Computer Vision and Image Understanding, Vol. 140, pp. 43-57, 2015

# Visual tracking using spatially weighted likelihood of Gaussian mixtures

Vasileios Karavasilis[1], Christophoros Nikou, Aristidis Likas
*Department of Computer Science and Engineering, University of Ioannina, Ioannina, Greece*

**Abstract**

A probabilistic real time tracking algorithm is proposed where the target's feature distribution is represented by a Gaussian mixture model (GMM). The target localization is achieved by maximizing its weighted likelihood in the image sequence. 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 real time performance is experimentally confirmed, while the algorithms has comparative performance with other state-of-the-art tracking algorithms.

*Keywords:* Visual tracking, kernel-based tracking, target representation, target localization, Gaussian mixture model (GMM), Expectation-Maximization (EM), weighted likelihood.

## 1. Introduction

The application of spatial kernels in visual tracking algorithms was proposed in [1]. Masking the object with a kernel allows for gradient-based optimization instead of a brute force search for target localization and real-time performance may be achieved on a standard personal computer. The shape of the target is approximated by an ellipse and its color distribution is modeled by a histogram. Combining the ellipse with a spa-

*Email address:* {vkaravas,cnikou,arly}@cs.uoi.gr (Aristidis Likas)

[1]V. Karavasilis is funded by the Greek State Scholarships Foundation.

tial kernel eliminates the effect of varying object dimensions (e.g. a long thin object) and allows tracking of a wide variety of targets. Using a histogram instead of a continuous distribution of the color allows the acceleration of the optimization procedure. However, if the feature dimension increases the histogram bins increase exponentially. It is also assumed that the color distribution of the object does not change significantly along the image sequence, which does not hold in many scenarios.

In this work, 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 Gaussian mixture model (GMM) instead of a histogram or histogram signatures, as it is the case in [1, 2, 3]. 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 [1, 4, 2, 5], 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.

A preliminary version of this work was presented in [6]. Herein, we present a more detailed theoretical description of the model, we incorporate the scale adaptation procedure, an approach to handle rotations of the target, we propose a principled update procedure of the GMM and we show the relation of the tracking algorithm to mean-shift. We have also made a more extended evaluation of the proposed method by including more experimental datasets and comparison measures.

In he remaining of the paper, section 2 reviews the related literature on visual tracking, the tracking algorithm relying on the maximization of the weighted target likelihood is described in section 3, experimental results are presented in section 4 and conclusions are drawn is section 6.

## 2. Related work

A large number of methods have been proposed for visual tracking, which rely on template matching [7, 8], small patch tracking [9, 10], particle filters [11, 12, 13, 14], sparse representations [15, 16], contour modeling [17] or image segmentation [18]. A detailed review and analysis may be found in [19, 20, 21]. In the following section, we survey recent tracking algorithms based on mean shift and Gaussian mixture models, which are the core of the method proposed herein and we also position our work with respect to these methods.

### *2.1. Mean shift tracking*

The key idea of the mean shift algorithm [1, 5] is the representation of the target by an ellipse. 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. This idea is the key to enable an explicit optimization of a cost function, which yields an estimation of the most likely position of the target. In order to increase its accuracy, the mean shift algorithm has been combined with other methods. In [22], the proposed tracking algorithm is an integration of mean shift and SIFT feature tracking. 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. The authors in [23] investigate the advantages of using a more detailed shape model instead of a generic ellipse for target representation.

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. 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 [2] minimizes the earth mover's distance (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. Similar in spirit is the algorithm proposed in [24] which minimizes the EMD between Gaussian mixture models. These two works are not explicit mean shift trackers but belong to the broader category of kernel-based trackers. In [25], two trackers that employ cross-bin metrics and are based on mean shift iterations are proposed. The work in [26], 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 [27], the target appearance is modeled using a sparse coding histogram based on a learned 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.

Another case where mean shift fails is when the object's motion is abrupt and the target ellipses in two consecutive frames do ton intersect, which results from the local optimization performed in the framework of kernel-based trackers. The work in [28] addresses this drawback by employing a pyramidal decomposition to capture distant targets between consecutive frames. An extension of the main algorithm is proposed in [29], 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.

