Kernel based Object Tracking Using Metric Distance Transform and RVM Classifier

1Saeed Rastegar, 2Mojtaba Bandarabadi, 3Yashar Toopchi and 4Saleh Ghoreishi

1,2Department of electrical and Computer engineering, Babol Noshirvani University P.O. 484, Babol, Iran.
4Science and research branch, Islamic Azad University, Tehran, Iran

Abstract: In this work, we aim to remedy one of its shortcomings Mean shift (MS) algorithm, which like other gradient ascent optimization methods, is designed to find local modes. An efficient MS procedure that uses the flexible kernels based on the normalized Metric Distance Transform is proposed and tested. The target shape which defines the da Distance Transform is found based on wavelet transform as feature vector extraction and Relevance Vector Machine (RVM) classifier to differentiate tracked target from background. This replaces the more usual Epanechnikov kernel (E-kernel), improving target representation and localization without increasing the processing time, minimizing the similarity measure using the Bhattacharya coefficient. The algorithm is tested on several image sequences and shown to achieve robust and reliable frame-rate tracking.

Key words: Object tracking, da Distance Transform, Mean Shift, wavelet transform, Relevance Vector Machine (RVM), Bhattacharyya coefficient, Epanechnikov kernel.

INTRODUCTION

Real-time object tracking is the critical task in many Computer vision applications such as surveillance, perceptual user interfaces, augmented reality, smart rooms, object-based video compression, and driver assistance.

Two major components can be distinguished in a typical Visual tracker. Target Representation and Localization is mostly a bottom-up process which has also to cope with the changes in the appearance of the target. Filtering and Data Association is mostly a top-down process dealing with the dynamics of the tracked object, learning of scene priors, and evaluation of different hypotheses. The way the two components are combined and weighted is application dependent and plays a decisive role in the robustness and efficiency of the tracker. For example, face tracking in a crowded scene relies more on target representation than on target dynamics, while in aerial video surveillance, the target motion and the ego-motion of the camera are the more important components. Therefore, it is desirable to ensure that the tracker is as efficient as possible. Kernel-based density estimation techniques for Computer vision has attracted a great deal of attention. One example is the mean shift (MS) technique which has been applied to image segmentation, visual tracking, etc. (Georgescu, B., I. Shimshoni, 2003) –(Modify Kernel Tracking Using an Efficient Color Model and Active, 2009) . MS is a versatile nonparametric density analysis tool introduced in (Fukunaga, K. and L.D. Hostetler, 1975) –(Comaniciu, D. and P. Meer, 2002) . In essence, it is an iterative mode detection algorithm in the density distribution space.

