[140] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Computing Surveys*, 38(4):1–45, 2006.

[141] A. Yilmaz, X. Li, and M. Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(11):1531–1536, 2004.

[142] L. Zappella, X. Llado, E. Provenzi, and J. Salvi. Enhanced local subspace affinity for feature-based motion segmentation. *Pattern Recognition*, 44:454–470, 2011.

[143] M. Zeppelzauer, M. Zaharieva, D. Mitrovic, and C. Breiteneder. A novel trajectory clustering approach for motion segmentation. In *Advances in Multimedia Modeling*, volume 5916, pages 433–443. Springer, 2010.

[144] K. Zhang, L. Zhang, and M.-H. Yang. Real-time object tracking via online discriminative feature selection. *IEEE Transactions on Image Processing*, 22(12):4664–4677, 2013.

[145] K. Zhang, L. Zhang, and M.-H. Yang. Fast compressive tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(10):2002–2015, 2014.

[146] L. Zhang, H. Dibeklioglu, and L. van der Maaten. Speeding up tracking by ignoring features. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14)*, pages 1266–1273, 2014.

[147] L. Zhang and L. van der Maaten. Preserving structure in model-free tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(4):756–769, 2014.

[148] T. Zhang, K. Jia, C. Xu, Y. Ma, and N. Ahuja. Partial occlusion handling for visual tracking via robust part matching. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14)*, pages 1258–1265, 2014.

[149] T. Zhang, S. Liu, C. Xu, S. Yan, B. Ghanem, N. Ahuja, and M.-H. Yang. Structural sparse tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'15)*, pages 150–158, 2015.

[150] Z. Zhang and K. H. Wong. Pyramid-based visual tracking using sparsity represented mean transform. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'14)*, pages 1226–1233, 2014.

[151] Q. Zhao, S. Brennan, and H. Tao. Differential EMD tracking. In *Proceedings of International Conference on Computer Vision (ICCV'07)*, pages 1–8, 2007.

[152] Q. Zhao and H. Tao. Differential earth mover's distance with its application to visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(5):274–287, 2010.

[153] H. Zhou, Y. Yuan, and C. Shi. Object tracking using sift features and mean shift. *Computer Vision and Image Understanding*, 113(3):345–352, 2009.

[154] K. Zimmermann, J. Matas, and T. Svoboda. Tracking by an optimal sequence of linear predictors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(4):677–692, 2009.

[155] Z. Zivkovic and B. Krose. An EM-like algorithm for color-histogram-based object tracking. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04)*, volume 1, pages 798–803, 2004.

APPENDIX A

THE KALMAN FILTER

In this section, the Kalman filter [42, 109, 32, 61, 140] is presented.

A.1 Motion Model

We assume that the data generation procedure is a linear system, which has an inner (unknown) state and observations made from it (known). More specifically, there is a discrete time system with its inner state at time n given by $\mathbf{x}_n$ ($\mathbf{x}_n$ may be a scalar or a vector). The state of the system at the next time $n+1$ is given by the linear equation:

$$\mathbf{x}_{n+1} = \mathbf{F}_n \mathbf{x}_n + \mathbf{w}_n \tag{A.1}$$

In equation (A.1), $\mathbf{x}_n$ and $\mathbf{x}_{n+1}$ are the system state at time n and $n+1$ respectively, $\mathbf{F}_n$ is the transition matrix from state $\mathbf{x}_n$ to the state $\mathbf{x}_{n+1}$ and $\mathbf{w}_n$ the state evolution noise at time n. We assume that $\mathbf{w}_n$ is white Gaussian noise with zero mean value and covariance matrix given by:

$$\mathbf{E}[\mathbf{w}_i \mathbf{w}_j^T] = \begin{cases} \mathbf{Q}_i & , \quad i = j \\ 0 & , \quad i \neq j \end{cases} \tag{A.2}$$

At every time n, observations are made from the system. These observations are produced from the state of the system, but they do not represent exactly the system state (generally the observation is a vector with different dimension than the state matrix). In general, observation at time n is given by the equation:

$$\mathbf{z}_n = \mathbf{H}_n \mathbf{x}_n + \mathbf{v}_n \tag{A.3}$$

where, $\mathbf{z}_n$ is the observation, $\mathbf{x}_n$ is the state of the system, $\mathbf{H}_n$ is the observation matrix and $\mathbf{v}_n$ is the noise that affects the observation at time n. Noise $\mathbf{v}_n$ is supposed to be white Gaussian with zero mean and covariance matrix given by:

$$\mathbf{E}[\mathbf{v}_i \mathbf{v}_j^T] = \begin{cases} \mathbf{R}_i & , \quad i = j \\ 0 & , \quad i \neq j \end{cases} \tag{A.4}$$

Notice that in equation (A.1) the state $\mathbf{x}_{n+1}$ depends only on the previous state $\mathbf{x}_n$. This means that in order to generate the next state, the previous states are not needed (and do not need to be stored). In order to start the system, an initial state is needed (that is the state $\mathbf{x}_{-1}$). This state may be be known or it can be taken as the mean value of the distribution that is supposed to follow at the initial state.

In equation (A.3), notice that the observation $\mathbf{z}_n$ depends only on the state $\mathbf{x}_n$, that is, the current state. Moreover, the noise $\mathbf{v}_n$ is independent from the noise $\mathbf{w}_n$ in equation (A.1).

The problem that is solved by Kalman filter is to estimate the system's state based on the observations. We assume knowledge on:

- The form of the system, that is, we know the dimension and the type (discrete, continuous) of vector $\mathbf{x}_n$ but we do not know the exact value of each coordinate.

- The state matrix $\mathbf{F}_n$. This means that we know how state $\mathbf{x}_{n+1}$ is generated from $\mathbf{x}_n$. The state matrix $\mathbf{F}_n$ is not constant in the general case but it can change through time. For instance, if the object is moving with constant velocity then the state of the object can be expressed by its position and its velocity. In this case, the next position of the object is given by:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \\ \Delta x_{n+1} \\ \Delta y_{n+1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ \Delta x_n \\ \Delta y_n \end{bmatrix} + \mathbf{w}_n$$

  where x, y is the position coordinates, Δx and Δy the velocity in each dimension and $\mathbf{w}_n$ the noise.

- Matrix $\mathbf{Q}_n$, which is the covariance matrix of the noise $\mathbf{w}_n$. This matrix may change through time.

- The initial state $\mathbf{x}_{-1}$ of the system. If we do not know the exact initial state, then we can estimate it by the mean value of the distribution describing the initial state.

- The observation matrix $\mathbf{H}_n$, that is how each observation is produced from the state. The observation matrix $\mathbf{H}_n$ may change through time. For instance when we observe an object with constant velocity, we observe only its position. In this case, the observation is given by:

$$\begin{bmatrix} Z_{x_n} \\ Z_{y_n} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_n \\ y_n \\ \Delta x_n \\ \Delta y_n \end{bmatrix} + \mathbf{v}_n$$

where Z_{x_n} and Z_{y_n} are the observation coordinates and $\mathbf{v}_n$ the noise.

- Matrix $\mathbf{R}_n$, which is the covariance matrix of the noise $\mathbf{v}_n$ that affects the observation. This matrix may change through time.

The only thing that we do not know (and must be estimated) is the state of the system $\mathbf{x}_n$, $0 \leq n \leq N$. Depending on the observations that we use in order to estimate the state $\mathbf{x}_n$ the problem is named:

- filtering, if the observations $\mathbf{z}_i$, $0 \leq i \leq n$ are used.
- prediction, if the observations $\mathbf{z}_i$, $0 \leq i \leq n-1$ are used.
- smoothing, if the observations $\mathbf{z}_i$, $0 \leq i \leq N$ are used.

We want to find the state $\mathbf{x}_n$ that fits best in the parameters that we have set.

Now we are going to define the optimal state. We are going to use scalar states, but the same analysis applies to vector states. Let z_i, $0 \leq i \leq n$ be the observations and $\hat{x}_n$ the posterior estimation of the state x_n. In general, the estimation $\hat{x}_n$ is different from the actual state x_n. We define the mean square error function as:

$$J_n = E[(x_n - \hat{x}_n)^2] = E[(\tilde{x}_n)^2] \tag{A.5}$$

where $\tilde{x}_n$ is the estimation error. Equation (A.5) is non negative and non decreasing. The dependance of the cost function J_n from n highlights the non static nature of the procedure. In order to estimate the optimal state $\hat{x}_n$ we need the following two theorems [32].

Theorem A.1 (Conditional Mean Estimator)**.** *If the stochastic processes $\{x_n\}$ and $\{z_n\}$ are jointly Gaussian, then the optimum estimate $\hat{x}_n$ that minimizes the mean square error J_n is the conditional mean estimator:*

$$\hat{x}_n = E[x_n | z_0, \ldots, z_n] \tag{A.6}$$

Theorem A.2 (Principle of orthogonality)**.** *Let the stochastic processes $\{x_n\}$ and $\{z_n\}$ be of zero means, that is:*

$$E[x_n] = E[z_n] = 0, \quad \forall n \tag{A.7}$$

then either the stochastic process $\{x_n\}$ and $\{z_n\}$ are jointly Gaussian, or if the optimal estimate $\hat{x}_n$ is restricted to be a linear function of the observations and the cost function is the mean square error then the optimum estimate $\hat{x}_n$ given the observations z_1, z_2, …, z_n is the orthogonal projection of x_n on the space spanned by these observations.

A.2 Linear Kalman Filter

The linear Kalman filter is a recursive procedure which can compute the optimal solution in linear filtering problems. The solution is produced from the previous solution and the observations. An advantage of this is that we do not have to store all the previous data. Moreover in order to do the filtering we do not need data from future times.

