Robust and Accurate Object Tracking under Various Types of Occlusions

Jiyan PAN, Bo HU, *Member, IEEE*, and Jian Qiu ZHANG, *Senior Member, IEEE*

***Abstract* – We propose a complete solution to robust and accurate object tracking in face of various types of occlusions. As incorrect judgment of occlusion situation and improper update of target template are much more likely when occlusions occur, preserving tracking stability and precision under occlusions is a challenging task. In order to overcome these difficulties, we first propose a content-adaptive progressive occlusion analysis (CAPOA) algorithm. By combining the information provided by spatiotemporal context, reference target, and motion constraints together, the algorithm makes a clear distinction between the target and outliers. A template mask is subsequently formed to prevent infiltration of outliers. Accurate tracking of an occluded target is achieved by rectifying the target location using the variant-mask template matching (VMTM). In order to deal with template drift during the process of template update, we propose a drift-inhibitive masked Kalman appearance filter (DIMKAF) which accurately evaluates the influence of template drift when updating the masked template. As a result, the appearance filter reaches an optimal balance between reducing template drift and keeping track of the target appearance variations. A fast algorithm of the DIMKAF is also given. Finally, we devise a local best match authentication (LBMA) algorithm to handle complete occlusions, so that a much more trustworthy detection of the end of an arbitrarily long complete occlusion is achieved. Both the real-world and synthetic video sequences show that our proposed solution tracks targets reliably and accurately no matter when they are under: short-term, long-term, partial or complete occlusions.**

***Index Terms* – Object tracking, occlusion handling, template drift, Kalman filter, template matching.**

I. INTRODUCTION

OBJECT tracking is a very important aspect of computer vision and has a very wide range of applications. Over recent years, much research has been devoted to object tracking under occlusions, because in real-world tracking, a target being partly or entirely covered by outliers for an uncertain period of time is normal. This phenomenon however, significantly increases the difficulty of object tracking: the precision of locating the target normally drops a lot in face of heavy occlusions, and more seriously, occlusions are likely to damage the template and eventually results in the loss of the target. Specifically, occlusions pose four challenges to object tracking algorithms. In the remainder of this section, we first describe the four challenges and review former efforts trying to solve them, and then we outline the approaches that we propose to overcome the challenges.

The first challenge is how to robustly determine the portion of the target that is occluded. As will be mentioned later, this is prerequisite to generating a correct template mask and preventing infiltration of

Manuscript received December 15, 2006. This work was supported by the National Basic Research Program 973 under Grant No. 2006CB705700.
The authors are with the Department of Electronic Engineering, Fudan University, Shanghai 200433, China (telephone: 86-021-65643633; 86-021-65642762; 86-021-55664226, e-mail: jiyanpan@fudan.edu.cn; bohu@fudan.edu.cn; jqzhang01@fudan.edu.cn).

occluders. Determining occlusion status is very hard for general-purpose trackers, where the only knowledge available on the target is its initial appearance and the camera itself could have arbitrary motions. When some parts of an occluder are similar to those of the target, they are easily undetected by tracking algorithms. Detection of occlusion status is further complicated by the variation of the target appearance, because tracking algorithms would have a hard time judging whether the changes in pixel grayscale are caused by occlusions or the target itself.

Various approaches that analyze occlusion situations have been proposed. In [25] and [26], all the foreground objects are localized by background subtraction and assigned a track index. The occurrences of occlusions are inferred simply by monitoring whether two tracks merge. Although this method is reliable, yet it only works with a fixed camera and known background. In addition, appearance models need to be established for all the foreground objects. A mixture of three distributions is used in [24] to model the observed value of each pixel, where outliers are characterized by the "lost" component which has a uniform distribution; some other approaches declare outlier pixels by examining whether the measurement error exceeds a certain value [7], [8], [23]. These algorithms work well when the statistical properties of occluders happen to agree with their assumptions. Unfortunately, in most cases the assumptions do not hold, because in real-world tracking scenarios, an occluder might be similar in color to the target, or fail to satisfy a uniform distribution, or occlude the target for a long time. In addition, since these algorithms operate on a single-pixel basis, they do not utilize any information provided by spatial context. Contextual information is exploited in [11] and [12]. In [11], before locating the target, small *blocks* of the current image are compared with the template to decide whether they are occluded. Reference [12] performs motion estimation between adjacent frames and utilizes motion vectors to handle occlusions. As these methods detect occlusions by comparing small image blocks, not just individual pixels, they have better performance in terms of analyzing occlusion situation, but as they either employ the template alone or the previous frame alone, errors are observed to frequently occur and propagate away. Further improvement can be made by making use of the joint information provided by *both* the template and the previous frame. It should be noted that the multiple hypotheses in particle-filtering based trackers [23] might enable some particles to find the target again when an occlusion is over. However, recapture of the target is conditioned on effective detection of occluders and protection of appearance models. Particle filtering alone could not achieve this, because it does not readily possess the capability of discriminating target from outliers [27], [30]. A specific occlusion-detection scheme

should be added as is implemented in [23]. The problem here is how to design such a scheme that is robust enough.

The second challenge is how to accurately locate the target when the occlusion situation of the current frame is still unknown. This is actually a chicken and egg problem: the occlusion situation must be obtained *before* the target can be accurately located by masking out the occluded portion of it, while the occluded portion of the target can reliably be determined by comparing with the template only *after* the correct location of the target is given in the first place. Using histograms to represent the target [28] could be a solution to this problem, because no structural constraint is imposed when searching for the target. However, a satisfactory performance is guaranteed only when the non-occluded part of the target has a histogram similar to that of the entire target. This assumption is often violated in real-world tracking. Therefore, an approach that *explicitly* addresses this problem should be developed.

The third challenge is how to properly update the template so as to keep track of the changes in the target appearance while preventing damages caused by outliers and template drift. The most straightforward method directly replaces the template every frame (or every n frames) with the image region believed to be the target [19], [20]. Indiscriminative update of the template in this method, however, results in infiltration of occluders into the template [8], [11]. In addition, hasty update of the template is found to suffer from gradual drift of the target out of the template, which is referred to as *template drift* [2], [10].