The MS algorithm moves to a kernel-weighted average of the observations within a smoothing window. This computation is repeated until convergence is attained at a local density mode. This way the density modes can be elegantly located without explicitly estimating the density. Cheng (1995) notes that MS is fundamentally a gradient ascent algorithm with an adaptive step size. Fashing and Tomasi (Fashing, M. and C. Tomasi, 2005) show the connection between MS and the Newton–Raphson algorithm. They also discover that MS is actually a quadratic bound optimization both for stationary and evolving sample sets. MS is also a fixed -point iteration procedure. Since Comaniciu et al. (2003) first introduced MS based object tracking, it has proven to be a promising alternative to popular particle altering based trackers(Zhou, S., R. Chellappa and B. Moghaddam, 2003) . A number of improvements to the method have been reported in the literature. In (Collins, R., 2003), the selection of kernel scale via linear search is discussed. El-gammal et al. (2007)
(2003) reformulates the tracking framework as a general form of joint feature-spatial distributions (Modify Kernel Tracking Using an Efficient Color Model and Active, 2009). Compared with the approach of Comaniciu et al., the advantage is that spatial structure information of the tracked region is incorporated. In (Kernel-Based Object Tracking With Multiple Features-Amir Babaeian, 2009), multiple spatially distributed kernels are adopted to accurately capture changes in the target’s orientation and scale. Another approach is developed in (Zivkovic, Z. and B. Kroese, 2004) for the same purpose. Furthermore, Fan et al. (2005) present a theoretical analysis of the similarity measure and arrive at a criterion, leading to kernel design strategies with prevention of singularity in kernel visual tracking. All the above mentioned trackers adopt MS or similar optimization strategies. Despite successful applications, MS trackers require that the displacement of the tracked target in consecutive frames to be small because the search is initialized by the detected location of the target in the previous frame. Larger inter-frame displacements will lead the tracker to become trapped in spurious locations in the multimodal density distribution space because MS is a local optimization method. So in many situations, however, we seek the global mode of a density function. The standard MS tracker assumes that the initialization point falls within the basin of attraction of the desired mode. On the other hand, in basic MS algorithm, the feature histogram-based target representations are regularized by spatial masking with a descent isotropic kernel. In most instances, the target does not have radial symmetry, so the use of a E-kernel includes foreground as background, or background as foreground pixels, or both. So non-included target pixels are weighted. Hence, the local and global modes may not correspond and the tracker is likely to fail. The main contribution of the paper is to introduce a new framework for efficient tracking of nonrigid objects. We show that such a kernel-based object tracking algorithm can be improved by using a kernel based on the Distance Transform. Wavelet transform as powerful tool is used to obtain feature vector on the target in first frame to train Relevance Vector Machine (RVM). The central of target localization in each frame is estimated by MS and then RVM is applied to differentiate the target from background. Weighting of segmented target pixels is done by metric Distance Transform which is used to estimate of target localization for next frame. We present experiments that demonstrate the superior performance of this approach in comparison with the basic MS algorithm or using self-RVM classifier (2008) by measuring accuracy, robustness and stability. The remainder of the paper is organized as follows. Section 2 explains the kernel density estimating and meanshift analysis. 3 describe the wavelet transform which is used as extraction of feature space for target and candidates model. Relevance Vector Machine classifier is explained in Section 4. Section 5 introduces metric distance transforms. Section 6 explains the experimental settings and the experimental results will be reported. Finally, a conclusion and discussion are presented in Section 7.

**Kernel Density Estimation:**

Motion of the target in two sequential single images describe with good approximation using mean shift algorithm Comaniciu et al. (2003). At the first stage, we make a model of the target using weighing Histogram, which exploits both color and location data. The region of target is supposed as elliptic in the first frame then points inside of elliptic are normalized to unit circle with 0 center. Marginal pixels are less important than central pixels because they are influenced by occlusion and interference with background. In order to consider this importance, we allocate a weight to each pixel based upon distance of that pixel from center. Therefore weighing Histogram is obtained as below:

\[
\hat{q}_u = C \sum_{i=1}^{X} k \left( \frac{\| x_i^u \|^2 }{ \sigma^2 } \right) \delta \left[ b \left( x_i^u \right) - \hat{u} \right]
\]

(1)

Where \( b \) function relates corresponding index to each pixel. \( K \) is a convex, uniformly decreasing and isotropic function. \( C \) constant is defined in order to provide \( \sum_{u=1}^{m} \hat{q}_u = 1 \). We search an elliptical region in new frame on which is most similar to targeted one. We search our target in new frame around the target location in previous frame and estimate most similarities as new location of the target because target motion in two sequential frames is not considerable. This exhaustive search is time consuming, so utilizing of proper recursive algorithm is recommended. We should consider another parameter other than target location.

This parameter is object scale \( (h) \) which modifies elliptic size in each frame. We define similarity criteria
for comparing two Histograms. In this paper we use Bhattacharya distance which defined below:

$$\bar{\rho}(y) = \rho[\rho(y), \bar{q}] = \sum_{u=1}^{m} \sqrt{\bar{\rho}_u(y) \bar{q}_u}$$  \hspace{1cm} (2)

If in previous frame target location $\bar{y}_0$ and target scale $\bar{h}$ are, in current frame similarity of Histogram around $\bar{y}_0$ will be calculated through Taylor expansion:

$$\rho[\rho(y), \bar{q}] \approx \frac{1}{2} \sum_{u=1}^{m} \sqrt{\bar{\rho}_u(y_0) \bar{q}_u} + \frac{1}{2} \sum_{u=1}^{m} \bar{\rho}_u(y) \sqrt{\frac{q_u}{\bar{\rho}_u(y_0)}}$$ \hspace{1cm} (3)