In this work, we propose a kernel-based tracking algorithm with a mean shift-like closed form update for the target location which mainly addresses the problem of illumination change in the standard mean shift algorithm. At first, the histogram of the target is captured by a weighted Gaussian mixture model instead of a larger number of bins, which makes makes the method more robust to illumination changes which result to histogram drifting. Furthermore, the likelihood of the proposed weighted Gaussian mixture model is directly maximized which is in contrast to the maximization of an approximation of the Bhattacharrya coefficient optimized in the standard mean shift algorithm.

## 2.2. *Tracking using Gaussian mixture models*

Gaussian mixtures have been widely used in computer vision for image segmentation [30], background subtraction [31, 32], image classification [33] and human pose estimation [34]. 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 [35], 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 [12, 14] 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 [37], 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.

The methods above do not take into account any prior knowledge or a confidence that some pixels in the candidate target may be more important than others, which may integrated by explicitly weighing the pixels of the candidate target. In this work, we model the target by a weighted GMM whose parameters are estimated only once, at the first frame. In the subsequent frames, only the likelihood of the pixels with respect to the initial GMM are evaluated and the motion of the target is obtained in closed form in a mean shift like formula.

## 3. 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 $k(\cdot)$:

$$w_n(\boldsymbol{y}) = k\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right), \tag{1}$$

where

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

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 (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:

$$L_n = \ln \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k), \tag{3}$$

is described by a GMM of $K$ components with mixing proportions $\pi_k$ 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 $\boldsymbol{y}$:

$$L(\boldsymbol{I}, \boldsymbol{w}(\boldsymbol{y}); \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{n=1}^{N} w_n(\boldsymbol{y}) L_n, \tag{4}$$

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 [38] 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 $\boldsymbol{z}_n = [z_{n,1}, z_{n,2}, , z_{n,K}]$ 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:

$$\ln p(\boldsymbol{I}, \boldsymbol{w}(\boldsymbol{y}), \boldsymbol{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \tag{5}$$

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:

$$\begin{aligned} \mathcal{L} &= p(\boldsymbol{z}; \boldsymbol{I}, \boldsymbol{w}(\boldsymbol{y}), \boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) \\ &\propto \prod_{n=1}^{N} \prod_{k=1}^{K} [\pi_k \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]^{z_{n,k} w_n(\boldsymbol{y})}, \end{aligned} \tag{6}$$

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:

$$Q = \sum_{n=1}^{N} w_n(\boldsymbol{y}) \sum_{k=1}^{K} r(z_{n,k}) [\ln \pi_k + \ln \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)]. \tag{7}$$

The EM algorithm can now be employed in order to maximize the weighted log-likelihood (4) with respect to $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{\pi}$.

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

$$E[z_{n,k}] = r(z_{n,k}) = w_n(\boldsymbol{y}) \frac{\pi_k \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{l=1}^{K} \pi_l \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l)}. \tag{8}$$

In the M-Step, the complete-data log likelihood (7) is maximized with respect to the parameters $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$, $\boldsymbol{\pi}$ leading to the following updates:

$$N_k = \sum_{n=1}^{N} r(z_{n,k}), \tag{9}$$

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} r(z_{n,k}) \boldsymbol{I}_n, \tag{10}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k} \sum_{n=1}^{N} r(z_{n,k}) \left(\boldsymbol{I}_n - \boldsymbol{\mu}_k\right) \left(\boldsymbol{I}_n - \boldsymbol{\mu}_k\right)^T, \tag{11}$$

$$\pi_k = \frac{N_k}{\sum_{n=1}^{N} w_n(\boldsymbol{y})}. \tag{12}$$

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 $\pi_k$ 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.

## *3.1. Gradient based update*

In order to estimate the position of the object in the next frame, the gradient of the weighted likelihood (4) with respect to $\boldsymbol{y}$ must be computed:

$$\begin{aligned}\frac{dL}{d\boldsymbol{y}} &= \frac{dL(\boldsymbol{I}, \boldsymbol{w}(\boldsymbol{y}); \boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma})}{d\boldsymbol{y}} \\ &= \sum_{n=1}^{N} \frac{dk\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right)}{d\boldsymbol{y}} L_n,\end{aligned} \tag{13}$$

where $L_n$ is the log-likelihood for the $n$-th pixel defined in (3) and

$$\frac{dk\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right)}{d\boldsymbol{y}} = \begin{bmatrix} \frac{dk(f(\boldsymbol{x}_n;\boldsymbol{y},\boldsymbol{h}))}{dy^{(1)}} \\ \frac{dk(f(\boldsymbol{x}_n;\boldsymbol{y},\boldsymbol{h}))}{dy^{(2)}} \end{bmatrix}. \tag{14}$$