Infiltration of occluders can be solved by applying a masking mechanism when updating the template as is done in [7], [8], and [23], where pixels believed to be occupied by outliers are left out or given a lower weight. Generating an effective occlusion mask, however, depends solely on acquiring a correct analysis of the occlusion situation, and the latter task, which is described in challenge 1, is by no means as simple.

The cause of template drift has been preliminarily and qualitatively investigated in the literature [2]-[4], where template drift is ascribed to the accumulation of small appearance errors introduced each time the template is updated. To reduce template drift, the initial template can be utilized to serve as a benchmark when locating the target position in subsequent frames [2], [3], [10], but the resort to the first template is less effective when the target appearance undergoes major changes.

The algorithm in [24] uses an EM-learned “stable component” to stabilize tracking, which helps reduce

template drift. However, the algorithm does not take into account the small appearance errors that lead to template drift. The distribution of such errors is not necessarily a uniform one and might therefore be learned by the "wandering component" (a component proposed in [24] that models rapid temporal variations), which undermines the performance in suppressing template drift.

Robust against noise due to its smoothing effect, Kalman appearance filter is a powerful tool to deal with template drift [4], [9]. However, both [4] and [9] require manual selection of a Kalman gain for individual sequences and the Kalman gain remains fixed once selected. Kalman appearance filter is also applied to template update in [7] and [8], where the Kalman gain is allowed to fluctuate according to how intensively the target appearance varies. Nevertheless, Reference [7] and [8] either assume the state transition noise or the measurement noise to be constant. These are the two extreme cases that seldom occur in real-world tracking scenarios. Consequently, much template drift is still observed. In order to achieve a better performance, we need to explicitly model template drift when designing the Kalman appearance filter.

The fourth challenge is how to reliably detect the reemergence of the target and recapture it after it is completely occluded for some time. Setting a similarity threshold is one method. Yet the optimal threshold value is difficult to determine, because it varies with different video sequences. This problem is circumvented in [7] and [8], where the image region that matches the best with the template over a prefixed duration is assumed to be the reappearing target. Nevertheless, the artificial limit of a maximal duration of the complete occlusion is a major drawback of this approach. Therefore, we need a new strategy that could remove such a restriction when detecting the end of complete occlusions.

In this paper, we propose a set of methods to achieve robust and accurate object tracking under occlusions. Consisting of both a normal mode and a complete-occlusion mode, the overall structure of our object tracking solution is illustrated in Table I and will be detailed in Section II. There are four key components in our solution that address the four aforementioned challenges respectively. They are highlighted in gray in Table I. The four key components are listed as follows:

1. Content-Adaptive Progressive Occlusion Analysis (CAPOA) algorithm which analyzes the occlusion situation within a given region of interest (ROI) and generates corresponding template mask. The CAPOA algorithm features much higher discriminating capability against outliers than other approaches.
2. Variant-Mask Template Matching (VMTM) operation which rectifies erroneous target location

caused by occlusions. In VMTM, the template mask varies with the candidate target regions so that the non-occluded portion of the target is always utilized to align the target from the initial erroneous location to its true location.

3. Drift-Inhibitive Masked Kalman Appearance Filter (DIMKAF) which quantifies the measurement noise caused by template drift and integrates it into the Kalman appearance filter. Explicit formulization of the influence of template drift allows for much more effective suppression of template drift.
4. Local Best Match Authentication (LBMA) algorithm which reliably detects the end of a complete occlusion. The possible remerging target is selected as the best match in a local temporal period. The authenticity of this guess is further examined by a novel strategy. The LBMA algorithm is not restricted by the maximum duration of the complete occlusion.

All the aforementioned algorithms enable our proposed solution to achieve robust and accurate object tracking under various kinds of occlusions, including: short-term partial, long-term partial, short-term complete, and even long-term complete ones.

The remainder of this paper is organized as follows. Section II provides a detailed description of the overall structure of our proposed solution. In Section III, after briefly reviewing the template matching technique, we discuss the DIMKAF for the sake of consistency and clarity. The CAPOA algorithm is then detailed in Section IV. Section V focuses on the VMTM operation. We follow in Section VI by describing the LBMA method. Experimental results are presented in Section VII, and Section VIII concludes this paper.

II. Overall Structure

Table I shows the overall structure of our proposed solution. The initial target is specified by selecting a target region either manually or automatically [29]. Typically covering a rectangular area, the target region is referred to as the region of interest (ROI) which tightly frames the target. Note that some background pixels might also get included into the ROI during the initialization, but this does not matter much, because they will be discriminated by the CAPOA algorithm later. We also initialize an *outlier map* (denoted as U_n) which represents the locations of outlier pixels in frame n. The outlier map is a binary matrix which has a value of 1 where a pixel does *not* belong to the target. The template $\hat{T}$, which reflects the current estimated target appearance, is initialized by sampling from the initial ROI through

coordinate transformation. The initial template mask is just an array of ones, indicating no template pixel is masked. As an important information source in the CAPOA algorithm, the *reference target* (denoted as T_{ref}) is initialized as the initial ROI.

In the normal mode, we predict the target location using the adaptive-velocity model proposed in [21] after a new frame comes in. The *approximate* target region (ROI_1) is then obtained through the first masked template matching. Template matching is performed by finding the coordinate transformation parameters that minimize the matching error. However, the target location acquired by the first template matching might be erroneous because it uses the template mask generated according to the occlusion situation of the *previous* frame. In order to rectify the target location, we analyze the occlusion situation within ROI_1 using the CAPOA algorithm and then perform the VMTM based on the result of the occlusion analysis. The VMTM yields a new ROI (ROI_2) whose occlusion situation is analyzed by the CAPOA algorithm again. The resulting occlusion situation of ROI_2 generates a new template mask (M') which guides the second masked template matching. This template matching determines the final ROI (ROI_3), within which the occlusion situation is analyzed by the CAPOA algorithm to yield the final outlier map and template mask. Having obtained the accurate target location and the final template mask, we update the template using the DIMKAF. It should be noted that when analyzing the occlusion situations of ROI_2 and ROI_3, we only need to determine the occlusion statuses of newly covered image regions.