Let $\{x_i\}_{i=1}^{n}$ be the normalized pixel locations in the current frame. Using the same kernel profile $k(x)$, but with bandwidth $h$, the probability of the future $u=1 \ldots m$ in the target candidate is given by

$$\bar{\rho}_u(y) = C_h \sum_{i=1}^{n} k\left(\frac{y-x_i}{h}\right) \delta\left[ b\left(x_i\right) - u \right]$$ \hspace{1cm} (4)

Recalling (4) result in

$$\rho[\rho(y), \bar{q}] \approx \frac{1}{2} \sum_{u=1}^{m} \sqrt{\bar{\rho}_u(y_0) \bar{q}_u} + \frac{1}{2} \sum_{u=1}^{m} \bar{\rho}_u(y) \sqrt{\frac{q_u}{\bar{\rho}_u(y_0)}}$$ \hspace{1cm} (5)

Where

$$w_i = \sum_{u=1}^{m} \bar{\rho}_u(y) \sqrt{\frac{q_u}{\bar{\rho}_u(y_0)}} \delta\left[ b\left(x_i\right) - u \right]$$ \hspace{1cm} (6)

We should maximize second term in (5) in order to have maximum similarity.

We consider this term as Kernel estimation $k(x)$ of a probability density function and also use mean shift algorithm for calculating maximum of this equation. In this algorithm current location moved to the new one as below:

$$\bar{y}_1 = \frac{\sum_{i=1}^{n} w_i \eta \left( \frac{y_0-x_i}{h} \right)}{\sum_{i=1}^{n} w_i \eta \left( \frac{y_0-x_i}{h} \right)^2}$$ \hspace{1cm} (7)

2-1 Background-Weighted Histogram:

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 cross scales, hence, to determine the appropriate target scale. A good approach is to derive a simple representation of the background features and to use it for selecting only the salient parts from the representations of the target model and target candidates Comaniciu et al. (2003).
Let \( \{ \tilde{O}_u \}_{u=1}^m \) (with \( \sum_{u=1}^m \tilde{O}_u = 1 \)) be the discrete representation (histogram) of the background in the feature space and \( \tilde{O}_v \) be its smallest nonzero entry. This representation is computed in a region around the target. The extent of the region is application dependent and we used an area equal to three times the target area. The weights:

\[
\begin{align*}
\nu_u &= \min \left( \frac{\tilde{O}_v}{\tilde{O}_u}, 1 \right) \\
\end{align*}
\]

are similar in concept to the ratio histogram computed for back projection (Swain, M. and D. Ballard, 1991). However, in our case, these weights are only 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 \( \nu_u \), i.e., are prominent in the background. The new target model representation is then defined by:

\[
q_u = CV_u \sum_{i=1}^n k \left( \frac{\| x_i^* \|^2}{h} \right) \delta \left( b(x_i^*) - u \right)
\]

Compare with (2) similarly, the new target candidate representation is:

\[
\rho_u(y) = CV_u \sum_{i=1}^n k \left( \frac{\| y - x_i \|^2}{h} \right) \delta \left( b(x_i) - u \right)
\]

The target localization algorithm is described as follow:

1- Calculate the target model \( \{ q_u \}_{u=1}^m \) and its position \( \tilde{y}_0 \) in first frame.
2- Calculate \( \{ \rho_u(\tilde{y}_0) \}_{u=1}^m \) and compute \( \rho = \rho_u(\tilde{y}_0) \cdot q_u \) in tent frame.
3- Obtain the weight \( w_i \) using (6).
4- Finding the new target location using (7).
5- If \( \| \tilde{y}_1 - \tilde{y}_0 \| < \varepsilon \) otherwise \( \tilde{y}_0 \leftarrow \tilde{y}_1 \) go to step 1.

3. Wavelet Transform:

3.1. Overview of the Wavelet Transform:

Unlike the Fourier transform, in which basis functions are sinusoids and redundant, the wavelet transforms are based on short-duration waves, called wavelets, of different frequency and restricted duration. This characteristic makes them a favorable choice to provide us with the frequency as well as temporal information for a given signal (Rafael, C. Gonzalez, Richard E. Woods, 2002). They can absolutely be implemented using multi-resolution techniques (Mallat, S.). The advantage of such an approach is that some features which might not be detected at one resolution may be found at some other resolutions. The 2-D Fast Wavelet Transform (FWT) result in four sub-band images in each level where, dH, dV, dD denote the approximation, horizontal detail, vertical detail, and diagonal detail sub-band images parameters, respectively (Airplane detection and tracking using wavelet features and SVM, 2009). The size of each of the four sub-band images is half of that of the input image.