We assume that the model is the same as in section A.1. The objective is to use the extra information provided from the observation $\mathbf{z}_n$ in order to make the estimation of the unknown state $\mathbf{x}_n$ more accurate. Let $\hat{\mathbf{x}}_n^-$ be the *a priori* prediction of the system's state (without the observation $\mathbf{z}_n$) at time n. We can express the posterior estimation of the state $\hat{\mathbf{x}}_n$ (with the observation $\mathbf{z}_n$) as:

$$\hat{\mathbf{x}}_n = \mathbf{G}_n^{(1)}\hat{\mathbf{x}}_n^- + \mathbf{G}_n\mathbf{z}_n \tag{A.8}$$

where matrixes $\mathbf{G}_n^{(1)}$ and $\mathbf{G}_n$ must be computed. The error in estimation (the difference from the actual state) is given by:

$$\tilde{\mathbf{x}}_n = \mathbf{x}_n - \hat{\mathbf{x}}_n \tag{A.9}$$

Applying the principle of orthogonality we can write:

$$E[\tilde{\mathbf{x}}_n\mathbf{z}_i^T] = 0, \quad \forall i = 0, \ldots, n-1 \tag{A.10}$$

By using equations (A.3), (A.8), (A.9) and (A.10) we get:

$$E[(\mathbf{x}_n\text{–}\mathbf{G}_n^{(1)}\hat{\mathbf{x}}_n^- - \mathbf{G}_n\mathbf{H}_n\mathbf{x}_n\text{–}\mathbf{G}_n\mathbf{v}_n)\mathbf{z}_i^T] = 0, \quad \forall i = 0, \ldots, n-1 \tag{A.11}$$

Due to the fact that noise $\mathbf{w}_n$ and $\mathbf{v}_n$ are uncorrelated, we get:

$$E[\mathbf{v}_n\mathbf{z}_n^T] = 0 \tag{A.12}$$

By using equation (A.12) and by adding $\mathbf{G}_n^{(1)}\mathbf{x}_n - \mathbf{G}_n^{(1)}\mathbf{x}_n$, equation (A.11) may be written as:

$$E[(\mathbf{I}\text{–}\mathbf{G}_n\mathbf{H}_n - \mathbf{G}_n^{(1)})\mathbf{x}_n\mathbf{z}_i^T + \mathbf{G}_n^{(1)}(\mathbf{x}_n - \hat{\mathbf{x}}_n^-)\mathbf{z}_i^T] = 0, \quad \forall i = 0, \ldots, n-1 \tag{A.13}$$

where $\mathbf{I}$ is the identity matrix. From the principle of orthogonality we get $E[(\mathbf{x}_n - \hat{\mathbf{x}}_n^-)\mathbf{z}_i^T] = 0$, equation (A.13) can be simplified to:

$$(\mathbf{I}\text{–}\mathbf{G}_n\mathbf{H}_n - \mathbf{G}_n^{(1)})E[\mathbf{x}_n\mathbf{z}_i^T] = 0, \quad \forall i = 0, \ldots, n-1 \tag{A.14}$$

Equation (A.14) must be satisfied for every value of $\mathbf{x}_n$ and $\mathbf{z}_i$. This is true if

$$\mathbf{I}\text{–}\mathbf{G}_n\mathbf{H}_n - \mathbf{G}_n^{(1)} = 0 \quad \Leftrightarrow \quad \mathbf{G}_n^{(1)} = \mathbf{I}\text{–}\mathbf{G}_n\mathbf{H}_n \tag{A.15}$$

By substituting equation (A.15) into (A.8) we can express the posterior estimation of state as:

$$\hat{\mathbf{x}}_n = \hat{\mathbf{x}}_n^- + \mathbf{G}_n(\mathbf{z}_n\text{–}\mathbf{H}_n\hat{\mathbf{x}}_n^-) \tag{A.16}$$

where $\mathbf{G}_n$ is called Kalman gain.

Kalman gain $\mathbf{G}_n$ must be explicitly expressed. From the principle of orthogonality, we have

$$E[(\mathbf{x}_n - \hat{\mathbf{x}}_n)\mathbf{z}_i^T] \quad \Leftrightarrow \quad E[(\mathbf{x}_n - \hat{\mathbf{x}}_n)\hat{\mathbf{z}}_i^T] \tag{A.17}$$

where $\hat{\mathbf{z}}_i^T$ is the estimation of $\mathbf{z}_i^T$ by using the previous estimations $\mathbf{z}_0, \ldots, \mathbf{z}_{n-1}$. We define $\tilde{\mathbf{z}}_n = \mathbf{z}_n - \hat{\mathbf{z}}_n$, which represents the contribution of $\mathbf{z}_n$ in the information about the observation. This can be expressed as

$$\tilde{\mathbf{z}}_n = \mathbf{z}_n\text{–}\mathbf{H}_n\hat{\mathbf{x}}_n^- = \mathbf{H}_n\mathbf{x}_n + \mathbf{v}_n\text{–}\mathbf{H}_n\hat{\mathbf{x}}_n^- = \mathbf{v}_n + \mathbf{H}_n\tilde{\mathbf{x}}_n^- \tag{A.18}$$

By subtracting the two parts of equation (A.17) we get:

$$E[(\mathbf{x}_n - \hat{\mathbf{x}}_n)\tilde{\mathbf{z}}_n^T] = 0 \tag{A.19}$$

Combining equations (A.3) and (A.16) we may express the error $\mathbf{x}_n - \hat{\mathbf{x}}_n$ as

$$\mathbf{x}_n - \hat{\mathbf{x}}_n = \tilde{\mathbf{x}}_n^- - \mathbf{G}_n(\mathbf{H}_n\tilde{\mathbf{x}}_n^- + \mathbf{v}_n) = (\mathbf{I} - \mathbf{G}_n\mathbf{H}_n)\tilde{\mathbf{x}}_n^- - \mathbf{G}_n\mathbf{v}_n \tag{A.20}$$

By substituting equations (A.18) and (A.20) into (A.19) we get:

$$E[((\mathbf{I} - \mathbf{G}_n\mathbf{H}_n)\tilde{\mathbf{x}}_n^- - \mathbf{G}_n\mathbf{v}_n)(\mathbf{H}_n\tilde{\mathbf{x}}_n^- + \mathbf{v}_n)] = 0 \tag{A.21}$$

Due to the fact that the noise $\mathbf{v}_n$ is independent of the state $\mathbf{x}_n$ and $\tilde{\mathbf{x}}_n^-$, equation (A.21) may be written as

$$(\mathbf{I} - \mathbf{G}_n\mathbf{H}_n)E[\tilde{\mathbf{x}}_n^-\tilde{\mathbf{x}}_n^{-T}]\mathbf{H}_n^T - \mathbf{G}_nE[\mathbf{v}_n\mathbf{v}_n^T] = 0 \tag{A.22}$$

We define the *a priori* covariance matrix as

$$\mathbf{P}_n^- = E[(\mathbf{x}_n - \hat{\mathbf{x}}_n^-)(\mathbf{x}_n - \hat{\mathbf{x}}_n^-)^T] = E[\tilde{\mathbf{x}}_n^-\tilde{\mathbf{x}}_n^{-T}] \tag{A.23}$$

By using equations (A.4) and (A.23) we can express (A.22) as

$$(\mathbf{I} - \mathbf{G}_n\mathbf{H}_n)\mathbf{P}_n^-\mathbf{H}_n^T - \mathbf{G}_n\mathbf{R}_n = 0 \tag{A.24}$$

and by solving for $\mathbf{G}_n$ we get

$$\mathbf{G}_n = \mathbf{P}_n^-\mathbf{H}_n^T[\mathbf{H}_n\mathbf{P}_n^-\mathbf{H}_n^T + \mathbf{R}_n]^{-1} \tag{A.25}$$

Equation (A.25) expresses $\mathbf{G}_n$ as a function of the *a priori* covariance matrix $\mathbf{P}_n^-$. To complete the computation, we must express the error covariance propagation, which describes the effect of time. This propagation involves two stages:

1. The *a priori* covariance matrix $\mathbf{P}_n^-$ at time n is given by equation (A.23). By using $\mathbf{P}_n^-$ we can compute the *a posteriori* covariance matrix $\mathbf{P}_n$ as

$$\mathbf{P}_n = E[(\mathbf{x}_n - \hat{\mathbf{x}}_n)(\mathbf{x}_n - \hat{\mathbf{x}}_n)^T] = E[\tilde{\mathbf{x}}_n\tilde{\mathbf{x}}_n^T] \tag{A.26}$$

2. Given the *a posteriori* covariance matrix $\mathbf{P}_{n-1}$ we can compute the *a priori* matrix $\mathbf{P}_n^-$ at time n.