When more than 85% of the target is occluded, our proposed solution enters the complete-occlusion

TABLE I
The Overall Structure of Our Proposed Tracking Solution

Initialize *the outlier map U_1 by selecting the target region (ROI). $U_1(\boldsymbol{x})=0$ if $\boldsymbol{x}$ belongs to the target region. $U_1(\boldsymbol{x})=1$ elsewhere.*
Initialize *the template $\hat{T}$ by sampling the ROI through coordinate transformation. The reference target T_{ref} is initialized as the ROI.*
Initialize *the template mask M_1 to be an array of ones with the same size as the template.*
Clear *the complete-occlusion flag F_{OCC} and the transition flag F_{TRA}.*
For *frame index n = 2,3,…*
  If *$F_{OCC} = 0$*
    / normal mode */*
    Predict *target location using adaptive-velocity model.*
    Run *ROI_1 =* FTM*$(\hat{T}, I_n, M_{n-1})$ to obtain the approximate target region ROI_1.* FTM *denotes "first template matching". I_n is frame n.*
    Run *$[U_{prlm}, dummy]$=*CAPOA*$(ROI_1, T_{ref}, I_{n-1}, U_{n-1})$ to obtain the preliminary outlier map U_{prlm}. "dummy" means a dummy variable.*
    Run *ROI_2 =* VMTM*$(\hat{T}, I_n, U_{prlm})$ to rectify the target region from ROI_1 to ROI_2. U_{prlm} is used to generate the variant mask M_A.*
    Run *$[dummy, M']$=*CAPOA*$(ROI_2, T_{ref}, I_{n-1}, U_{n-1})$ to get a new template mask M'.*
    If *more than 85% of ROI_2 is occluded*
      Set *the complete-occlusion flag F_{OCC}.*
    Else
      Run *ROI_3 =* STM*$(\hat{T}, I_n, M')$ to obtain the final target region ROI_3.* STM *denotes "second template matching".*
      Run *$[U_n, M_n]$=*CAPOA*$(ROI_3, T_{ref}, I_{n-1}, U_{n-1})$ to acquire the final outlier map and template mask.*
      Run *$\hat{T} \leftarrow$* DIMKAF*$(\hat{T}, ROI_3, M_n)$ to update the template.*
      Update *T_{ref} by incremental interpolation and filtering.*
      Perform *Kalman filtering on target speed.*
    End
  Else
    / complete-occlusion mode */*
    If *$F_{TRA} = 0$*
      Predict *target location using constant-velocity model.*
      Perform LBMA *to detect the end of complete occlusion.*
      If *the end of complete occlusion is detected*
        Set *the transition flag F_{TRA}.*
      End
    Else
      Perform *template matching without masking.*
      If *it is the last frame of the transition period*
        Clear *F_{OCC} and F_{TRA}.*
        Reinitialize *U_n and M_n.*
      End
    End
  End
End

mode, in which the target location is predicted using a constant-velocity model. The reappearance of the target is reliably detected by the LBMA method. Once the end of a complete occlusion is declared, the tracker undergoes a 5-frame transition period in which we neither use the template mask nor update the template. At the end of the transition period, the outlier map and the template mask are reinitialized and our tracker resumes the normal mode.

III. Updating Template Using Drift-Inhibitive Masked Kalman Appearance Filter

A. Locating the Target by Masked Template Matching

Denoted as $T(\boldsymbol{x})$, the template of a target is a sub-image reflecting the grayscale appearance of the target. Because of the existence of measurement noise, the true appearance of the template cannot be obtained. As a result, what is actually utilized by the tracking algorithm is the estimated template $\hat{T}$.

For each frame in a video sequence, the estimated template is mapped to the frame by coordinate transformation $\phi(\boldsymbol{x};\boldsymbol{a})$ which describes the motion of the target. The type of the transformation is determined by its parameter vector $\boldsymbol{a}$. In this paper, target motion is characterized by translation and scaling; all other types of motion (including in-plane/3D rotation, non-rigid deformation, etc) are regarded as variations in target appearance. Therefore, $\boldsymbol{a}$ has three components which describe horizontal translation, vertical translation and scale of the target, respectively.

The location of the target in frame n is determined by performing the parameter search:

$$\hat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}} \frac{1}{\mathrm{sum}(M)} \sum_{\boldsymbol{x}\in\boldsymbol{\Omega}_T} \left| I_n[\phi(\boldsymbol{x};\boldsymbol{a})] - \hat{T}(\boldsymbol{x}) \right| \cdot M(\boldsymbol{x}), \tag{1}$$

where $\hat{\boldsymbol{a}}$ is the estimated transformation parameter vector, I_n denotes frame n, $\boldsymbol{\Omega}_T$ represents the ensemble of the template pixels in the template coordinate system, and M denotes the template mask. M has a value of 0 where the corresponding template pixel is occluded, and a value of 1 elsewhere. sum(M) calculates the number of non-occluded template pixels. Equ. (1) represents the masked template matching operation. In implementation, $\hat{\boldsymbol{a}}$ is obtained by various searching algorithms [1], [15]-[17] or through particle filtering [23], and the initial searching point can be chosen according to the adaptive-velocity model proposed in [21]. One thing that should be noted here is that only *translational* transformation parameters are involved in the first template matching, because we do not have a reliable template mask yet. In the second template matching when a new template mask is acquired, *all* the transformation

parameters are under search (see Table I). Hereby we follow the strategy of obtaining optimal coordinate transformation parameters from lower-complexity ones to higher-complexity ones, so as to increase stability and robustness [22].

Each time the estimated transformation parameter vector is obtained by (1), the *non-occluded* part of the estimated template $\hat{T}$ is updated from the measured target appearance $I[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})]$ to incorporate possible variations in the appearance of the target. However, as (1) is always conducted in a discrete vector space, the quantization error between the acquired value $\hat{\boldsymbol{a}}$ and the true value $\boldsymbol{a}_0$ results in small "drift" of the measured target *appearance* $I[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})]$ from the true appearance $I[\phi(\boldsymbol{x};\boldsymbol{a}_0)]$. If we update the template without any discretion, such drift would gradually accumulate in the template each time it is updated and eventually lead to tracking failure. This is the ultimate cause of template drift. The errors in the measured target appearance caused by such small drift in each updating step can be viewed as a major component of measurement noise. In this paper, it is referred to as *drift noise*. Drift noise, along with camera noise, forms the final measurement noise. In order to ensure an optimal estimation of the true target appearance in face of the measurement noise, Kalman filtering is employed to update the template.