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

$$\frac{dk\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right)}{d\boldsymbol{y}} = 2\boldsymbol{A}_n(\boldsymbol{y}) g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right), \tag{15}$$

where

$$\boldsymbol{A}_n(\boldsymbol{y}) = \left[\frac{x_n^{(1)} - y^{(1)}}{h^{(1)^2}}, \frac{x_n^{(2)} - y^{(2)}}{h^{(2)^2}}\right]^T, \tag{16}$$

leading to:

$$\frac{dL}{d\boldsymbol{y}} = \sum_{n=1}^{N} 2\boldsymbol{A}_n(\boldsymbol{y}) g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right) L_n. \tag{17}$$

Once (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 [2], 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 (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 (4) is that the gradient in (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.

## *3.2. 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 (4) by setting its derivative (17) with respect to $\boldsymbol{y}$ equal to zero, thus obtaining:

$$\boldsymbol{y} = \frac{\sum_{n=1}^{N} \boldsymbol{x}_n g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right) L_n}{\sum_{n=1}^{N} g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}, \boldsymbol{h}\right)\right) L_n}, \tag{18}$$

which is a mean shift like update [1]. In (18), the log-likelihood $L_n$ for the $n$-th pixel, which is obtained from (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

$$L'_n = \ln\left(B \times \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right) = \ln B + L_n \tag{19}$$

where $B$ is a normalization factor such that

$$B \times \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{I}_n; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \geq 1, \; \forall\, n \in \{1, \ldots, N\}, \tag{20}$$

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 (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 (19) is the sum $L'_n = \ln(B) + L_n$ and an update equation like (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:

$$\boldsymbol{y}_{new} = \frac{\sum_{n=1}^{N} \boldsymbol{x}_n g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}_{old}, \boldsymbol{h}\right)\right) L'_n}{\sum_{n=1}^{N} g\left(f\left(\boldsymbol{x}_n; \boldsymbol{y}_{old}, \boldsymbol{h}\right)\right) L'_n}. \tag{21}$$

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.

## *3.3. Scale adaptation*

In order to scale the target, we could use the derivative of the weighted log-likelihood (4) with respect to the components of $\boldsymbol{h}$. For $h^{(1)}$, this would result to:

$$\frac{dL}{dh^{(1)}} = \sum_{n=1}^{N} 2g(f(x_n; \boldsymbol{y}, \boldsymbol{h})) \frac{(x_n^{(1)} - y_n^{(1)})^2}{(h^{(1)})^3} L_n. \tag{22}$$

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 (4) we had to shrink the ellipse every time, until it reaches 1 pixel.

One commonly used technique [1, 2, 24] is to scale up and down the ellipse which represents the target by a scale factor and keep the scale which maximizes (4). However, due to the fact that the log-likelihood function (4) depends on the number of pixels $N$, we can not use (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 (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. 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. 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 (2) are:

$$\begin{aligned}\left(\frac{P'^{(1)} - y^{(1)}}{(1+\alpha) * h^{(1)}}\right)^2 &= \left(\frac{(1+\alpha) * \left(P^{(1)} - y^{(1)}\right)}{(1+\alpha) * h^{(1)}}\right)^2 \\ &= \left(\frac{P^{(1)} - y^{(1)}}{h^{(1)}}\right)^2,\end{aligned} \tag{23}$$

while the second terms in (2) are equal because there is no scale in the vertical direction. Furthermore, smoothing by a $5 \times 5$ 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 (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. 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 $2\alpha, 3\alpha, \dots$ until the log-likelihood is decreased or a maximum scale is reached. A similar approach is used

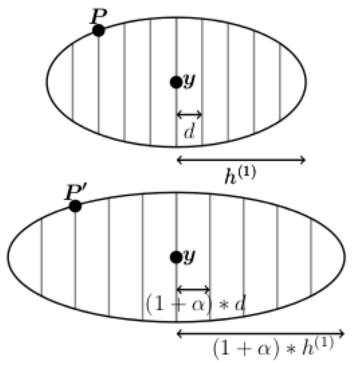


Figure 1: The original ellipse (top) and the horizontally scaled ellipse (bottom). The pixels that are used in (4) are represented by the gray columns. When the size of the ellipse increases by $\alpha\%$, the inter-column distance is also increased by the same amount. Thus, the number of pixels $N$ is constant and $f(\boldsymbol{P}; \boldsymbol{y}, \boldsymbol{h}) = f(\boldsymbol{P}'; \boldsymbol{y}, \boldsymbol{h})$.