In order to compute $\mathbf{P}_n$ we substitute equation (A.20) into (A.26), and due to the fact that the noise $\mathbf{v}_n$ is independent of the *a priori* estimation error $\tilde{\mathbf{x}}_n^-$ we get

$$\begin{aligned}\mathbf{P}_n &= (\mathbf{I} - \mathbf{G}_n\mathbf{H}_n)E[\tilde{\mathbf{x}}_n^-\tilde{\mathbf{x}}_n^{-T}](\mathbf{I} - \mathbf{G}_n\mathbf{H}_n)^T + \mathbf{G}_nE[\mathbf{v}_n\mathbf{v}_n^T]\mathbf{G}_n^T \\ &= (\mathbf{I} - \mathbf{G}_n\mathbf{H}_n)\mathbf{P}_n^-(\mathbf{I} - \mathbf{G}_n\mathbf{H}_n)^T + \mathbf{G}_n\mathbf{R}_n\mathbf{G}_n^T\end{aligned} \tag{A.27}$$

By using $\mathbf{G}_n\mathbf{R}_n = (\mathbf{I}–\mathbf{G}_n\mathbf{H}_n)\mathbf{P}_n^-\mathbf{H}_n^T$ form (A.25) and substituting this to (A.27) we get

$$\begin{aligned}\mathbf{P}_n &= (\mathbf{I}–\mathbf{G}_n\mathbf{H}_n)\mathbf{P}_n^-–(\mathbf{I}–\mathbf{G}_n\mathbf{H}_n)\mathbf{P}_n^-\mathbf{H}_n^T\mathbf{G}_n^T + \mathbf{G}_n\mathbf{R}_n\mathbf{G}_n^T \\ &= (\mathbf{I}–\mathbf{G}_n\mathbf{H}_n)\mathbf{P}_n^- - \mathbf{G}_n\mathbf{R}_n\mathbf{G}_n^T + \mathbf{G}_n\mathbf{R}_n\mathbf{G}_n^T \\ &= (\mathbf{I}–\mathbf{G}_n\mathbf{H}_n)\mathbf{P}_n^- \end{aligned} \tag{A.28}$$

In order to compute $\mathbf{P}_n^-$ we must notice that the *a priori* estimate of the state is given by:

$$\hat{\mathbf{x}}_n^- = \mathbf{F}_n\hat{\mathbf{x}}_{n-1} \tag{A.29}$$

By using equations (A.1) and (A.29) we get the *a priori* estimation error

$$\begin{aligned}\tilde{\mathbf{x}}_n^- &= \mathbf{x}_n - \hat{\mathbf{x}}_n^- \\ &= (\mathbf{F}_n\mathbf{x}_{n-1} + \mathbf{w}_{n-1})–(\mathbf{F}_n\hat{\mathbf{x}}_{n-1}) \\ &= \mathbf{F}_n(\mathbf{x}_{n-1} - \hat{\mathbf{x}}_{n-1}) + \mathbf{w}_{n-1} \\ &= \mathbf{F}_n\tilde{\mathbf{x}}_{n-1} + \mathbf{w}_{n-1} \end{aligned} \tag{A.30}$$

Substituting (A.30) in (A.23) and due to the fact that the noise $\mathbf{w}_n$ is independent of $\hat{\mathbf{x}}_{n-1}$ we get

$$\begin{aligned}\mathbf{P}_n^- &= \mathbf{F}_n E[\tilde{\mathbf{x}}_{n-1}\tilde{\mathbf{x}}_{n-1}^T]\mathbf{F}_n^T + E[\mathbf{w}_{n-1}\mathbf{w}_{n-1}^T] \\ &= \mathbf{F}_n\mathbf{P}_{n-1}\mathbf{F}_n^T + \mathbf{Q}_{n-1} \end{aligned} \tag{A.31}$$

Using equations (A.16), (A.25), (A.28) and (A.31) we can construct the recursive procedure, shown in algorithm 8. We choose the initial state as $\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$ and the inital *a posteriori* covariance matrix as $\mathbf{P}_0 = E[(\mathbf{x}_0–E[\mathbf{x}_0])(\mathbf{x}_0–E[\mathbf{x}_0)^T]$, due to the fact that we have no other information about the distribution.

Kalman filter assumes that the noise is white Gaussian. The noise $\mathbf{w}_n$ increase the uncertainty (spreads the distribution), the state evolution matrix $\mathbf{F}_n$ translates the distribution and the observation $\mathbf{z}_n$ decreases the uncertainty.

A.3 Extended Kalman Filter

In cases where the system model is not linear, the linear Kalman filter is not the optimal solution. However, we can use the linear Kalman filter, if we make a linear approximation of the system [32]. The result is known as extended Kalman filter.

Let a nonlinear dynamical system described by:

$$\mathbf{x}_{n+1} = \mathbf{f}(n, \mathbf{x}_n) + \mathbf{w}_n \tag{A.32}$$

$$\mathbf{z}_n = \mathbf{h}(n, \mathbf{x}_n) + \mathbf{v}_n \tag{A.33}$$

where $\mathbf{w}_n$ and $\mathbf{v}_n$ are independent white Gaussian noise processes with covariance matrices $\mathbf{R}_n$ and $\mathbf{Q}_n$. Functions $f(n, \mathbf{x}_n)$ and $h(n, \mathbf{x}_n)$ express that the state evolution matrix is non linear

Algorithm 8 Kalman Filter

1 Initialization:

$$\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$$

$$\mathbf{P}_0 = E[(\mathbf{x}_0 – E[\mathbf{x}_0])(\mathbf{x}_0 – E[\mathbf{x}_0])^T]$$

2 Prediction:

$$\hat{\mathbf{x}}_n^- = \mathbf{F}_n \hat{\mathbf{x}}_{n-1}$$

$$\mathbf{P}_n^- = \mathbf{F}_n \mathbf{P}_{n-1} \mathbf{F}_n^T + \mathbf{Q}_n$$

$$\mathbf{G}_n = \mathbf{P}_n^- \mathbf{H}_n^T [\mathbf{H}_n \mathbf{P}_n^- \mathbf{H}_n^T + \mathbf{R}_n]^{-1}$$

3 Estimation:

$$\hat{\mathbf{x}}_n = \hat{\mathbf{x}}_n^- + \mathbf{G}_n(\mathbf{z}_n – \mathbf{H}_n \hat{\mathbf{x}}_n^-)$$

$$\mathbf{P}_n = (\mathbf{I} – \mathbf{G}_n \mathbf{H}_n) \mathbf{P}_n^-$$

4 Increase time by one and go to step 2.

and time dependant. The main idea of the extended Kalman filter is to linearize the model described by equations (A.32) and (A.33) around the most recent state prediction $\hat{\mathbf{x}}_n^-$ or state estimation $\hat{\mathbf{x}}_n$. After the approximation is made, the linear Kalman filter can be used.

More specifically, the approximation may be done in two steps:

1. Matrices $\mathbf{F}_n$ and $\mathbf{H}_n$ are constructed as follows:

$$\mathbf{F}_n = \left. \frac{\partial \mathbf{f}(n, \mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}_n} \tag{A.34}$$

$$\mathbf{H}_n = \left. \frac{\partial \mathbf{h}(n, \mathbf{x})}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}_n^-} \tag{A.35}$$

2. After the evaluation of matrices $\mathbf{F}_n$ and $\mathbf{H}_n$ we can use Taylor approximation around $\hat{\mathbf{x}}_n$ and $\hat{\mathbf{x}}_n^-$ respectively.

$$\mathbf{f}(n, \mathbf{x}_n) \approx \mathbf{f}(n, \hat{\mathbf{x}}_n) + \mathbf{F}_n \cdot (\mathbf{x} – \hat{\mathbf{x}}_n) \tag{A.36}$$

$$\mathbf{h}(n, \mathbf{x}_n) \approx \mathbf{h}(n, \hat{\mathbf{x}}_n^-) + \mathbf{H}_n \cdot (\mathbf{x} – \hat{\mathbf{x}}_n^-) \tag{A.37}$$

Using the approximations of the above two steps, we can approximate equations (A.32) and (A.33) by

$$\mathbf{x}_{n+1} = \mathbf{F}_n \mathbf{x}_n + \mathbf{w}_n + \mathbf{c}_n \tag{A.38}$$

$$\mathbf{z}_n = \mathbf{H}_n \mathbf{x}_n + \mathbf{v}_n + \mathbf{d}_n \tag{A.39}$$

where

$$\mathbf{c}_n = \mathbf{f}(n, \hat{\mathbf{x}}_n) – \mathbf{F}_n \hat{\mathbf{x}}_n \tag{A.40}$$

$$\mathbf{d}_n = \mathbf{h}(n, \hat{\mathbf{x}}_n^-) – \mathbf{H}_n \hat{\mathbf{x}}_n^- \tag{A.41}$$

Using equations (A.38), (A.39), (A.40) and (A.41) we present the extended Kalman filter in algorithm 9.

Algorithm 9 Extended Kalman filter

1 Initialization:

$$\hat{\mathbf{x}}_0 = E[\mathbf{x}_0]$$

$$\mathbf{P}_0 = E[(\mathbf{x}_0 – E[\mathbf{x}_0])(\mathbf{x}_0 – E[\mathbf{x}_0])^T]$$

2 Prediction:

$$\hat{\mathbf{x}}_n^- = \mathbf{f}(n, \hat{\mathbf{x}}_{n-1})$$

$$\mathbf{F}_{n-1} = \left.\frac{\partial \mathbf{f}(n-1, \mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_{n-1}}$$

$$\mathbf{H}_n = \left.\frac{\partial \mathbf{h}(n, \mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_n^-}$$

$$\mathbf{P}_n^- = \mathbf{F}_{n-1}\mathbf{P}_{n-1}\mathbf{F}_{n-1}^T + \mathbf{Q}_n$$

$$\mathbf{G}_n = \mathbf{P}_n^-\mathbf{H}_n^T[\mathbf{H}_n\mathbf{P}_n^-\mathbf{H}_n^T + \mathbf{R}_n]^{-1}$$

3 Estimation:

$$\hat{\mathbf{x}}_n = \hat{\mathbf{x}}_n^- + \mathbf{G}_n(\mathbf{z}_n – \mathbf{h}(n, \hat{\mathbf{x}}_n^-))$$

$$\mathbf{P}_n = (\mathbf{I} – \mathbf{G}_n\mathbf{H}_n)\mathbf{P}_n^-$$

4 increase time by one and go to step 2.