B. Applying Masked Kalman Appearance Filter to Template Update

The masked Kalman appearance filter is applied only on *non-occluded* template pixels in order to prevent infiltration of outliers. All the coordinate arguments $\boldsymbol{x}$ in Section III-B and III-C refer to non-occluded template pixels, that is, $M(\boldsymbol{x})=1$. Processing occluded template pixels will be discussed in Section III-D.

For the sake of simplicity, independence is assumed among template pixels, and the Kalman appearance filter is applied on a single-pixel basis. The state equation here can be expressed as:

$$T(\boldsymbol{x},n)=T(\boldsymbol{x},n-1)+\varepsilon_S(\boldsymbol{x},n-1), \tag{2}$$

where $T(\boldsymbol{x},n)$ denotes the grayscale of a template pixel $\boldsymbol{x}$ at frame n, and $\varepsilon_S(\boldsymbol{x},n-1)$ is the state transition noise which reflects the variation of the target appearance caused by the target *itself* from frame n–1 to frame n. It is reasonable to assume that $\varepsilon_S(\boldsymbol{x},n)$ is a zero-mean white noise with power spectrum $\sigma_S^2(\boldsymbol{x},n)$. For simplicity, we use "power" to refer to "power spectrum" in the discussion below.

The measurement equation is:

$$I_n[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})]=T(\boldsymbol{x},n)+\varepsilon_M(\boldsymbol{x},n), \tag{3}$$

where $\varepsilon_M(\boldsymbol{x},n)$ is the measurement noise which is also white and zero-mean. The power of $\varepsilon_M(\boldsymbol{x},n)$ is $\sigma_M^2(\boldsymbol{x},n)$, which is contributed to by the drift noise and the camera noise.

According to the theory of Kalman filtering [13], Equs. (4) to (7) form a complete iteration to update the estimated value of the template pixel:

$$\sigma_P^2(\boldsymbol{x},n) = \sigma_E^2(\boldsymbol{x},n-1) + \sigma_S^2(\boldsymbol{x},n-1), \tag{4}$$

$$G(\boldsymbol{x},n) = 1/[1+\sigma_M^2(\boldsymbol{x},n)/\sigma_P^2(\boldsymbol{x},n)], \tag{5}$$

$$\sigma_E^2(\boldsymbol{x},n) = [1-G(\boldsymbol{x},n)]\sigma_P^2(\boldsymbol{x},n), \tag{6}$$

$$\hat{T}(\boldsymbol{x},n+1) = \hat{T}(\boldsymbol{x},n) + G(\boldsymbol{x},n)\{I_n[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})] - \hat{T}(\boldsymbol{x},n)\} = \hat{T}(\boldsymbol{x},n) + G(\boldsymbol{x},n)\alpha(\boldsymbol{x},n). \tag{7}$$

Here, $\alpha(\boldsymbol{x},n)$ is the innovation at frame *n*; σ_P^2 and σ_E^2 are the powers of the prediction error and the estimation error, respectively. They are automatically calculated in the iterations of the Kalman filtering. What is left to be estimated are the powers of the two noise models, σ_S^2 and σ_M^2. They are related by the following equation [7]:

$$\sigma_\alpha^2(\boldsymbol{x},n) = \sigma_E^2(\boldsymbol{x},n-1) + \sigma_S^2(\boldsymbol{x},n-1) + \sigma_M^2(\boldsymbol{x},n), \tag{8}$$

where $\sigma_\alpha^2(\boldsymbol{x},n)$ is the power of the innovation. It can be approximated by averaging the squared innovations of non-occluded template pixels over space and time, that is,

$$\sigma_\alpha^2(\boldsymbol{x},n) \approx \frac{1}{N_L}\sum_{k=n-L+1}^{n}\sum_{z\in\boldsymbol{\Omega}_L(\boldsymbol{x})}[\alpha(z,k)]^2, \tag{9}$$

where *L* is the length of the temporal moving-average window, $\boldsymbol{\Omega}_L(\boldsymbol{x})$ represents the non-occluded part of a spatial neighborhood centered at the current pixel $\boldsymbol{x}$, and N_L denotes the number of pixels involved in the averaging process. In this paper, we set *L* to be 20 frames and $\boldsymbol{\Omega}_L(\boldsymbol{x})$ to be the non-occluded part of an 11-by-11 square region around $\boldsymbol{x}$.

According to (8), if the power of one noise model is obtained, the other one can be trivially calculated. Both [7] and [8] assume one of the two noise models to be constant in power. This assumption, however, is appropriate only in very specific tracking cases. In fact, the measurement noise power σ_M^2 can be quantitatively evaluated, as will be detailed in Section III-C. Therefore, σ_S^2 can be expressed as:

$$\sigma_S^2(\boldsymbol{x},n-1) = \sigma_\alpha^2(\boldsymbol{x},n) - \sigma_E^2(\boldsymbol{x},n-1) - \sigma_M^2(\boldsymbol{x},n). \tag{10}$$

In some cases, equation (10) yields a negative value, indicating little, if any, variation of the target appearance at this location is caused by the target itself. As a result, σ_S^2 should be set as zero and the value of σ_M^2 should be adjusted accordingly:

$$\sigma_M^2(\boldsymbol{x},n)=\sigma_\alpha^2(\boldsymbol{x},n)-\sigma_E^2(\boldsymbol{x},n-1). \tag{11}$$

Now we discuss the issue of initialization. The only term that needs to be initialized is the power of the estimation error. As the only cause of the error between the initial template and the true target is the camera noise, the power of the initial estimation error is therefore equal to the camera noise power:

$$\sigma_E^2(\boldsymbol{x},0)=\sigma_{MC}^2, \tag{12}$$

where σ_{MC}^2 is the camera noise power.