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 $\alpha$ results to bigger scale changes, but results to more coarse estimation of the scale) and the computational speed (smaller $\alpha$ 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 [5].

### 3.4. 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 $\pi_k$ 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 $\pi_k$ 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 our work 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 $\pi_k$ 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 [9] for tracking individual points and in [39] for scale estimation. 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 $\boldsymbol{y}'_T, \boldsymbol{y}'_{T-1}, \dots, \boldsymbol{y}'_{T-M}$ (the same applies to $\boldsymbol{h}'_t$). The average Euclidian distance between the centers and axis, respectively, is defined as:

$$Euc_y(\boldsymbol{y}, \boldsymbol{y}') = \frac{1}{M} \sum_{t=0}^{M} \sqrt{\sum_{j=1}^{2} \left( \frac{\boldsymbol{y}_{T-t}^{(j)} - \boldsymbol{y}_{T-t}^{\prime(j)}}{\frac{\boldsymbol{h}_{T-t}^{(j)} + \boldsymbol{h}_{T-t}^{\prime(j)}}{2}} \right)^2}, \tag{24}$$

$$Euc_h(\boldsymbol{h}, \boldsymbol{h}') = \frac{1}{M} \sum_{t=0}^{M} \sqrt{\sum_{j=1}^{2} \left( \frac{\boldsymbol{h}_{T-t}^{(j)} - \boldsymbol{h}_{T-t}^{\prime(j)}}{\boldsymbol{h}_{T-t}^{(j)}} \right)^2}, \tag{25}$$

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 $\pi_k$. 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. 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 $[-45°, +45°]$ at each iteration and selects the angle providing the maximum value of the log-likelihood (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 1. The update of the GMM parameters is described in Algorithm 2.

## 4. Experimental results

The evaluation of the proposed tracking algorithm was performed using nine real datasets (Fig. 5). We used two variations of the proposed method, one based on the derivative (17), which will be referred as WLT, and one based on the mean shift-like formula (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

**Algorithm 1** *WLT* algorithm

```
 1: function WLT(Image sequence, $M$)
 2:     Input: an image sequence consisting of $T$ frames and the frequency $M$ of
    updating the target model.
 3:     Output: the ellipse center $\boldsymbol{y}$ at each frame.
 4:     Initialization:
 5:     Determine the initial position $\boldsymbol{y}_1$ and the size $\boldsymbol{h}_1$ of the target.
 6:     Compute the parameters $\pi_k$, $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$ of the GMM describing the target
    using (10), (11) and (12).
 7:     Tracking:
 8:     for frame $t = 2, \ldots, T$ do
 9:         $\boldsymbol{y}_t := \boldsymbol{y}_{t-1}$
10:         $\boldsymbol{h}_t := \boldsymbol{h}_{t-1}$
11:         while the likelihood in (4) increases do
12:             Move to $\boldsymbol{y}_t$ using (17).
13:         end while
14:         Estimate horizontal and vertical sizes of the target $\boldsymbol{h}_t = [h_t^{(1)}, h_t^{(2)}]^T$.
15:         Estimate the rotation $R$ of the target.
16:         if mod$(t, M) == 0$ then
17:             Update the target model using Algorithm 2.
18:         end if
19:     end for
20: end function
```