APPENDIX B

THE PARTICLE FILTER

This family of algorithms, which are based on particle filtering, assume multiple states in every time step, which can lead to successfully tracking an object even if some states are erroneously predicted. Moreover, the noise is not restricted to be white Gaussian.

B.1 The Condensation algorithm

In this section the Condensation algorithm [54] is presented.

B.1.1 Discrete-time propagation of state density

The Condensation algorithm assumes a discrete time system, with the system state described by $\mathbf{x}_n$ and the observations by $\mathbf{z}_n$ at time n. There is no assumption about the distributions of $\mathbf{x}_n$ and $\mathbf{z}_n$. Each state depends only on the immediately previous state:

$$p(\mathbf{x}_{n+1}|\mathbf{x}_1, \ldots, \mathbf{x}_n) = p(\mathbf{x}_{n+1}|\mathbf{x}_n) \tag{B.1}$$

The observations $\mathbf{z}_n$ are independent from each other and they depend only on the associated state, that is:

$$p(\mathbf{z}_1, \ldots, \mathbf{z}_n, \mathbf{x}_{n+1}|\mathbf{x}_1, \ldots, \mathbf{x}_n) = p(\mathbf{x}_{n+1}|\mathbf{x}_1, \ldots, \mathbf{x}_n) \prod_{i=1}^{n} p(\mathbf{z}_i|\mathbf{x}_i) \tag{B.2}$$

By integrating both sides of the equation over $\mathbf{x}_{n+1}$, we get:

$$p(\mathbf{z}_1, \ldots, \mathbf{z}_n|\mathbf{x}_1, \ldots, \mathbf{x}_n) = \prod_{i=1}^{n} p(\mathbf{z}_i|\mathbf{x}_i) \tag{B.3}$$

The observation procedure depends only on $p(\mathbf{z}_n|\mathbf{x}_n)$ at time n.

The probability of a state $\mathbf{x}_n$ is given by:

$$p(\mathbf{x}_n|\mathbf{z}_1,\ldots,\mathbf{z}_n) = k_n p(\mathbf{z}_n|\mathbf{x}_n)p(\mathbf{x}_n|\mathbf{z}_1,\ldots,\mathbf{z}_{n-1}) \tag{B.4}$$

where

$$p(\mathbf{x}_n|\mathbf{z}_1,\ldots,\mathbf{z}_{n-1}) = \int_{\mathbf{x}_{n-1}} p(\mathbf{x}_n|\mathbf{x}_{n-1})p(\mathbf{x}_{n-1}|\mathbf{z}_1,\ldots,\mathbf{z}_{n-1}) \tag{B.5}$$

and k_n a normalization constant, which is independent of $\mathbf{x}_n$.

Equation (B.4) is the Bayes rule for the posterior probability. The prior probability given by equation (B.5) is a prediction of the state $\mathbf{x}_n$, which is computed from the posterior probability $p(\mathbf{x}_{n-1}|\mathbf{z}_1,\ldots,\mathbf{z}_{n-1})$ of the state $\mathbf{x}_{n-1}$ and the single step probability from state $\mathbf{x}_{n-1}$ to $\mathbf{x}_n$. In equation (B.4) we multiply by $p(\mathbf{z}_n|\mathbf{x}_n)$ in order to take into account the observation $\mathbf{z}_n$. These probabilities are non Gaussian. The problem now is how to apply a nonlinear filter to evaluate the state density over time. This is computationally expensive and an approximation is necessary.

B.1.2 Factored sampling

The objective is to locate a target described by the state $\mathbf{x}_n$ with probability $p(\mathbf{x}_n)$ and having observation $\mathbf{z_n}$ of a single image. The posterior probability $p(\mathbf{x}_n|\mathbf{z}_n)$ gives the estimation of the state $\mathbf{x}_n$ using the observation $\mathbf{z}_n$. From Bayes rule we get:

$$p(\mathbf{x}_n|\mathbf{z}_n) = k_n p(\mathbf{z}_n|\mathbf{x}_n)p(\mathbf{x}_n) \tag{B.6}$$

where k_n is a normalization factor independent of $\mathbf{x}_n$. The factored sampling algorithm generates a random variable $\mathbf{x}_n$ from a distribution $\tilde{p}(\mathbf{x}_n)$ that approximates $p(\mathbf{x}_n|\mathbf{z}_n)$. First a set with M samples $\mathbf{s}_n^1,\ldots,\mathbf{s}_n^M$ is constructed from the prior probability $p(\mathbf{x}_n)$. Next, an index $i \in \{1,\ldots,M\}$ is chosen with probability π^i, where

$$\pi_n^i = \frac{p(\mathbf{z}_n|\mathbf{s}_n^i)}{\sum_{j=1}^M p(\mathbf{z}_n|\mathbf{s}_n^j)} \tag{B.7}$$

If $\mathbf{x}_n^i$ is chosen in this way, it is produced from a distribution which approximates $p(\mathbf{x}_n|\mathbf{z}_n)$ as N increases.

The posterior mean properties $E[g(\mathbf{x}_n)|\mathbf{z}_n]$ can be generated from samples $\mathbf{s}_n^1,\ldots,\mathbf{s}_n^M$ using $p(\mathbf{z}_n|\mathbf{x}_n)$ as weight:

$$E[g(\mathbf{x}_n)|\mathbf{z}_n] \approx \frac{\sum_{i=1}^M g(\mathbf{s}_n^i)p(\mathbf{z}_n|\mathbf{s}_n^i)}{\sum_{i=1}^M p(\mathbf{z}_n|\mathbf{s}_n^i)} \tag{B.8}$$

For instance, the mean value can be approximated by using $g(\mathbf{x}_n) = \mathbf{x}_n$ and the variance by using $g(\mathbf{x}_n) = \mathbf{x}_n\mathbf{x}_n^T$.

B.1.3 The Condensation algorithm

The Condensation algorithm [54] is based on factored sampling. At every time n, factored sampling is employed and the result is the weighted sum of $\{\mathbf{s}_n^i,\ i = 1, \ldots, M\}$ with weights π_n^i, which approximates the distribution $p(\mathbf{x}_n|\mathbf{z}_1, \ldots, \mathbf{z}_n)$. In order to produce the set $\{\mathbf{s}_n^i,\ i = 1, \ldots, M\}$ we must know the prior probability $p(\mathbf{x}_n|\mathbf{z}_n, \ldots, \mathbf{z}_{n-1})$. Usually this probability function is not in closed form. We approximate the probability $p(\mathbf{x}_{n-1}|\mathbf{z}_1, \ldots, \mathbf{z}_{n-1})$ with the set $\{(\mathbf{s}_{n-1}^i, \pi_{n-1}^i),\ i = 1, \ldots, M\}$, which is the algorithm's result at time $n-1$. Therefore, the prediction $p(\mathbf{x}_n|\mathbf{z}_1, \ldots, \mathbf{z}_{n-1})$ can be computed using (B.5).

The objective is to approximate $p(\mathbf{x}_n|\mathbf{z}_1, \ldots, \mathbf{z}_n)$ with low computational cost. The first thing to do is the sampling (with replacement) M times from the set $\{(\mathbf{s}_{n-1}^i, \pi_{n-1}^i),\ i = 1, \ldots, M\}$, choosing a specific index i with probability π_{n-1}^i. Some elements, especially these with large weights may be chosen many times, giving multiple copies of the same element. On the other hand, elements with small weight may not be chosen at all.

Every element chosen in the new set is used by the prediction step. Firstly, every element of the set undergoes a drift which is deterministic for all the elements. After that, each element is diffused randomly, and multiply chosen elements split. In this point the set $\{\mathbf{s}_n^i,\ i = 1, \ldots, M\}$ has been constructed. This set is an approximation of the *a priori* probability $p(\mathbf{x}_n|\mathbf{z}_1, \ldots, \mathbf{z}_{n-1})$ at time n. Finally, the observation step of factored sampling is applied, generating weights from the observation density $p(\mathbf{z}_n|\mathbf{x}_n)$ to produce the set $\{(\mathbf{s}_n^i, \pi_n^i),\ i = 1, \ldots, M\}$ at time M. This procedure is summarized in algorithm 10.

Algorithm 10 Condensation algorithm

From the set $\{(\mathbf{s}_{n-1}^i, \pi_{n-1}^i, c_{n-1}^i),\ i = 1, \ldots, M\}$ at time $n-1$ construct the set $\{(\mathbf{s}_n^i, \pi_n^i, c_n^i),\ i = 1, \ldots, M\}$ at time n.

Construct the i^{th} sample by:

1 Selection: select a sample $\mathbf{s}'^i_n$ by generating a random number $r \in [0, 1]$ and choose the element $\mathbf{s}_{n-1}^j$ for which j is the minimum that satisfies $c_{n-1}^j \geq r$.

2 Prediction: sample from $p(\mathbf{x}_n|\mathbf{x}_{n-1} = \mathbf{s}'^i_n)$ to produce $\mathbf{s}_n^i$.

3 Estimation: using the observation $\mathbf{z}_n$ compute the weight $\pi_n^i = p(\mathbf{z}_n|\mathbf{x}_n = \mathbf{s}_n^i)$. Normalise so that $\sum_{i=1}^M \pi_n^i = 1$ and compute c_n^i where $c_n^0 = 0$ or $c_n^i = c_n^{i-1} + \pi_n^i, (i = 1, \ldots, M)$.

After the production of the M samples, the state estimation at time n can be obtained $E[f(\mathbf{x}_n)] = \sum_{i=1}^M \pi_n^i f(\mathbf{s}_n^i)$. For the mean position $f(\mathbf{x}) = \mathbf{x}$ can be used.