C. Correctly Evaluating the Measurement Noise Power to Add Drift-Inhibitive Feature

As is mentioned in Section III-A, the influence of template drift is reflected on the measurement noise of the Kalman appearance filter. Therefore, the filter can be made drift-inhibitive by obtaining the correct evaluation of the measurement noise power which has two components: the camera noise power σ_{MC}^2 and the drift noise power σ_{MD}^2.

1) Estimation of the Camera Noise Power

The camera noise power is assumed to be constant and can be acquired by referring to the specifications of the camera sensor or calculating the variance over all the pixels of a uniform grey background before the beginning of tracking:

$$\sigma_{MC}^2=\frac{1}{N_G}\sum_{\boldsymbol{x}\in\boldsymbol{\Omega}_G}\left[I_G(\boldsymbol{x})-\frac{1}{N_G}\sum_{\boldsymbol{x}\in\boldsymbol{\Omega}_G}I_G(\boldsymbol{x})\right]^2, \tag{13}$$

where I_G is the uniform gray background, $\boldsymbol{\Omega}_G$ represents the region under test, and N_G is the number of pixels within $\boldsymbol{\Omega}_G$.

2) Estimation of the Drift Noise Power

The evaluation of σ_{MD}^2 is not so straightforward, as it is not necessarily constant. As has been discussed in Section III-A, the discrepancy between $\hat{\boldsymbol{a}}$ and $\boldsymbol{a}_0$ leads to the inaccuracy of the transformed coordinate $\phi(\boldsymbol{x};\hat{\boldsymbol{a}})$ and hence the drift error in $I_n[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})]$. Fig. 1 indicates the situation in which the *true* position of a template pixel $\boldsymbol{x}$ might lie in a region $\boldsymbol{\Omega}_u$ which is centered at $\phi(\boldsymbol{x};\hat{\boldsymbol{a}})$ in the current frame I_n, and the *true* value of the pixel $\boldsymbol{x}$ is therefore equal to the value of a certain point within $\boldsymbol{\Omega}_u$, which is probably not $\phi(\boldsymbol{x};\hat{\boldsymbol{a}})$. Increasing precision of (1) results in a smaller size of $\boldsymbol{\Omega}_u$ and thus less drift noise.

For the simplicity of notation, we use $\boldsymbol{a}$ instead of $\boldsymbol{a}_0$ to denote the *true* value of the transformation parameter vector. The drift noise power of a template pixel located at $\boldsymbol{x}$ can be formulated as

$$\sigma_{MD}^2(\boldsymbol{x},n)=\int_{\boldsymbol{a}}\{I_n[\phi(\boldsymbol{x};\boldsymbol{a})]-I_n[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})]\}^2 p_{\boldsymbol{a}}(\boldsymbol{a}\mid\hat{\boldsymbol{a}})d\boldsymbol{a}\,, \tag{14}$$

where $\sigma_{MD}^2(\boldsymbol{x},n)$ is the drift noise power of pixel $\boldsymbol{x}$ at frame n, and p_a is the joint posterior distribution of the components of $\boldsymbol{a}$ after $\hat{\boldsymbol{a}}$ is given. The posterior distributions of individual transformation parameters are independent of one another when $\hat{\boldsymbol{a}}$ is in the close vicinity of $\boldsymbol{a}$, because in this case the selection of the value of a certain parameter is little affected by the values that the other parameters have taken. This condition is always satisfied when the target is under track, and (14) can therefore be written as

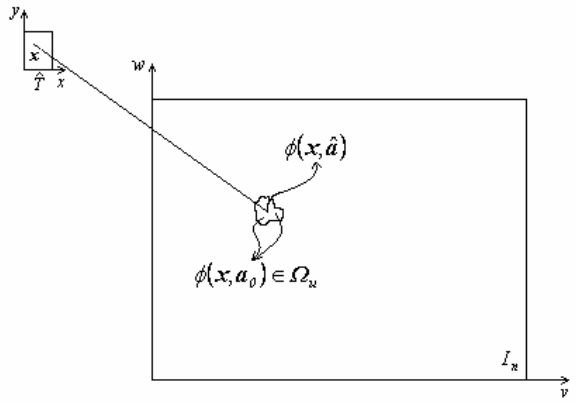


Fig. 1. Template drift occurs when the true mapped position $\phi(\boldsymbol{x};\boldsymbol{a}_0)$ lies in a region around the searching result $\phi(\boldsymbol{x};\hat{\boldsymbol{a}})$.

$$\sigma_{MD}^2(\boldsymbol{x},n)=\iiint_{a_1a_2a_3}\{I_n[\phi(\boldsymbol{x};\boldsymbol{a})]-I_n[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})]\}^2\prod_{i=1}^{3}p_i(a_i\mid\hat{a}_i)da_i\,, \tag{15}$$

where p_i is the posterior distribution of a_i, the i-th component of $\boldsymbol{a}$.

Now we focus on the calculation of p_i. According to Fig. 2, since $\hat{a}_i$ can only take discrete values, the likelihood of a_i is

$$P_i(\hat{a}_i\mid a_i)=\begin{cases}1, & |\hat{a}_i-a_i|\le\Delta_i/2\\ 0, & else\end{cases}, \tag{16}$$

where $P_i(\hat{a}_i\mid a_i)$ is the likelihood of a_i under the observation $\hat{a}_i$, and Δ_i is the final step size with which (1) searches for $\hat{a}_i$. From Bayes' rule, the posterior distribution of a_i can be expressed as

$$p_i(a_i\mid\hat{a}_i)=\frac{P_i(\hat{a}_i\mid a_i)p(a_i)}{\int P_i(\hat{a}_i\mid a_i)p(a_i)da_i}. \tag{17}$$

Substituting (16) into (17) yields

$$p_i(a_i\mid\hat{a}_i)=\begin{cases}\dfrac{p(a_i)}{\int_{\hat{a}_i-\Delta_i/2}^{\hat{a}_i+\Delta_i/2}p(a_i)da_i}, & |a_i-\hat{a}_i|\le\Delta_i/2\\ 0, & else\end{cases}. \tag{18}$$

Although it is difficult to acquire the exact value of $p(a_i)$, we can reasonably assume that $p(a_i)$ is

approximately constant within the integral interval, because $p(a_i)$ is relatively flat near its maximum and Δ_i is relatively small. Under this assumption, Equ. (18) is reduced to

$$p_i(a_i \mid \hat{a}_i) = \begin{cases} 1/\Delta_i, & |a_i - \hat{a}_i| \le \Delta_i/2 \\ 0, & else \end{cases}. \tag{19}$$

As $\phi(\boldsymbol{x};\boldsymbol{a})$ does not necessarily generate integer coordinates, sub-pixel interpolation is needed to promote the precision of template matching and noise evaluation. Considering both the interpolating performance and the computational burden, we choose bilinear interpolation to achieve this purpose.