3.2. Wavelets Coefficients from Each Channel:
As our first step, we divide image in first frame and other frames (incoming frames) which localized by mean shift procedure into its three RGB channels. In the first resolution level, we apply 4th level Daubechies’s wavelet in each RGB channel to get images of 240×320 pixels. Images contain approximation, horizontal detail, vertical detail, and diagonal detail. A similar procedure is applied on 240×320 approximation image for each RGB channel get next images of 120×160 pixels. Fig. 1 shows how four sub-images with size of 120×160 are obtained from R channel of original image. We also use this method for G and B channels. As our final step depicted in Fig.1, we have twelve sub-images with size of 120×160 for each color image. Here we obtain a feature vector with size of 19,200 to train an RVM classifier which is discussed in the following section.

4. Relevance Vector Machine
4.1. RVM for Classification:

As mentioned extraction of tracked target in rectangular region is done by trained RVM classifier which defines as following:

\[ \hat{y}(X; \tilde{W}) = \sum_{i \in \mathcal{N}} w_i K(X, X_i) \omega_0 \]  

Where \( K(X, X_i) \) is a kernel function, effectively defining one basis function for each example in the training set. Relevance vector machine (RVM) is a Bayesian framework for achieving the sparse linear model. In sparse model, the majority of the \( W \)'s are zero. The sparsity of model is based on a hierarchical prior, where an independent Gaussian prior is defined on the weight parameters in the first level:

\[ P(W | \alpha) = \prod_{i=1}^{N} N(w_i | 0, \alpha_i^{-1}) \]  

Where \( \alpha = (\alpha_1, \alpha_2, ..., \alpha_N) \) is a vector consisting of \( N \) hyper parameters. An independent Gamma hyper prior is used for the variance parameters in the second level:

\[ p(\alpha_i) = \text{Gamma}(a, b) \]  

Where \( a \) and \( b \) are constants. The key point of this method is using the maximum a posteriori (MAP) instead of maximum likelihood (ML) for the weight estimation. Given the \( N \) pairs of training data \( \{X_i, t_i\}_{i=1}^{N} \), the dataset likelihood is defined by applying the logistic sigmoid link function \( \sigma(y) = \frac{1}{1 + e^{-y}} \) to \( \hat{y}(X) \); the Bernoulli distribution for \( P(t | X) \):

\[ P(t | \tilde{W}) = \prod_{i=1}^{N} \sigma \{ y(X_i; \tilde{W}) \}^{t_i} [1 - \sigma \{ y(X_i; \tilde{W}) \}]^{1-t_i} \]  

Where class label is denoted by \( t_i \in \{0, 1\} \). The parameters \( w_i \) are then obtained by maximizing the posterior distribution of the class labels given the input vectors with respect to prior information. For this maximization, a numerical method is suggested as follows:

1. For the current, fixed \( \alpha \), values of, the most probable weights \( W_{\text{MP}} \) are found, giving the location of the mode of the posterior distribution.

Since \( P(t | W, \alpha) \sim P(t | W) P(W | \alpha) \) this is equivalent to finding the maximum, over \( W \), of:

\[ \log \{ P(t | \tilde{W}) P(W | \alpha) \} = \sum_{i=1}^{N} t_i \log y_i + (1 - t_i) \log(1 - y_i) - \frac{1}{2} \tilde{W}^{T} A \tilde{W} \]  

With
2. Laplace’s method is simply a quadratic approximation to the log-posterior around its mode. The quantity (eq. 15) is differentiated twice to give:

$$\Delta W \Delta \hat{W} \log P(W | F, A)_{W_{ij}} = 0 - (\Phi^T B \Phi + A)$$

(16)

Where

$$B = \text{diag}(B_1, B_2, \ldots, B_N)$$

and

$$B_n = \sigma(y(X_n)) [1 - \sigma(y(X_n))]$$

The posterior is approximated around $W_{ij}$ by a Gaussian approximation with Covariance:

$$\sum = (\Phi^T B \Phi + A)^{-1}$$

(17)