In algorithm 10 the cumulative weights c_{n-1}^i (constructed in step 3) are used in order to achieve efficient sampling in step 1. Moreover, in step 2 (prediction) the single step evolution

probability could be given from a formula like $\mathbf{s}_n^i = \mathbf{A}\mathbf{s}i_n^i + \mathbf{B}\mathbf{w}_n^i$, where $\mathbf{w}_n^i$ is a vector following Gaussian distribution.

An advantage of condensation algorithm is its simplicity despite its general appliance. In order to use the Condensation algorithm we must set:

- the initial state $\{(\mathbf{s}_0^i, \pi_0^i),\ i = 1, \ldots, M\}$.
- the single step evolution function $p(\mathbf{x}_n = \mathbf{s}_n^i | \mathbf{x}_{n-1}^j)$. More specifically, we do not need to evaluate $p(\mathbf{x}_n = \mathbf{s}_n^i | \mathbf{x}_{n-1}^j)$ but we must be able to sample from it. For example if the relation between the new position and the old position could be $\mathbf{x}_{n+1} - \bar{\mathbf{x}} = \mathbf{A}(\mathbf{x}_n - \hat{\mathbf{x}}) + \mathbf{B}\mathbf{w}_n$, where $\bar{\mathbf{x}}$ is the mean state, $\mathbf{A}$, $\mathbf{B}$ matrices and $\mathbf{w}_n$ the noise, then sampling could be done as
$$\mathbf{x}_{n+1} = \bar{\mathbf{x}} + \mathbf{A}(\mathbf{x}_n - \hat{\mathbf{x}}) + \mathbf{B}\mathbf{w}_n$$
- the density function $p(\mathbf{z}_n | \mathbf{x}_n) = \mathbf{x}_n^i$. For example if we assume that the observation is normally located around the state $\mathbf{x}_n$, then we can use
$$p(\mathbf{z}_n | \mathbf{x}_n) = \frac{1}{\sqrt{2\pi}\sigma} \exp - \frac{(\mathbf{z}_n - \mathbf{x}_n)^2}{2\sigma^2}$$
where σ is the covariance of the distribution in one dimension case.

B.2 The ICondensation algorithm

In the condensation algorithm the samples $\mathbf{s}_n^i$ are defined by the samples from the previous time $\{(\mathbf{s}_{n-1}^i, \pi_{n-1}^i)\}$ and the single step evolution probability $p(\mathbf{x}_n = \mathbf{s}_n^i | \mathbf{x}_{n-1}^j)$. The parts of the image from which samples are taken are being set before any observation is made. This is appropriate when the samples $\mathbf{s}_n^i$ approximate the state density accurate. But in general, the state density change through time and the random movement of the object gives a non zero probability in many positions around it. In the desired case this will give samples around the object with high probability and we will be able to track the object. But the usage of finite number of samples will lead to samples only near the object. This means that a large number of samples will exist in a specific area (with every one of them has large probability) and less or no at all samples in other areas. To make the algorithm more robust in abrupt motion changes, we increase the number of samples, but this would make the algorithm slower. We will employ importance sampling for a better approach.

B.2.1 Importance sampling

Importance sampling improves the efficiency of factored sampling in section B.1.2 and it is appropriate when the form of an importance function $g_n(\mathbf{x}_n)$ is known. The main idea is to use the function $g_n(\mathbf{x}_n)$, instead of $p(\mathbf{x}_n)$, in order to produce samples $\mathbf{s}_n^i$ in areas where the object is more likely to appear. Using this technique we can reduce the production of low weight samples, which are unlikely to be selected.

A correction term f_n/g_n is added to the sample weights giving

$$\pi_n^i = \frac{f_n(\mathbf{s}_n^i)}{g_n(\mathbf{s}_n^i)} p(\mathbf{z}_n|\mathbf{x}_n = \mathbf{s}_n^i) \tag{B.9}$$

This term ensures that for large sample number M, importance sampling has no effect on the consistency of the approximation function $\tilde{p}(\mathbf{x}_n|\mathbf{z}_n)$. Any importance function $g_n(\mathbf{x}_n)$ could be chosen and if the number of samples M is large enough then $\tilde{p}(\mathbf{x}_n|\mathbf{z}_n)$ is a good approximation of $p(\mathbf{x}_n|\mathbf{z}_n)$. Importance sampling purpose is to improve the accuracy of $\tilde{p}(\mathbf{x}_n|\mathbf{z}_n)$ for a given (small) number of samples M. Since samples are drawn from $g_n(\mathbf{x}_n)$ it plays the part of a probability density, although it does not necessarily correspond to the distribution of any particular random variable.

B.2.2 The ICondensation algorithm

Importance sampling can be applied to Condensation (section B.1) leading to ICondensation algorithm [55]. The importance function at time n is given by $g_n(\mathbf{x}_n)$. In Condensation samples are drawn from:

$$f_n(\mathbf{s}_n^i) = \tilde{p}(\mathbf{x}_n = \mathbf{s}_n^i|\mathbf{z}_1, \ldots, \mathbf{z}_{n-1}) = \sum_{j=1}^{M} \pi_{n-1}^j p(\mathbf{x}_n = \mathbf{s}_n^i|\mathbf{x}_{n-1} = \mathbf{s}_{n-1}) \tag{B.10}$$

Instead of sampling from $\tilde{p}(\mathbf{x}_n|\mathbf{z}_1, \ldots, \mathbf{z}_{n-1})$, samples $\mathbf{s}_n^i$ are drawn from $g_n(\mathbf{x}_n)$ with weights:

$$\pi_n^i = \frac{f_n(\mathbf{s}_n^i)}{g_n(\mathbf{s}_n^i)} p(\mathbf{z}_n|\mathbf{x}_n = \mathbf{s}_n^i) \tag{B.11}$$

Although the samples are drawn from g_n, the set $\{(\mathbf{s}_n^i, \pi_n^i)\}$ approximates the distribution $p(\mathbf{x}_n|\mathbf{z}_n)$. To use (B.11) we must be able to evaluate $p(\mathbf{x}_n|\mathbf{x}_{n-1})$.

The use of the importance function g_n may omit some peaks of $p(\mathbf{z}_n|\mathbf{x}_n)$. To eliminate this drawback, we can use factored sampling to produce some samples and importance sampling the rest. In this way the object will be successfully tracked even if one of the two subsets fail to estimate the object.

We can also reinitialize the state with probability q, which is independent of the past observations. This can locate an object when it enters the image or it was lost. The amended model is of the form:

$$\tilde{p}'(\mathbf{x}_n|\mathbf{z}_{n-1}) = (1-q)\tilde{p}(\mathbf{x}_n|\mathbf{z}_{n-1}) + qp(\mathbf{x}_n) \tag{B.12}$$

where $p(\mathbf{x}_n)$ is the initialisation prior. Using this model, we use factored sampling with probability $1-q-r$, importance sampling with probability r or we reinitialize with probability q. In the absence of knowledge about $p(\mathbf{x}_n)$ we can use $p(\mathbf{x}_n) = g_n(\mathbf{x}_n)$. We summarize ICondensation procedure in algorithm 11.

In order to use ICondensation algorithm we must set:

- the initial state $\{(\mathbf{s}_0^i, \pi_0^i),\ i = 1, \ldots, M\}$.

Algorithm 11 ICondensation algorithm

From the set $\{(\mathbf{s}_{n-1}^i, \pi_{n-1}^i),\ i = 1, \ldots, M\}$ at time $n-1$ construct the set $\{(\mathbf{s}_n^i, \pi_n^i),\ i = 1, \ldots, M\}$ at time n. The importance function $g_n(\mathbf{x}_n)$ is known.

Construct the i^{th} sample by:

1 Selection: choose a uniform random number $a \in [0, 1)$.

2 Sampling: sample from $\tilde{p}'(\mathbf{x}_n|\mathbf{z}_{n-1})$ as follows:

1. If $a < q$ use the initialisation prior. Choose $\mathbf{s}_n^i$ by sampling from $g_n(\mathbf{x}_n)$ and set the importance correction factor $\lambda_n^i = 1$.
2. If $q \leq a \leq q + r$ use importance sampling. Choose $\mathbf{s}_n^i$ by sampling from $g_n(\mathbf{x}_n)$ and set the importance correction factor $\lambda_n^i = f_n(\mathbf{s}_n^i)/g_n(\mathbf{s}_n^i)$, where

$$f_n(\mathbf{s}_n^i) = \sum_{j=1}^{M} \pi_{n-1}^j p(\mathbf{x}_n = \mathbf{s}_n^i | \mathbf{x}_{n-1} = \mathbf{s}_{n-1}^j)$$

3. If $a \geq q + r$ use factored sampling from Condensation algorithm. Choose a sample $\mathbf{s}_{n-1}^i$ with probability π_{n-1}^i, then choose $\mathbf{s}_n^i$ sampling from $p(\mathbf{x}_n|\mathbf{x}_{n-1} = \mathbf{s}_{n-1}^i)$ and set the importance factor $\lambda_n^i = 1$.

3 Estimation: using the observation $\mathbf{z}_n$ we update the importance sampling correction term:

$$\pi_n^i = \lambda_n^i p(\mathbf{z}_n|\mathbf{x}_n = \mathbf{s}_n^i)$$

Normalize so that $\sum_{i=1}^{M} \pi_n^i = 1$ and store as $\{(\mathbf{s}_n^i, \pi_n^i)\}$.

- the single step evolution function $p(\mathbf{x}_n|\mathbf{x}_{n-1} = \mathbf{s}_{n-1}^i)$.
- the density function $p(\mathbf{z}_n|\mathbf{x}_n = \mathbf{x}_n^i)$.
- the importance function $g_n(\mathbf{x}_n)$.
- the values for q and r.