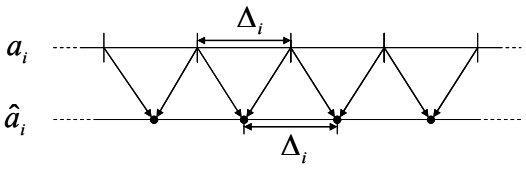


Fig. 2. An illustration of the quantization error when searching for the transformation parameters. Note no matter what value a_i takes within the interval $[\hat{a}_i - \Delta_i/2, \hat{a}_i + \Delta_i/2]$, Equ. (1) will yield the same result: $\hat{a}_i$.

While it is a challenging task to arrive at an analytical expression of $\sigma^2_{MD}(\boldsymbol{x},n)$ from (15) and (19), we can still obtain a numerical result by replacing the integral with summation:

$$\begin{aligned} \sigma^2_{MD}(\boldsymbol{x},n) &\approx \sum_{k_1}\sum_{k_2}\sum_{k_3}\{I_n[\phi(\boldsymbol{x};\boldsymbol{a_k})] - I_n[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})]\}^2 \prod_{i=1}^{3} p_i(k_i\Delta a_i \mid \hat{a}_i)\Delta a_i \\ &= \left(\prod_{i=1}^{3}\frac{\Delta a_i}{\Delta_i}\right)\cdot\sum_{k_1}\sum_{k_2}\sum_{k_3}\{I_n[\phi(\boldsymbol{x};\boldsymbol{a_k})] - I_n[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})]\}^2 \end{aligned}, \tag{20}$$

where $\Delta a_1 \Delta a_2 \Delta a_3$ forms an elementary cube in the space of $\boldsymbol{a}$ as a unit for summation, and $\boldsymbol{a_k} = [k_1\Delta a_1, k_2\Delta a_2, k_3\Delta a_3]^T$. The range of integer k_i in the summation satisfies

$$|k_i\Delta a_i - \hat{a}_i| \le \Delta_i/2,\ i = 1,2,3. \tag{21}$$

After we have now acquired the drift noise power, the final estimate of the measurement noise power can be calculated as:

$$\sigma^2_M(\boldsymbol{x},n) = \sigma^2_{MC} + \eta\cdot\sigma^2_{MD}(\boldsymbol{x},n). \tag{22}$$

where η is a constant larger than 1 which accounts for possible errors involved in the bilinear interpolation. In this paper, we set η to be 2, a value suitable for all the tracking scenarios we have encountered.

From the discussion above, it can be seen that the measurement noise power is heavily dependent on the target appearance (higher density of textures or edges contained in the target appearance results in larger measurement noise power). This is not surprising, because the same amount of deviation of the target location will cause greater appearance errors (and hence severer template damage) to the template

pixels surrounded by more complex target appearance. As a result, template drift is more prone to occur when the target appearance contains more details. In our algorithm, more appearance details lead to higher measurement noise power which precludes the Kalman gain of the appearance filter from getting too large, and consequently template drift is significantly reduced.

It is worth mentioning that the discussion above can be trivially generalized to coordinate transformations containing any number of parameters, not just translation and scaling.

3) Fast Algorithm

The computational complexity of (20) can be reduced if we can acquire the joint posterior distribution, p_u, of the transformed coordinate $\boldsymbol{u} = \phi(\boldsymbol{x};\boldsymbol{a}) = [v, w]^T$ from p_a and ϕ, so that (14) becomes

$$\sigma_{MD}^2(\boldsymbol{x}, n) = \int_{\boldsymbol{u}\in\boldsymbol{\Omega}_{\boldsymbol{u}}} \left[I_n(\boldsymbol{u}) - I_n(\hat{\boldsymbol{u}})\right]^2 p_{\boldsymbol{u}}(\boldsymbol{u} \mid \hat{\boldsymbol{u}}) d\boldsymbol{u}, \tag{23}$$

and (20) becomes

$$\sigma_{MD}^2(\boldsymbol{x}, n) \approx \Delta v \Delta w \sum_{l_1} \sum_{l_2} \left[I_n(l_1\Delta v, l_2\Delta w) - I_n(\hat{v}, \hat{w})\right]^2 p_{\boldsymbol{u}}(l_1\Delta v, l_2\Delta w \mid \hat{v}, \hat{w}). \tag{24}$$

Here, $\hat{\boldsymbol{u}} = \phi(\boldsymbol{x};\hat{\boldsymbol{a}}) = [\hat{v}, \hat{w}]^T$, $\boldsymbol{\Omega}_{\boldsymbol{u}}$ is the region where $\phi(\boldsymbol{x};\boldsymbol{a})$ might be located (as is shown in Fig. 1), and $\Delta v \Delta w$ is an elementary rectangle as a summation unit. Integers l_1 and l_2 take such values that

$$[l_1\Delta v, l_2\Delta w]^T \in \boldsymbol{\Omega}_{\boldsymbol{u}}. \tag{25}$$

By doing so, we can merge multiple terms of different sets of transformation parameters yielding the same mapped coordinate into a single term when calculating the summation in (20), and the three-dimensional summation is therefore reduced to a two-dimensional one. The problem here is how to calculate p_u.