And mean

$$\mu = \Phi^T B \mu$$

(18)

3. Using the statistics $\Sigma$ and $\mu$ of the approximation, the hyper parameters $\alpha$ are updated as follows:

$$\alpha_i^{\text{new}} = \frac{\gamma_i}{\mu_i^2}$$

(19)

Where $\mu_i$ is the $i$-th posterior mean weight from (19) and

$$\gamma_i = 1 - \alpha_i^{\text{old}} N_i$$

which $N_i$ is the $i$-th diagonal element of $\Sigma$. Since computing the $\mu$ and $\Sigma$ based on above mentioned steps takes so much time, we use incremental DFT-RVM for simplicity on implementation.
4.2. RVM Initialization and Train:

In the first frame, we draw an ellipse around the desired object in order to separate the target from its surrounding background. The region of target is obtained as an ellipse at our first frame of image, then the points inside the ellipse are labeled with one and points outside are labeled with zero as depicted in Fig. 2. This results in a binary classification problem which can appropriately be solved using RVM classifier. We train RVM network with features vectors, derived from the first image as previously described in Section 2.2. (Tipping, M.E., A. Faul, 2003)

5. Using a Kernel Based on the Metric Distance Transform:

The equivalence of the mean shift procedure to gradient ascent on the similarity function holds for kernels that are radially symmetric, non-negative, non-increasing and piecewise continuous over the profile (Cheng, Y.Z., 1995). A radially symmetric kernel can be described by a 1D profile rather than a 2D (or higher order) image. The usual choice for K is the optimal Epanechnikov kernel (E-kernel) that has a uniform derivative of G=1 which is also computationally simple. However, in tracking an object through a video sequence and applying the mean shift algorithm to move the position of the target window, the bounds of the domain R2 are altered on each successive application of the algorithm. In most instances, the target does not have radial symmetry, so the use of a E-kernel includes foreground as background, or background as foreground pixels, or both. Depending on the shifting of pixels between background and foreground, and on the similarity of the two color distributions (in a worst case the background has similar properties to the target), then multiple modes are formed in the pdf. In this case local and global modes don’t correspond and the tracker is likely to fail.

Therefore, our contribution is to use a distance transform (DT), matched to the shape of the tracked object, as a kernel function. Although this kernel does not change shape through the sequence, it can change size, scaling as the subject expands or contracts in the camera field of view. For the DT each foreground pixel is given a value that is a measure of the distance to the nearest edge pixel. The edge and background pixels are set to zero. We use the normalized Metric distance Transform (MDDT) rather than the true Euclidean distance and isotropic descent kernel, as it is an efficient approximation (A Distance Transformation for Kernel based Object Tracking-Saeed, 2009), as shown in Fig. 3. The MDDT kernel better represents the color distribution of the tracked target, yet retains the more reliable centre weighting of the radially symmetric kernels.

Let P be a binary picture (defined on grid G) in which

\[
\phi = \{ p : p \in G & P(p) = 1\}\; \text{or}\; \xi = \{ p : p \in G & P(p) = 0\}
\]

Let \(P\) be a binary picture (defined on grid G) in which

\[
\phi = \{ p : p \in G & P(p) = 1\}\; \text{or}\; \xi = \{ p : p \in G & P(p) = 0\}
\]

For any grid metric \(d_n\), distance transform of \(P\) associates with every pixel \(p\) of \(\phi\) the \(d_n\) distance from \(p\) to \(\phi\). We assume that pixels of the background component (i.e., containing all pixels outside of the rectangular region G) all have value 0. The \(d_n\) or \(d_s\) distance transform of \(p\) can be computed by performing a series of local operations while scanning G twice. For any \(p \in G\) let \(B(p)\) be the set of pixels (4-or 8-) adjacent to \(p\) that precedes \(p\) when \(G\) is scanned in standard order.