APPENDIX C

THE MEAN SHIFT ALGORITHM

Mean shift [30] is a robust optimization method which finds local maxima of a cost function. The term robust refers to the statistical analysis of the algorithm, which ignores data far from the mean value (outliers). Moreover it uses data of a specific area only. It iterates until the local maximum is reached.

C.1 Target representation

In order to represent an object, a feature space must be chosen. The target model is represented by its pdf q in the feature space. The target model is considered as centered at the spatial location $\mathbf{0}$. In the next frame a target candidate is defined at spatial location $\mathbf{y}$ and is characterized by the pdf $p(\mathbf{y})$. The pdfs are approximated by histograms with m bins:

$$\begin{array}{rlll} target\ model: & \hat{\mathbf{q}} = \{\hat{q}_u\}_{u=1\ldots m} & \sum_{u=1}^{m} \hat{q}_u = 1 \\ target\ candidate: & \hat{\mathbf{p}}(\mathbf{y}) = \{\hat{p}_u(\mathbf{y})\}_{u=1\ldots m} & \sum_{u=1}^{m} \hat{p}_u = 1 \end{array}$$

We will denote the similarity function between $\hat{\mathbf{p}}$ and $\hat{\mathbf{q}}$ by

$$\hat{\rho}(\mathbf{y}) \equiv \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] \tag{C.1}$$

Function $\hat{\rho}(\mathbf{y})$ can be viewed as the likelihood. Its maxima indicate the presence of objects having representation similar to $\hat{\mathbf{q}}$. Due to the discrete nature of the image, gradient-based optimization may not be applied and exhaustive search have large computational cost. By masking

the object with an isotropic kernel in the spatial domain, the feature space representations $\hat{\rho}(\mathbf{y})$ becomes a smooth function of $\mathbf{y}$.

The object that we are interested in will be referred as target and is represented by an ellipsoidal region in the image. The ellipse is centered at spatial location $\mathbf{0}$ and is reduced to a unit circle in order to eliminate the influence of different target dimensions. This is done by dividing each dimension with the corresponding ellipse axis h_x and h_y. Let $\{\mathbf{x}_i^*\}_{i=1\ldots n}$ be the normalized pixel locations inside the target. An isotropic kernel with convex and monotonic decreasing kernel profile $k(x)$ [1], assigns smaller weights to pixels farther from the center. By using smaller weights to pixels far from the center, the robustness of the algorithm increases. The function $b : \Re^2 \rightarrow \{1 \ldots m\}$ assigns a pixel to a bin.

The probability of each bin in the target model is given by:

$$\hat{q}_u = C \sum_{i=1}^{n} k(\|\mathbf{x}_i^*\|^2)\delta[b(\mathbf{x}_i^*) - u] \tag{C.2}$$

where δ is the Kronecher delta function and C is a normalization factor which is derived by imposing the condition $\sum_{u=1}^{m} \hat{q}_u = 1$, giving:

$$C = \frac{1}{\sum_{i=1}^{n} k(\|\mathbf{x}_i^*\|^2)} \tag{C.3}$$

In the next frame the object, which is referred as target candidate, is moving to the normalized spatial location $\mathbf{y}$. Let $\{\mathbf{x}_i\}_{1\ldots n_h}$ be the normalized pixel locations os the target candidate. Using the same kernel profile $k(x)$, but with bandwith h, the probability of each bin in the target candidate is given by:

$$\hat{p}_u(\mathbf{y}) = C_h \sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right)\delta[b(\mathbf{x}_i) - u] \tag{C.4}$$

where

$$C_h = \frac{1}{\sum_{i=1}^{n_h} k\left(\left\|\frac{\mathbf{y}-\mathbf{x}_i}{h}\right\|^2\right)} \tag{C.5}$$

being the normalization constant, which does not depend on $\mathbf{y}$ and can be precalculated for different values of h. The bandwidth h defines the scale of the target.

The similarity function (C.1) inherits properties from the kernel function profile $k(x)$ when we use (C.2) and (C.4) to represent the target model and candidate. A differentiable kernel profile yields a differentiable similarity function and efficient gradient-based optimization procedure can be used to find its maxima.

[1] The profile of the kernel K is defined as a function $k : [0, \infty) \rightarrow \Re$ such that $K(x) = k(\|x\|^2)$.

C.2 Histogram Distance

The similarity function defines a distance between the target model and the target candidates. We define the distance between two discrete distributions as

$$d(\mathbf{y}) = \sqrt{1 - \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}]} \tag{C.6}$$

where

$$\hat{\rho}(\mathbf{y}) \equiv \rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\mathbf{y})\hat{q}_u} \tag{C.7}$$

is the sample estimate of the Bhattacharyya coefficient between $\mathbf{p}$ and $\mathbf{q}$ [58]. The Bhattacharyya coefficient is the cosine of the angle between the m-dimensional unit vectors $(\sqrt{\hat{p}_1}, \ldots, \sqrt{\hat{p}_m})^T$ and $(\sqrt{\hat{q}_1}, \ldots, \sqrt{\hat{q}_m})^T$ and it can be proved that it is a metric [30].

C.3 The Mean shift algorithm

The minimization of distance (C.6) is equivalent to the maximization of (C.7). The search for the new target location in the current frame starts at the spatial location $\hat{\mathbf{y}}_0$ of the target in the previous frame. Thus, the probabilities $\{\hat{p}_u(\hat{\mathbf{y}}_0)\}_{u=1\ldots m}$ of the target candidate in the current frame are computed first. Using Taylor expansion around the values of $\hat{p}_u(\hat{\mathbf{y}}_0)$, the linear approximation of (C.7) is given by:

$$\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] \approx \frac{1}{2}\sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{\mathbf{y}}_0)\hat{q}_u} + \frac{1}{2}\sum_{u=1}^{m} \hat{p}_u(\mathbf{y}) \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}} \tag{C.8}$$

This approximation is satisfactory when the target candidate does not change drastically from the initial target model. Using (C.4) in (C.8) we get:

$$\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}] \approx \frac{1}{2}\sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{\mathbf{y}}_0)\hat{q}_u} + \frac{C_h}{2}\sum_{u=1}^{n_h} w_i k\left(\left\|\frac{\mathbf{y} - \mathbf{x}_i}{h}\right\|^2\right) \tag{C.9}$$

where

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{\hat{q}_u}{\hat{p}_u(\hat{\mathbf{y}}_0)}}\delta[b(\mathbf{x}_i) - u] \tag{C.10}$$

In order to minimize (C.6), the second term in (C.10) must be maximized (the first term is independent of $\mathbf{y}$). The second term represents the density estimate computed with kernel profile $k(x)$ at $\mathbf{y}$ in the current frame with the date being weighted by w_i (C.10). The local maxima of this density function can be found using the mean shift method [30]. In this procedure, the kernel is recursively moved from the current location $\hat{\mathbf{y}}_0$ to the new location $\hat{\mathbf{y}}_1$ according to the relation:

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h} \mathbf{x}_i w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{\hat{\mathbf{y}}_0 - \mathbf{x}_i}{h}\right\|^2\right)} \tag{C.11}$$

Algorithm 12 Bhattacharyya coefficient $\rho[\hat{\mathbf{p}}(\mathbf{y}), \hat{\mathbf{q}}]$ maximization

Input: the target model $\{\hat{q}_u\}_{u=1...m}$ and its location $\mathbf{y}_0$ in the previous frame.

1 Initialize the location of the target in the current frame with $\mathbf{y}_0$, compute $\{\hat{p}_u(\hat{\mathbf{y}}_0)\}_{u=1...m'}$ and evaluate

$$\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_0), \hat{\mathbf{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{\mathbf{y}}_0)\hat{q}_u}$$

2 Derive the weights $\{w_i\}_{i=1...n_h}$ according to (C.10).

3 Find the next location of the target candidate according to (C.11).

4 Compute $\{\hat{p}_u(\hat{\mathbf{y}}_1)\}_{u=1...m'}$, and evaluate

$$\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_1), \hat{\mathbf{q}}] = \sum_{u=1}^{m} \sqrt{\hat{p}_u(\hat{\mathbf{y}}_1)\hat{q}_u}$$

5 While $\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_1), \hat{\mathbf{q}}] < \rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_0), \hat{\mathbf{q}}]$

Do $\hat{\mathbf{y}}_1 \leftarrow \frac{1}{2}(\hat{\mathbf{y}}_0 + \hat{\mathbf{y}}_1)$

Evaluate $\rho[\hat{\mathbf{p}}(\hat{\mathbf{y}}_1), \hat{\mathbf{q}}]$

6 If $\|\hat{\mathbf{y}}_1 - \hat{\mathbf{y}}_0\| < \epsilon$ Stop.

Otherwise set $\hat{\mathbf{y}}_0 \leftarrow \hat{\mathbf{y}}_1$ and go to step 2.

---

where $g(x) = -k'(x)$. The mean shift method for target localization is presented in algorithm 12.

The stoping criterion threshold ϵ in step 6 of algorithm 12 is derived by constraining the vectors $\hat{\mathbf{y}}_0$ and $\hat{\mathbf{y}}_1$ to be within the same pixel in original image coordinates. In practice, step 5 can be removed because its role is only to avoid potential numerical problems in the mean shift based maximization. As a result, there is no need to evaluate the Bhattacharyya coefficient in steps 1 and 4. Using kernels with Epanechnikov profile [29]

$$k(x) = \begin{cases} \frac{1}{2}c_d^{-1}(d+2)(1-x) & if \quad x \leq 1 \\ 0 & otherwise \end{cases} \tag{C.12}$$

the derivative of the profile $g(x)$ is constant and (C.11) reduces to

$$\hat{\mathbf{y}}_1 = \frac{\sum_{i=1}^{n_h} \mathbf{x}_i w_i}{\sum_{i=1}^{n_h} w_i} \tag{C.13}$$

In order to handle scale changes we can change the bandwidth h. Denote by h_{prev} the bandwidth in the previous frame. We estimate the bandwidth h_{opt} in the current frame by

running algorithm 12 three times, with bandwidths $h = h_{prev}$, $h = h_{prev} + \Delta h$ and $h = h_{prev} - \Delta h$. A typical value is $\Delta h = 0.1 h_{prev}$. The best result, h_{opt}, yielding the largest Bhattacharyya coefficient, is retained. To avoid oversensitive scale adaptation, the bandwidth associated with the current frame is obtained through filtering:

$$h_{new} = \gamma h_{opt} + (1 - \gamma) h_{prev} \tag{C.14}$$

where the default value for γ is 0.1.