The distribution function of $\boldsymbol{u}$ is given by

$$F_{\boldsymbol{u}}(\boldsymbol{u} \mid \hat{\boldsymbol{u}}) = P\{V \le v, W \le w \mid \hat{v}, \hat{w}\}, \tag{26}$$

where we use capitalized letters to denote random variables. Since the coordinate transformation can be expressed as

$$\boldsymbol{u} = \phi(\boldsymbol{x};\boldsymbol{a}) = \begin{bmatrix} v \\ w \end{bmatrix} = a_1 \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} a_2 \\ a_3 \end{bmatrix}, \tag{27}$$

we rewrite (26) as follows:

$$\begin{aligned} F_{\boldsymbol{u}}(\boldsymbol{u} \mid \hat{\boldsymbol{u}}) &= P\{A_2 \le v - A_1 x, A_3 \le w - A_1 y \mid \hat{a}_1, \hat{a}_2, \hat{a}_3\} \\ &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{v-a_1 x} p_2(a_2 \mid \hat{a}_2, a_1) da_2 \cdot \int_{-\infty}^{w-a_1 y} p_3(a_3 \mid \hat{a}_3, a_1) da_3 \right] p_1(a_1 \mid \hat{a}_1) da_1 . \\ &= \int_{-\infty}^{\infty} \left[\int_{-\infty}^{v-a_1 x} p_2(a_2 \mid \hat{a}_2) da_2 \cdot \int_{-\infty}^{w-a_1 y} p_3(a_3 \mid \hat{a}_3) da_3 \right] p_1(a_1 \mid \hat{a}_1) da_1 \end{aligned} \tag{28}$$

The latter equality holds because of the independence among the transformation parameters.

The joint posterior distribution of $\boldsymbol{u}$ can be yielded by calculating the partial derivatives of (28):

$$p_{\boldsymbol{u}}(\boldsymbol{u} \mid \hat{\boldsymbol{u}}) = \frac{\partial^2 F_{\boldsymbol{u}}(v,w)}{\partial v \partial w} = \int_{-\infty}^{\infty} p_2(v - a_1 x \mid \hat{a}_2) p_3(w - a_1 y \mid \hat{a}_3) p_1(a_1 \mid \hat{a}_1) da_1 . \tag{29}$$

Substituting (19) for the distributions in (29), we have

$$p_{\boldsymbol{u}}(\boldsymbol{u} \mid \hat{\boldsymbol{u}}) = \frac{1}{\Delta_1 \Delta_2 \Delta_3} \int_{B_L}^{B_H} da_1 = \begin{cases} (B_H - B_L)/(\Delta_1 \Delta_2 \Delta_3), & B_H \ge B_L \\ 0, & B_H < B_L \end{cases}, \tag{30}$$

where B_L and B_H can be shown to take the following values:

$$B_L = \max\left\{ \hat{a}_1 - \frac{\Delta_1}{2}, \quad \frac{v - \hat{a}_2}{x} - \frac{\Delta_2}{2|x|}, \quad \frac{w - \hat{a}_3}{y} - \frac{\Delta_3}{2|y|} \right\}, \quad x \ne 0, y \ne 0 , \tag{31}$$

$$B_H = \min\left\{ \hat{a}_1 + \frac{\Delta_1}{2}, \quad \frac{v - \hat{a}_2}{x} + \frac{\Delta_2}{2|x|}, \quad \frac{w - \hat{a}_3}{y} + \frac{\Delta_3}{2|y|} \right\}, \quad x \ne 0, y \ne 0 . \tag{32}$$

When x is zero, the second terms in the braces of (31) and (32) disappear if $|v - \hat{a}_2| \le \Delta_2/2$, otherwise $p_u(\boldsymbol{u} \mid \hat{\boldsymbol{u}}) = 0$. When y is zero, the third terms in the braces of (31) and (32) disappear if $|w - \hat{a}_3| \le \Delta_3/2$, otherwise $p_u(\boldsymbol{u} \mid \hat{\boldsymbol{u}}) = 0$. The derivations of B_L and B_H are omitted here for conciseness. Using (24), (30), (31) and (32), we can evaluate the drift noise power in a computationally efficient manner.

D. Operations for Occluded Template Pixels

The occluded template pixels do not undergo Kalman filtering and their estimated grayscales remain unchanged during the occlusion period. However, for an occluded template pixel, the power of the estimation error σ_E^2 might increase with time due to the change of the target appearance. As will be seen in Section IV, σ_E^2 play an important role in analyzing occlusion situation. In addition, a proper value of σ_E^2 is also needed when the pixel becomes uncovered again. Consequently, careful evaluation of σ_E^2 for occluded pixels is indispensable.

We approximate σ_E^2 according to the following equation when a pixel is occluded:

$$\sigma_E^2(\boldsymbol{x}, n) = \sigma_E^2(\boldsymbol{x}, n-1) + \left[\frac{1}{L} \sum_{k=r-L+1}^{r} \sigma_S^2(\boldsymbol{x}, k) \right] \cdot \exp[-\lambda(n-r)], \tag{33}$$

where r is the frame index immediately *before* the pixel gets occluded, and the increment of the estimation error power each frame is assumed to be the average of the state transition noise powers over the last L non-occluded frames enveloped by an exponential term with the time constant $1/\lambda$. The purpose of this exponential term is to prevent the estimation error power from becoming too large when the pixel is occluded for a long time. The time constant can be set according to the rigidness of the target: higher rigidness is associated with a smaller time constant (or larger λ), and vice versa.

IV. Content-Adaptive Progressive Occlusion Analysis Algorithm

The overall scheme of the CAPOA algorithm is shown in Fig. 3. The function block in gray is further expanded in Fig. 4. The two figures will be detailed in the subsequent sub-sections. The prototype of the CAPOA algorithm is

$$[U_{out}, M_{out}] = \mathrm{CAPOA}(ROI, T_{ref}, I_{prev}, U_{prev}), \tag{34}$$

where ROI is the region of interest of the current frame under analysis, T_{ref} is the reference target which will be introduced in Section IV-C, I_{prev} is the previous frame, U_{prev} is the previous outlier map, U_{out} is the updated outlier map incorporating the result of the occlusion analysis, and M_{out} is the updated template mask corresponding to the occlusion situation within the ROI of the updated outlier map.