If \(p\) has coordinates (\(xy\),B contains (\(xy + 1\)) and (\(x - 1y\)), and if we use 8-adjacency it also contains (\(x + 1y + 1\)) and (\(x - 1y + 1\)). For first scan we have

\[
\phi = \{ p : p \in G & P(p) = 1\}\; \text{or}\; \xi = \{ p : p \in G & P(p) = 0\}
\]
Fig. 3: Metric \( d \), distances transform. (a) Binary image, (b) \( d \) distance transform. After first scan, (c) \( d \) distance transform after the first scan, (d) \( d \) distance transform after the second scan and (e) \( d \) distance transform after the second scan.

\[
f_1(p) = \begin{cases} 
0 & \text{if } p \in <P> \\
\min f_1(q) + 1 : q \in B(p) & \text{if } p \notin <P>
\end{cases}
\]  

(21)

Compute \( f_1(p) \) for all \( p \in G \) in a single standard scan of \( G \); for each \( p, f_1 \) has already been computed for all of the \( q \)'s in \( B(p) \), if \( p \) is on the top row or in the left column of \( G \), some of these \( q \)'s are outside \( G \) with \( f_1 = 0 \) and for second scan we have:

\[
f_2(p) = \min\{f_1(p), f_2(p) + 1 : q \in A(p)\}
\]  

(22)

Compute \( f_2(p) \) for all \( p \in G \) in a single reverse standard (i.e., right-to-left, bottom-to-top) scan of \( G \) (each \( p, f_2 \) has already been computed for all of the \( q \)'s in \( A(p) \) or is known because they are outside of \( G \)).

In this work, \( d \) transform is applied to the target which separated from the background by RVM classifier. This weighting can increase the accuracy and robustness of representation of the pdf's as the target moves, excluding the peripheral pixels that occur within a radially symmetric window. Applying the MDDT kernel
to the region of interest, and weighting the colour distributions accordingly, we determine whether the exclusion of the erroneous background pixels, for example, from the density estimate of the target, and giving increased weighting to those more reliable pixels towards the centre, will outweigh the possibility of forming false modes. Of course, although the MDDT may produce false modes, this also occurs with radially symmetric kernels due to badly defined densities. As the scale of the target may change, the size of the kernel is adapted accordingly. When the exact location of the target is found, we measure the similarity criterion for three values

\[ h = [0.9 1.1] \times h_{\text{prev}} \] and its maximum is named \( h_{\text{opt}} \). \( h_{\text{prev}} \) (Scale for current frame) is computed as below:

\[ h_{\text{new}} = \lambda h_{\text{opt}} + (1 - \lambda) h_{\text{prev}} \]  

(23)

The best result, \( h_{\text{opt}} \), yielding the largest Bhattacharyya coefficient is retained.

In applying the MDDT kernel to the mean shift procedure, due to the object shape changes in the long or short term, we have two options. First we can define the MDDT on the basis of the first frame, and use this for the whole sequence. Second we can segment the subsequent frame and apply a different kernel weighting. The algorithm described above applies to the first option, which is used for the experimental results in the next section. To modify or update for each model frame, for example, the segmentation and MDDT computation code is included inside the outer repeat-until loop. Otherwise, the iterative algorithm that we use to test and compare the respective kernels is the same as that defined in reference Comaniciu et al. (2003):

1. Define target centric, \( y_0 \), in first frame.
2. Use wavelet transform as feature extraction
3. Train the Relevance Vector Machine (RVM) algorithm
4. Compute (normalized) MDDT-kernel form model histogram, \( q \), in color space
5. Repeat
6. Fetch next frame
7. Repeat
8. Compute candidate histogram \( p(y_0) \) in color
9. Space using MDDT-kernel
10. Find next location \( y_1 \) of candidate compute error
11. \( e = y_1 - y_0 \)
12. Set \( y_0 = y_1 \)
13. Until \( e \leq \epsilon \), an error threshold or maximum iteration reached
14. \( y_0 \) is the new location
15. Until (end of input sequence)

Adaptive kernels have also been used by Porikli and Tuzel (2003). Like our approach, their algorithm does not maintain fully the mean-shift convergence conditions. However, the MDDT presented here satisfies it partly with its decreasing Profile. Practical tests show that even if theoretical convergence conditions are not fully satisfied, convergence is achieved.