**Algorithm 2** 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 $\pi_k$ 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
```

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 [2]. The first criterion is the number of frames which the object is correctly 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:

$$p = \frac{1}{T}\sum_{i=1}^{T} p_i, \tag{26}$$

where

$$p_i = \frac{\text{number of correctly tracked pixels in frame } i}{\text{number of tracked pixels in frame } i}, \tag{27}$$

the average recall:

$$r = \frac{1}{T}\sum_{i=1}^{T} r_i, \tag{28}$$

where

$$r_i = \frac{\text{number of correctly tracked pixels in frame } i}{\text{number of target pixels in frame } i}, \tag{29}$$

and the average F-measure:

$$F = \frac{1}{T}\sum_{i=1}^{T}\frac{p_i \times r_i}{p_i + r_i}. \tag{30}$$

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

$$k(x) = \begin{cases} e^{(-x/\sigma)} & \text{if} \quad x \leq 1 \\ 0 & \text{otherwise} \end{cases} \tag{31}$$

Consecutively, the derivative of (17) becomes:

$$\frac{dL}{d\boldsymbol{y}} = \sum_{n=1}^{N} \boldsymbol{A}_n(\boldsymbol{y}) w_n(\boldsymbol{y}) L_n. \tag{32}$$

We compared our method with the OpenCV's implementation of Camshift algorithm [5, 40] which is a robust version of the mean shift algorithm [1] with scale adaptation and the FRAG tracker [41]. 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 [5]. 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 1-6 and Figures 2-4, 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. 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 (21), the new center may be significantly further with respect to the current center, and may not provide the maximum of the log-likelihood (4). On the other hand, WLT uses one pixel displacements in every iteration, after evaluating (17), and results to smoother position changes and estimated final locations which minimize the log-likelihood (4).

Also, we evaluated the performance of the algorithm when the target rotates. In Fig. 9, representative frames of the image sequence that is used for testing are shown. The

Table 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 |

Table 2: Performance of camshift, FRAG, WLT and WLTMS in terms of position error (mean±std).

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

Table 3: Performance of camshift, FRAG, WLT and WLTMS in terms of size error (mean±std).

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

Table 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: 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 |

Table 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 |

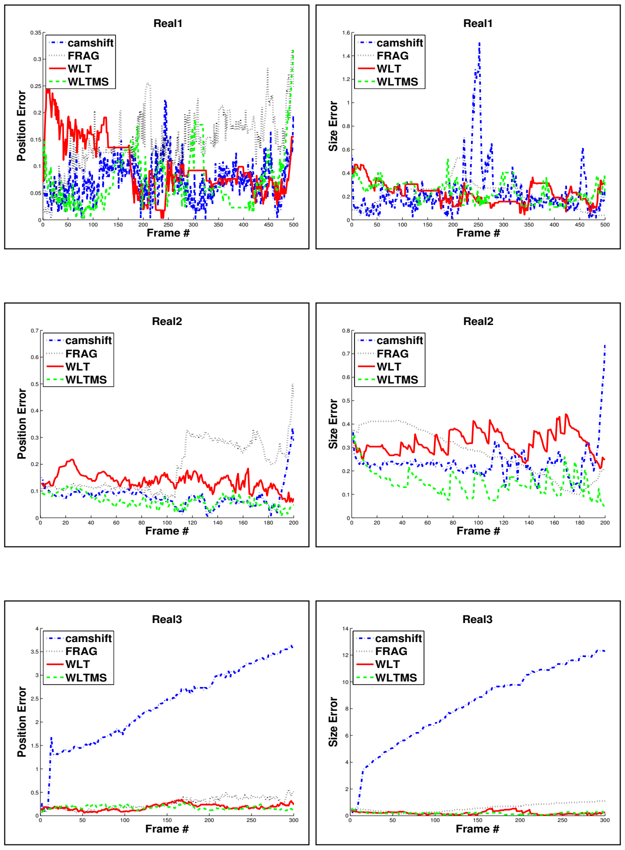


Figure 2: Performance of camshift, FRAG, WLT and WLTMS in terms of position and size error.

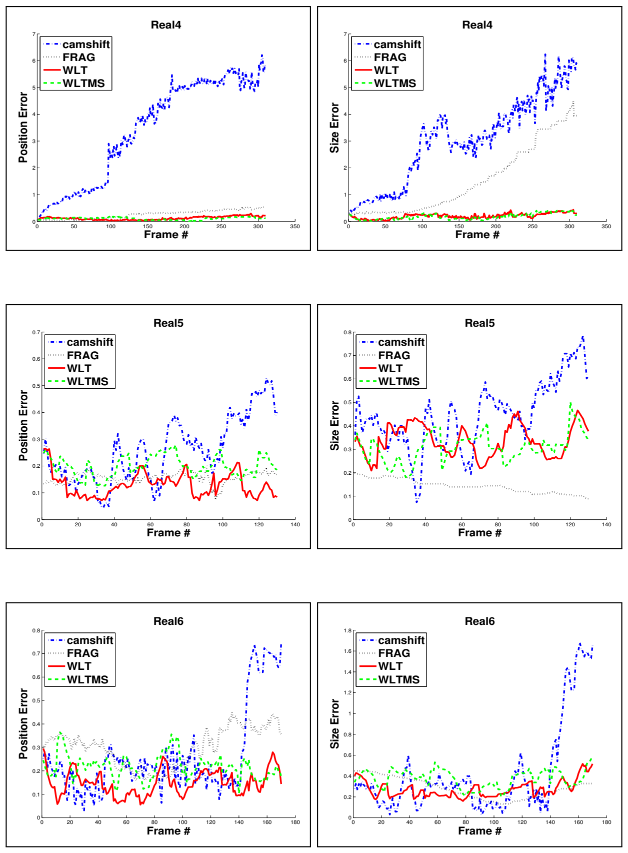