C.4 Background modeling

The background information is important for at least two reasons. First, if some of the target features are also present in the background, their relevance for the localization of the target is diminished. Second, in many applications, it is difficult to exactly delineate the target, and its model might contain background features as well. At the same time, the improper use of the background information may affect the scale selection algorithm, making impossible to measure similarity across scales, hence, to determine the appropriate target scale.

Let $\{\hat{o_u}\}_{u=1...m}$ (with $\sum_{u=1}^{m} \hat{o}_u = 1$) be the discrete representation (histogram) of the background in the feature space and $\hat{o}^*$ ne its smallest nonzero entry. This representation is computed in a region around the target (usual three times the target area). The weights

$$\left\{ v_u = min\left(\frac{\hat{o}^*}{\hat{o}_u}, 1 \right) \right\}_{u=1...m} \tag{C.15}$$

are employed to define a transformation for the representations of the target model and candidates. The transformation diminishes the importance of those features which have low v_u. The new target model representation is defined as

$$\hat{q}_u = C v_u \sum_{i=1}^{n} k(\|\mathbf{x}_i^*\|^2) \delta[b(\mathbf{x}_i^*) - u] \tag{C.16}$$

with the normalization factor C expressed as

$$C = \frac{1}{\sum_{i=1}^{n} k(\|\mathbf{x}_i^*\|^2) \sum_{u=1}^{m} v_u \delta[b(\mathbf{x}_i^*) - u]} \tag{C.17}$$

Similarly, the new target candidate representation is

$$\hat{p}_u(\mathbf{y}) = C_h v_u \sum_{i=1}^{n_h} k\left(\left\| \frac{\mathbf{y} - \mathbf{x}_i}{h} \right\|^2 \right) \delta[b(\mathbf{x}_i) - u] \tag{C.18}$$

where

$$C_h = \frac{1}{\sum_{i=1}^{n_h} k\left(\left\| \frac{\mathbf{y} - \mathbf{x}_i}{h} \right\|^2 \right) \sum_{u=1}^{m} v_u \delta[b(\mathbf{x}_i) - u]} \tag{C.19}$$

APPENDIX D

THE DIFFERENTIAL EARTH MOVER'S DISTANCE

In this section, the differential Earth Mover's Distance algorithm is presented [152]. EMD is a similarity measure that is robust to illumination changes. However, its computational complexity prevents a direct use in many applications. The DEMD algorithm employs the derivative of EMD so that the computation of EMD is not necessary in every position.

D.1 The Earth Mover's Distance

The main idea behind EMD distance is that there are two distributions, with one being a mass of earth and the other a collection of holes. The EMD estimates the minimum amount of work needed to fill the holes with earth. The unit work corresponds to transporting a unit of earth by a unit of ground distance.

The EMD could be employed to compare the color distribution of the object model and that of the object candidates. The distributions are represented by signatures:

$$\mathbf{s} = \{s_u\}_{u=1\ldots m}, \quad s_u = (a_u, w_u) \tag{D.1}$$

where m is the number of clusters in the signature, a_u is the mean and w_u the weight of the u-th cluster.

Representing the target model as a model signature and the target candidates (at position $\mathbf{y}$) as candidate signatures, we denote the ground distance between the u-th cluster in the model

signature and the v-th cluster in the candidate signature as d_{uv} and the flow between them as $f_{uv}(\mathbf{y})$. We use superscript M to denote the object model and C for the object candidate. The signature for the model will be denoted as $\mathbf{s}^M = \{s_u^M\}_{u=1\ldots m^M}, s_u^M = (a_u^M, w_u^M)$ and for the candidate $\mathbf{s}^C(\mathbf{y}) = \{s_v^C(\mathbf{y})\}_{v=1\ldots m^C}, s_v^C(\mathbf{y}) = (a_v^C(\mathbf{y}), w_v^C(\mathbf{y}))$. We use an isotropic kernel function with profile $k(x)$ to produce the weights:

$$w_u^M = \beta \sum_{n=1}^{N} k\left(\left\|\frac{\mathbf{x}_n}{h}\right\|^2\right)\delta[c(\mathbf{x}_n)-u] \tag{D.2}$$

and

$$w_v^C(\mathbf{y}) = \gamma \sum_{n=1}^{N} k\left(\left\|\frac{\mathbf{x}_n - \mathbf{y}}{h}\right\|^2\right)\delta[c(\mathbf{x}_n)-v] \tag{D.3}$$

where β and γ are normalization factors, $c(\mathbf{x}_n)$ assigns a pixel to a cluster and $\delta(x)$ is the Kronecker delta function.

In order to find the image coordinates $\hat{\mathbf{y}}$ for the candidate signature with the smaller EMD distance from the model signature

$$\hat{\mathbf{y}} = arg\min_{\mathbf{y}}(EMD(\mathbf{y}))) \tag{D.4}$$

where $EMD(\mathbf{y})$ is the Earth Mover's distance between the model histogram and the candidate histogram at position $\mathbf{y}$, defined as

$$EMD(\mathbf{y}) = \min_{f_{uv}(\mathbf{y})} \sum_{u=1}^{m^M}\sum_{v=1}^{m^C} d_{uv} f_{uv}(\mathbf{y}) \tag{D.5}$$

subject to

$$\begin{gathered}
\textstyle\sum_{u=1}^{m^M} f_{uv}(\mathbf{y}) = w_v^C(\mathbf{y}), \qquad 1 \le v \le m^C \\
\textstyle\sum_{v=1}^{m^C} f_{uv}(\mathbf{y}) = w_u^M, \qquad 1 \le u \le m^M \\
\textstyle\sum_{u=1}^{m^M}\sum_{v=1}^{m^C} f_{uv}(\mathbf{y}) = 1 \\
f_{uv}(\mathbf{y}) \ge 0, \quad 1 \le u \le m^M, \quad 1 \le v \le m^C
\end{gathered}$$

In the next section a gradient-based method is presented in order to estimate a local minimum of (D.4).

D.2 The DEMD algorithm

In order to solve (D.4) the formula of the derivative of (D.5) must be explicitly computed and a gradient based method can be employed. Since EMD is a linear programming problem, the derivative of the EMD cannot be directly computed. The derivation is done in two stages [152]. First we take the derivative of EMD with respect to the weights and the derivative of the weights with respect to position. More specifically

$$\nabla_{\mathbf{y}} EMD(\mathbf{y}) = \sum_{v=1}^{m^C} \frac{\partial EMD(\mathbf{y})}{\partial w_v^C(\mathbf{y})} \nabla_{\mathbf{y}} w_v^C(\mathbf{y}) \tag{D.6}$$

Equation (D.4) and (D.5) are transformed in a matrix form. We define the vectors $\mathbf{d}$ and $\mathbf{f}(\mathbf{y})$ with dimensions $(m^M \times m^C) \times 1$ containing the distances d_{uv} and flows $f_{uv}(\mathbf{y})$. Stacking the constrains of (D.5) we can form the matrix $\mathbf{H}$ whith 1 and size $(m^M + m^C + 1) \times (m^M \times m^C)$, containing 0 and 1, and the vector $\mathbf{b}(\mathbf{y})$ of size $(m^M + m^C + 1) \times 1$, containing $[(\mathbf{w}^C(\mathbf{y}))^T, (\mathbf{w}^M)^T, 1]^T$. Using these notations the problem can be expressed as

$$EMD(\mathbf{y}) = \min_{\mathbf{f}(\mathbf{y})} \mathbf{d}^T \mathbf{f}(\mathbf{y}) \tag{D.7}$$

subject to

$$\mathbf{H}\mathbf{f}(\mathbf{y}) = \mathbf{b}(\mathbf{y})$$
$$\mathbf{f}(\mathbf{y}) \geq \mathbf{0}$$

Matrix $\mathbf{H}$ is of rank $m^M + m^C + 1$ (or it can be made to be [152]) so there are $m^M + m^C + 1$ basic and $m^M \times m^C - (m^M + m^C + 1)$ non basic variables. Grouping all the basic and all the non basic together we split the flow vector $\mathbf{f}(\mathbf{y})$ into $[\mathbf{f}_B^T(\mathbf{y})\ \mathbf{f}_{NB}^T(\mathbf{y})]^T$, the distance vector $\mathbf{d}$ into $[\mathbf{d}_B^T\ \mathbf{d}_{NB}^T]^T$ and matrix $\mathbf{H}$ into $[\mathbf{H}_B\ \mathbf{H}_{NB}]$. The starting tableau for the simplex method is written as in table D.1.

Table D.1: Starting Tableau

$EMD(\mathbf{y})$	$\mathbf{f}_B(\mathbf{y})$	$\mathbf{f}_{NB}(\mathbf{y})$	**RHS**
1	$-\mathbf{d}_B^T$	$-\mathbf{d}_{NB}^T$	0
$\mathbf{0}$	$\mathbf{H}_B$	$\mathbf{H}_{NB}$	$\mathbf{b}$

Applying the simplex algorithm yields to the optimal tableau shown in table D.2.