6. Experimental Results:

In this section, we present the evaluation of the modified mean shift object tracking using the MDDT-kernel in comparison with the radially symmetric E-kernel. We track moving objects, a static object with a moving camera and a combination of the two. All the tests were carried out on a Pentium 4 CPU 3.60 GHz with 1GB RAM. So that it would be reasonable to assume a considerable increase in processing speed if re-implemented with more advanced sets. In the first experiment, we compare the tracking of a moving car in a video sequence that includes 350 frames of 480 × 640 pixels, comparing the normal E-kernel with the MDDT kernel. Fig.4 shows the value of iteration computed for each frame. In Fig.5 distance function which calculated by the Bhattacharyya coefficient (Eq. 2) is presented. The peak in the E-kernel data is 0.636 which increases to the 0.7 in MDDT. Fig.6 (a) and (b) show some examples, frames 1, 35, 60 and 75, from the whole sequence. In frame 60 some of the original car is still contained within the window, but after the 75th frame, the car is lost completely in Fig.6 (a), as the tracker finally latches on to another crossing car. This demonstrates that the inclusion of the background of the tracked car (in this case another car) includes pixels that are similar in color space, so that the algorithm fails to identify the correct distribution in succeeding frames and hence follows the wrong target. Fig.7 shows the similarity surfaces made by candidate models in frame 52 with E-kernel and MDDT kernel, respectively. Initial point is center of target model in frame of 48
and extend of simulation is 60×60.

Fig. 4: Iteration Value for selected Frames

Fig. 5: The Bhattacharya distance values for the crossing car

(a) E-Kernel

(b) MDDT-Kernel
Fig. 6: Tracking the crossing car

![Fig. 6](image)

Fig. 7: The similarity surfaces (values of the Bhattacharyya coefficient) for frame 52. The initial points and convergence points are shown. (a) The result from the E-kernel. (b) The result from the MDDT-kernel.

In terms of complexity, computed from 20 executions of the program, the average selected frames per second of the MDDT kernel and the E-kernel are 12.36 and 17.27 respectively, while This time length to 24.52 when trained RVM is used for target tracking (2008). The maximum numbers of iterations within a single frame are 14 and 20, respectively. The average times per frame are roughly comparable because although the speed of convergence is quicker with the MDDT-kernel, additional processing is required to segment the target window, in order to get more robust and accurate tracking. To further test the robustness of the MDDT-kernel algorithm and convergence properties, the contrast of input selected frames is increased which can result by different environment illumination. (Fig.8). From Table 1, which shows quantitative results, the MDDT kernel algorithm needs on average only 7.6 iterations to converge to the optimal result, but the E-kernel needs 14 iterations on average. Again, the greater complexity of computing the MDDT kernel is balanced by the greatly reduced number of iterations, so the processing speed per frame is comparable. As it shown self-RVM classifier among three methods has most consuming time.

![Fig. 8](image)

**Fig. 8:** (a) Original image. (b) Intensity of pixels in selected frame is increased which can result by different environment illumination.

<table>
<thead>
<tr>
<th>Method</th>
<th>Average iterations</th>
<th>CPU time (sec./frame)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>max</td>
<td>min</td>
</tr>
<tr>
<td>E-kernel</td>
<td>14</td>
<td>0.4505</td>
</tr>
<tr>
<td>RVM</td>
<td>-</td>
<td>0.7634</td>
</tr>
<tr>
<td>Proposed Method</td>
<td>7.6</td>
<td>0.4240</td>
</tr>
</tbody>
</table>

Table 1: Comparison results of consuming time in three methods
Fig. 9: An example for tracking. (a) We show four frames from a 250-frame long sequence. (b) The corresponding confidence map. We track a blue car that is undergoing rotations and partial occlusions. The sequence was taken with a hand-held camera. (c) Result obtained by employing morphological operation.

Conclusions:

We have described the implementation of a scaling, normalized Metric-8 distance kernel as a weighting and constraining function applied to the mean shift tracking algorithm that maximizes the similarity between model and candidate distributions in color space. The tracked target for weighting is differentiated from background by RVM classifier in each frame. This work uses 4th level Daubechies’s wavelet in each RGB channel to extract feature vector. In comparison with the E-kernel, used as exemplar of a radially symmetric function, application of the reject false nodes those are caused by the inclusion of changing background pixels. The processing time is sufficiently small for real time operation, as the added cost of foreground-background separation is offset by the more rapid finding of the correct mode. The results presented on a number of video sequences show that the MDDT-kernel algorithm performs well in terms of improved stability, accuracy and robustness on camera motion and partial occlusions.

REFERENCES