Figure 3: Performance of camshift, FRAG, WLT and WLTMS in terms of position and size error.

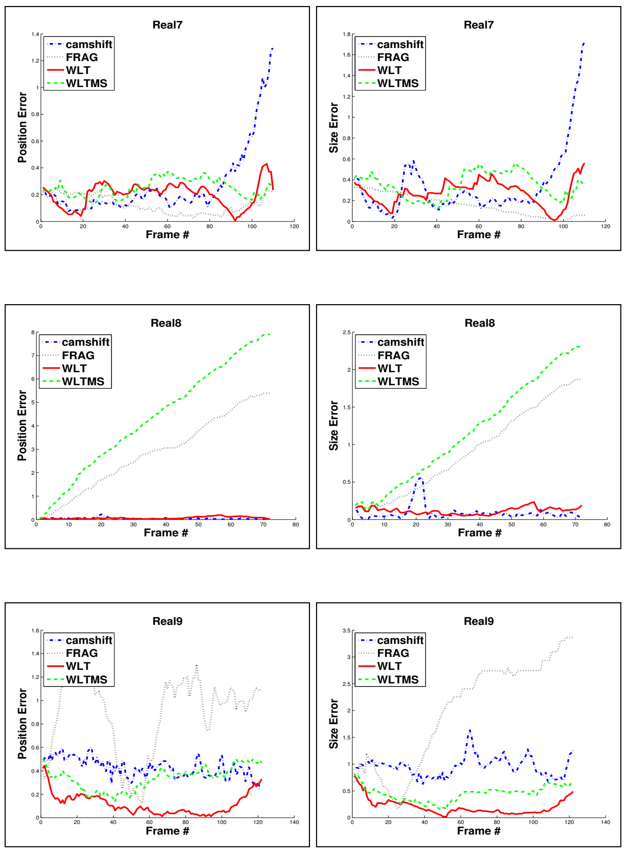


Figure 4: Performance of camshift, FRAG, WLT and WLTMS in terms of position and size error.

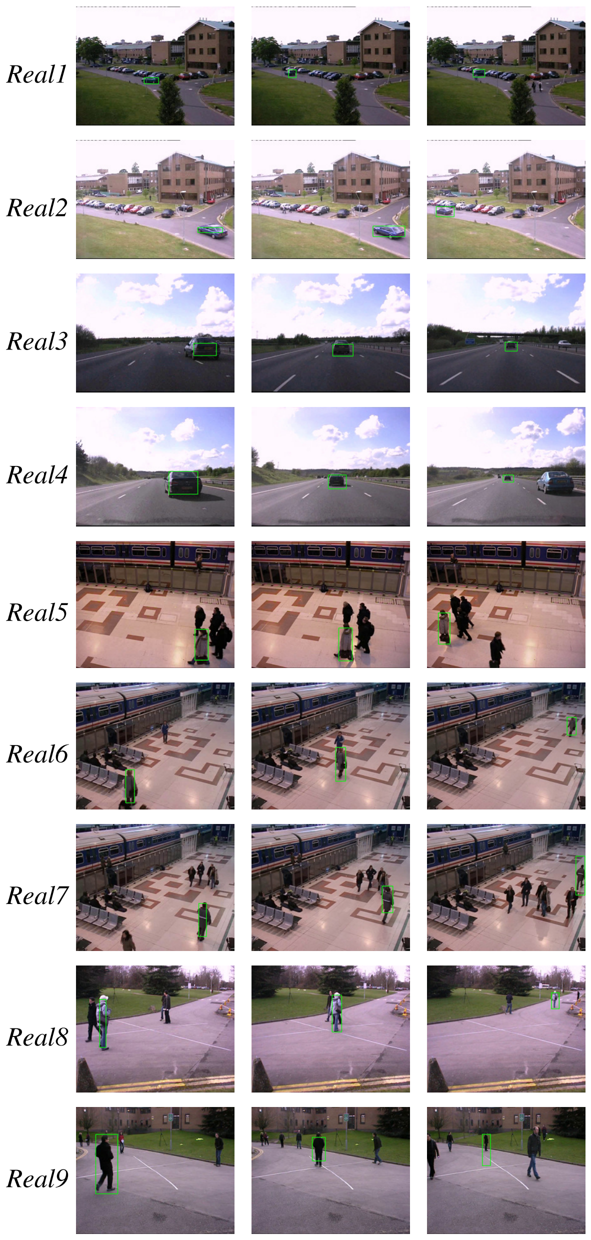


Figure 5: Representative results on the real datasets used in the experiments *Real1*, *Real2*, *Real3* and *Real4*, *Real5*, *Real6*, *Real7*, *Real8* and *Real9* using WLT. Although the inscribed ellipse is used in the computations, the target is bounded by a green rectangle for visualization purposes.

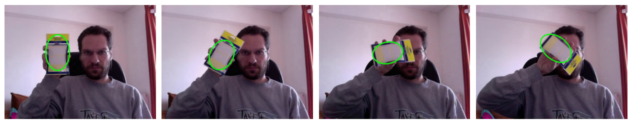


Figure 6: Representative frames of the sequence used for the evaluation of the algorithm on rotations of the target.

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 $1.86$.

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 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 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

Table 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* | $74 \times 32$ | $48 \times 22$ | 2.242 |
| *Real2* | $130 \times 27$ | $128 \times 25$ | 1.096 |
| *Real3* | $112 \times 62$ | $114 \times 60$ | 1.015 |
| *Real4* | $211 \times 162$ | $146 \times 128$ | 1.837 |
| *Real5* | $60 \times 170$ | $50 \times 130$ | 1.569 |
| *Real6* | $50 \times 162$ | $50 \times 154$ | 1.051 |
| *Real7* | $42 \times 150$ | $36 \times 150$ | 1.166 |
| *Real8* | $30 \times 250$ | $29 \times 218$ | 1.186 |
| *Real9* | $100 \times 400$ | $82 \times 400$ | 1.219 |

these results in Table 7.

In these image sequences, the rectangles which represent the targets have dimensions around $150 \times 70$ 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.

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. 7). We check for an update according to Algorithm 2 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

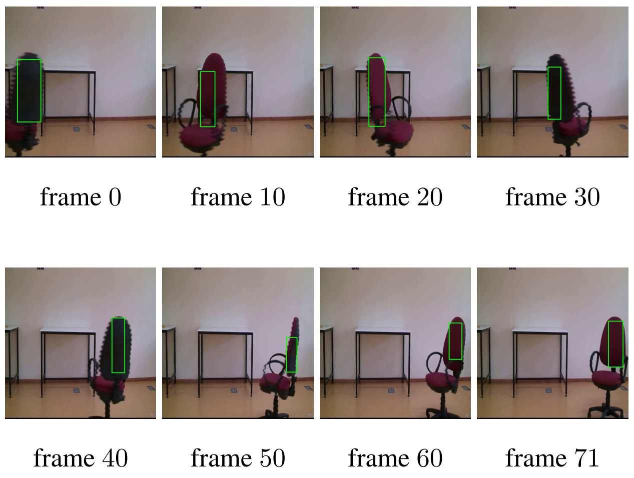


Figure 7: Representative frames for the sequence that is used for the qualitative evaluation of the model update (the total number of frames that were used during the tracking procedure is 71). The chair rotates around its axis and moves from left to right. The model update procedure is applied every 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.

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. 8, quantitative results are presented for the the position and size 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.

## 5. 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 [42]. VOT provides the toolset in order to evalu-

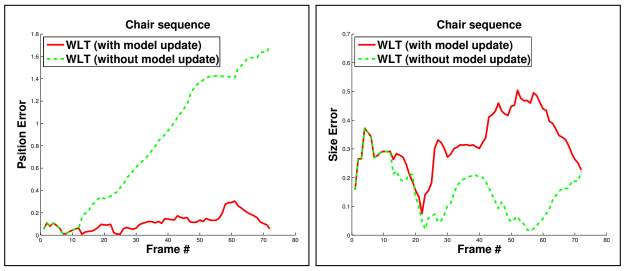


Figure 8: Performance of WLT without model update (green) and with model update (red).

ate 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 [42].

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

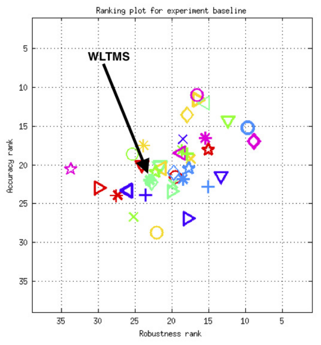
$$acc = \frac{A^G \cap A^T}{A^G \cup A^T}. \tag{33}$$