Table D.2: Reformulated optimal tableau

$EMD(\mathbf{y})$	$\mathbf{f}_B(\mathbf{y})$	$\mathbf{f}_{NB}(\mathbf{y})$	**RHS**
1	0	$-\mathbf{d}_{NB}^T + \mathbf{d}_B^T\mathbf{H}_B^{-1}\mathbf{H}_{NB}$	$\mathbf{d}_B^T\mathbf{H}_B^{-1}\mathbf{b}$
$\mathbf{0}$	$\mathbf{I}$	$\mathbf{H}_B^{-1}\mathbf{H}_{NB}$	$\mathbf{H}_B^{-1}\mathbf{b}$

Based on table D.2 we analyze the sensitivity if the EMD to a change in the cluster weights $\mathbf{w}^C(\mathbf{y})$ of the candidate signature. Using the second row we have $EMD(\mathbf{y}) = \mathbf{d}_B^T\mathbf{H}_B^{-1}\mathbf{b}$. Assume $\mathbf{b}(\mathbf{y})$ is changed to $\mathbf{b}'(\mathbf{y})$, where $b_i' = b_i + \Delta b_i$, $(1 \leq i \leq m^C)$. The optimal solution becomes:

$$EMD'(\mathbf{y}) = \mathbf{d}_B^T\mathbf{H}_B^{-1}\mathbf{b}' = \mathbf{d}_B^T\mathbf{H}_B^{-1}\mathbf{b} + \mathbf{d}_B^T\mathbf{H}_B^{-1}[0 \ldots 0 \Delta b_i 0 \ldots 0]^T = \mathbf{d}_B^T\mathbf{H}_B^{-1}\mathbf{b} + k_i\Delta b_i \tag{D.8}$$

where $k_i = \sum_{l=1}^{m^M+m^C+1} (\mathbf{d}_B)_l (\mathbf{H}_B^{-1})_{li}$. Therefore,

$$\frac{\partial EMD(\mathbf{y})}{\partial b_i} = \lim_{\Delta b_i \to 0} \frac{\Delta EMD(\mathbf{y})}{\Delta b_i} = \frac{k_i \Delta b_i}{\Delta b_i} = k_i \tag{D.9}$$

As the sum of the cluster weights of the candidate signature is 1, we get:

$$\frac{\partial EMD(\mathbf{y})}{\partial b_i} = k_i - \sum_{j \neq i} k_j \frac{b_j}{\sum_{l \neq i,j} b_l} \tag{D.10}$$

Taking the gradient of the cluster weights in equation (D.3), we have the density gradient of the color feature

$$\nabla_{\mathbf{y}} w_v^C(\mathbf{y}) = \frac{2\gamma}{h^2} \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{y}) g\left(\left\| \frac{\mathbf{x}_n - \mathbf{y}}{h} \right\|^2 \right) \delta[c(\mathbf{x}_n) - v] \tag{D.11}$$

where $g(x)$ is the negative gradient of the kernel profile function, $g(x) = -k'(x)$.

Using equations (D.6), (D.10) and (D.11) we get

$$\nabla_{\mathbf{y}} EMD(\mathbf{y}) = \frac{2\gamma}{h^2} \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{y}) g\left(\left\| \frac{\mathbf{x}_n - \mathbf{y}}{h} \right\|^2 \right) \pi_n \tag{D.12}$$

where π_n is the weight of each pixel:

$$\pi_n = \sum_{v=1}^{m^C} \left(k_v - \sum_{j \neq v} k_j \frac{b_j}{\sum_{l \neq i,j} b_j} \right) \delta[c(\mathbf{x}_n) - v] \tag{D.13}$$

where $k_i = \sum_{l=1}^{m^M + m^C + 1} (\mathbf{d}_B)_l (\mathbf{H}_B^{-1})_{li}$. The distance minimization can be efficiently achieved using algorithm 13.

Algorithm 13 Differential EMD (DEMD)

Input: Object center of the previous frame: $\mathbf{y}_0 = \mathbf{y}^{i-1}$

Output: Object center for the current frame $\mathbf{y}^i$

1 Initialize the location of the object in the current frame with $\mathbf{y}_0$. Evaluate $EMD(\mathbf{y})$ using (D.5).

2 Compute the weights $\{\pi_n\}_{n=1\ldots N}$ for the pixels in the tracking window according to (D.13).

3 Compute the gradient $\nabla_{\mathbf{y}} EMD(\mathbf{y}_0)$ based on (D.12).

4 Move the object along the gradient vector to one of its 8 neighboring pixels $\mathbf{y}_1$. Evaluate $EMD(\mathbf{y}_1)$ using (D.5).

5 If $EMD(\mathbf{y}_1) > EMD(\mathbf{y}_0)$, set $\mathbf{y}_0^i \leftarrow \mathbf{y}_0$ and stop; otherwise, set $\mathbf{y}_0 \leftarrow \mathbf{y}_1$ and go to step 2.

D.3 DEMD extensions

DEMD algorithm successfully tracks an object in many cases. However, there are cases when the object scale changes or occlusion occurs. In order to estimate the scale and the position of

the object, local background scenes as well as foreground objects are modeled and the similarities of both components are considered in order to determine the object state. the goal is to find the object position $\mathbf{y}$ and scale h corresponding to the smallest sum of the EMD for the foreground object and the EMD for the local background scene

$$arg\min_{\mathbf{y},h}(EMD(\mathbf{y},h) + EMD^{BG}(\mathbf{y},h)) \tag{D.14}$$

where the superscript Bg denotes the local background scenes.

To achieve real-time performance the initial object location for the current frame is obtained by the DEMD tracking algorithm. The method and the detailed algorithm for the adjustment step is given in algorithm 14.

Algorithm 14 Algorithm to Adjust Object Scale and Position with Both Foreground and Background Cues

Input: Object center: $\mathbf{y}_0 = \mathbf{y}^i$ returned by algorithm 13. Object scale from the previous frame $h_0 = h^{i-1}$.

Output: Object center $\mathbf{y}^i$ and scale h^i of the current frame.

1 Initialize the object location with $\mathbf{y}_0$, vary h_0 by $+/-10\%$ and evaluate which scale is the best using (D.14).

2 If the scale with the smallest distance h_1 equals h_0, set $h^i \leftarrow h_0$, $\mathbf{y}^i \leftarrow \mathbf{y}_0$ and stop; otherwise, set $h_0 \leftarrow h_1$ and run algorithm 13 to obtain the new location $\mathbf{y}_1$.

3 If $\mathbf{y}_1$ equals $\mathbf{y}_0$, set $h^i \leftarrow h_0$, $\mathbf{y}^i \leftarrow \mathbf{y}_0$ and stop; otherwise, set $\mathbf{y}_0 \leftarrow \mathbf{y}_1$ and go to step 1.

AUTHOR'S PUBLICATIONS

Journal publications

J1 V. Karavasilis, C. Nikou and A. Likas. Visual tracking using the earth mover's distance between Gaussian mixtures and Kalman filtering. Image and Vision Computing, Vol. 29, No 5, pp. 295-305, 2011.

J2 V. Karavasilis, K. Blekas and C. Nikou. A novel framework for motion segmentation and tracking by clustering incomplete trajectories. Computer Vision and Image Understanding, Vol. 116, No 11, pp. 1135-1148, 2012.

J3 V. Karavasilis, C. Nikou and A. Likas, Visual tracking using spatially weighted likelihood of Gaussian mixtures, Computer Vision and Image Understanding, Vol. 140, pp. 43-57, 2015.

J4 V. Karavasilis, C. Nikou and A. Likas, Real time visual tracking using a spatially weighted von Mises mixture model, submitted.

Conference publications

C1 V. Karavasilis, C. Nikou and A. Likas. Visual tracking by adaptive Kalman filtering and mean shift. Proceedings of the 6th Hellenic Conference on Artificial Intelligence (SETN'10), Lecture Notes in Artificial Intelligence, Vol. 6040, pp. 153-162, 4-7 May 2010, Athens, Greece.

C2 V. Karavasilis, K. Blekas and C. Nikou. Motion segmentation by model-based clustering of incomplete trajectories. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML - PKDD'11). Lecture Notes in Artificial Intelligence, Vol. 2, pp.146-161, 5-9 September 2011, Athens, Greece.

C3 V. Karavasilis, C. Nikou and A. Likas. Gaussian Mixture-based Mean Shift for Tracking under Abrupt Illumination Changes. Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP'12), pp. 510-513, 18-20 July 2012, Athens, Greece.

C4 V. Karavasilis, C. Nikou and A. Likas. Visual tracking by weighted likelihood maximization. 24th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'12), pp. 246-252, 7-9 November 2012, Athens, Greece.

Publications not related with this thesis

1. M. Vrigkas, V. Karavasilis, C. Nikou and I. A. Kakadiaris, Matching mixtures of curves for human action recognition, Computer Vision and Image Understanding, Vol. 119C, pp. 27-40, February 2014.

2. M. Vrigkas, V. Karavasilis, C. Nikou and I. Kakadiaris, Action recognition by matching clustered trajectories of motion vectors, in Proc. 8th International Conference on Computer Vision Theory and Applications, pp. 112-117, Barcelona, Spain, 21-24 February 2013.

SHORT VITA

Vasileios Karavasilis received the Diploma and the MSc in Computer Science from the University of Ioannina, Greece, in 2007 and 2009, respectively. Since 2009, he has been a PhD candidate and a researcher in the Department of Computer Science and Engineering of the University of Ioannina. His main research interests lie in the fields of computer vision, image processing and machine learning. He is a member of IEEE.