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.

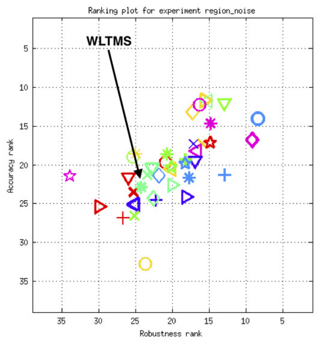
The evaluation procedure is as follows: the tracker runs on each sequence at most 15 times (3 times if it is deterministic, 15 if not deterministic) and the average accuracy and robustness of the sequences is indicated. Moreover, two set of experiments are performed: (i) the baseline experiments, in 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 $\pm 0.1$ radians.

The performance of our WLTMS method is summarized in Table 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 [42]. Moreover, as it is stated in [42], 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 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 $20 \times 30$ 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



(a)



(b)

Figure 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.

of particle filtering is employed.

A graphical representation of the performance of the proposed method with respect to the state of the art is shown in Fig.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.

## 6. Conclusion

From the point of view of the target modeling and localization, the proposed algorithm belongs to the same family as the histogram based methods [1, 5, 2, 24, 40]. 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 [1, 5] minimizes the Bhattacharyya distance while in [2, 24] 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

Table 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) |

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.

The method, in its current form addresses the problem of single object tracking in real time. A perspective of this work is to integrate it into more sophisticated schemes including data association methods, for multiple object tracking.

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

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

[3] Z. Zivkovic, B. Krose, An EM-like algorithm for color-histogram-based object tracking, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'04), Vol. 1, 2004, pp. 798–803.

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

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

[6] V. Karavasilis, C. Nikou, A. Likas, Visual tracking by weighted likelihood maximization, in: 24th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'12), 7-9 November 2012, Athens, Greece, pp. 246–252.

[7] K. Zimmermann, J. Matas, T. Svoboda, Tracking by an optimal sequence of lin-

ear predictors, IEEE Transactions on Pattern Analysis and Machine Intelligence 31 (4) (2009) 677–692.

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

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

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

[11] G. Tzimiropoulos, S. Zafeiriou, M. Pantic, Sparse representations of image gradient orientations for visual recognition and tracking, in: Proceedings of IEEE Intl Conf. Computer Vision and Pattern Recognition (CVPR-W11), Workshop on CVPR for Human Behaviour Analysis, 2011, pp. 26–33.

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

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

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

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

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

[17] E. Horbert, K. Rematas, 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), 2011, pp. 1–8.

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

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

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

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

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

[23] Z. Wang, M. B. Salah, H. Salah, N. Ra, Shape based appearance model for kernel tracking, Image and Vision Computing 30 (45) (2012) 332–344.

[24] V. Karavasilis, C. Nikou, A. Likas, Visual tracking using the earth mover's distance between Gaussian mixtures and Kalman filtering, Image and Vision Computing 29 (5) (2011) 295–305.

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

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

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

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

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

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

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

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

[33] R. P. Browne, P. D. McNicholas, 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) (2012) 814–817.

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

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

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

[37] J. Kim, Z. Lin, I. S. Kweon, Rao-blackwellized particle filtering with gaussian mixture models for robust visual tracking, Computer Vision and Image Understanding 125 (0) (2014) 128–137.

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

[39] T. Vojir, J. Noskova, J. Matas, Robust scale-adaptive mean-shift for tracking, in: Proc. of the 18th Scandinavian Conf. on Image Analysis (SCIA), 2013, pp. 652–663.

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

[41] A. Adam, E. Rivlin, I. Shimshoni, Robust fragments-based tracking using the integral histogram, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR'06), Vol. 1, 2006, pp. 798–805.

[42] 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, Z. Niu, The visual object tracking vot2014 challenge results, in: Proceedings of the European Conference on Computer Vision (ECCV) Visual Object Tracking Challenge Workshop, Zurich, Switzerland, 2014, pp. 98–111.