A. Progressive Scanning of the Region of Interest

The boundaries of the region of interest (ROI) are the coordinate-transformed boundaries of the template, so the target is tightly framed by the ROI in most cases. Therefore, we only need to analyze the occlusion situation within the ROI to reduce computation.

In the CAPOA algorithm, occlusion detection is based on image blocks, not individual pixels as is done in [7], [8], [23] and [24], because spatial context plays an important role in deciding whether a target is occluded. This is also how we humans make such decisions. For example, when two faces partly overlap, only by exploiting the differences of spatial structures can we know that an occlusion occurs.

In order to obtain a good trade-off between reliability and resolution, we use a progressive scanning procedure. The ROI undergoes multiple scans. In each new scan, the sizes of the blocks under analysis are halved,

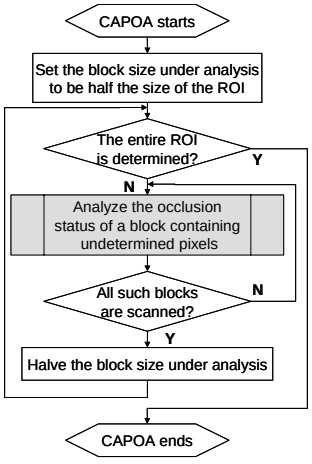


Fig. 3. The flowchart of the overall CAPOA algorithm.

and we only analyze the blocks within which the occlusion situation has not been determined by the previous scans. The progressive scanning terminates when the occlusion situation of the entire ROI is determined (see Fig. 3). Let D_1 and D_2 be the length of the two sides of the ROI, the total number of scans N_S is

$$N_S = \min\left\{\left\lfloor \log_2\left(\min(D_1, D_2)/5\right)\right\rfloor,\ 3\right\}, \tag{35}$$

so that the minimum size of any block under analysis is 5 and the maximum number of scans is 3. That is, the highest spatial resolution is an 8-by-8 division of the target, which is enough in practice. The determination of the occlusion status of a block is described in the sub-sections below.

B. Primary Information Source: Previous Outlier Map

As is mentioned in Section I, the observed target appearance alone cannot give a reliable decision on occlusion situation, because the information of outliers is not encoded there. The prior information regarding outliers is *only* embodied in the non-target region during the initialization. The evolvement of the non-target region offers clues whether it has "encroached" on the target region. Therefore, in order to determine whether a block in the ROI is occluded, we perform backward motion estimation of the block and see whether its corresponding block in the previous frame is within the non-target region. By doing so, the occluding status of the block can be traced down all the way from the first frame, in which the occlusion situation is *a priori* known. The spatiotemporal context around the target region is thus exploited in this process. In order to trace the occlusion statuses of image blocks within the ROI, we generate a binary image called an *outlier map*, which is described in Section II. For each block in the ROI

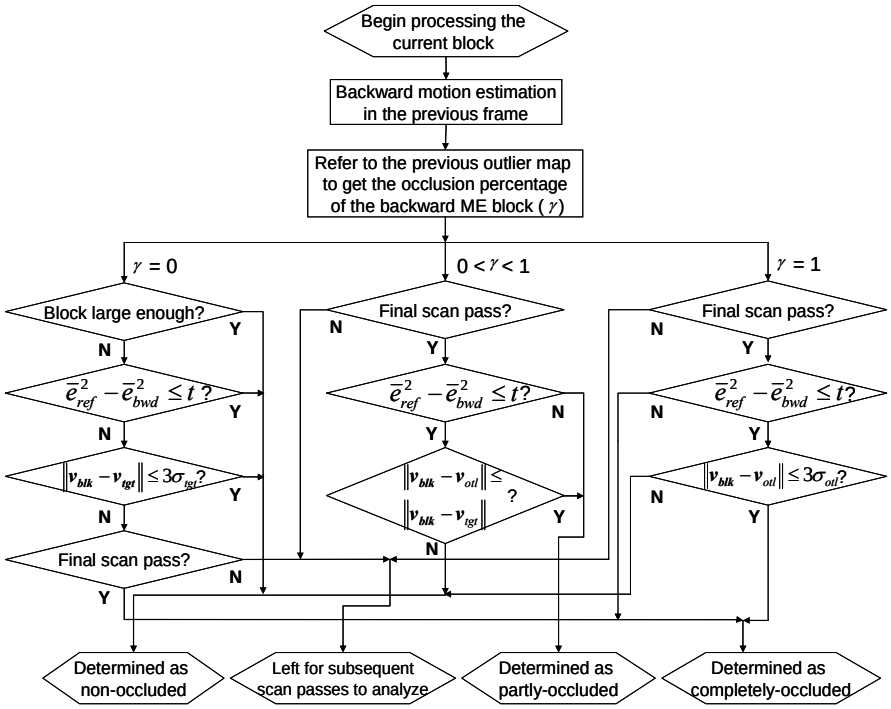


Fig. 4: The flowchart of analyzing the occlusion status of a current block..

(referred to as a *current block*), we perform backward motion estimation in the previous frame, and the motion-estimated block in the previous frame is called a *backward ME block* (ME stands for motion estimation). Theoretically, the occlusion situation within the current block can just be copied from the previous outlier map associated with the corresponding backward ME block. However, if the current block is relatively small or is newly uncovered, the motion estimation would be less reliable and the judgment *solely* based on the previous outlier map is not trustworthy. Therefore, what is derived at this stage is only a *temporary outlier map* which is subject to further scrutiny. The temporary outlier map associated with the current block located at $\boldsymbol{\omega_b}$ can be expressed as follows:

$$\boldsymbol{U}'(\boldsymbol{\omega_b}) = \boldsymbol{U}_{prev}(\tilde{\boldsymbol{\omega}}_{\boldsymbol{b}}) \tag{36}$$

where $\boldsymbol{U}'$ denotes the temporary outlier map, and $\boldsymbol{U}_{prev}(\tilde{\boldsymbol{\omega}}_{\boldsymbol{b}})$ is part of the previous outlier map associated with the corresponding backward ME block located at $\tilde{\boldsymbol{\omega}}_{\boldsymbol{b}}$.

For a current block located at $\boldsymbol{\omega_b}$, let $\gamma(\boldsymbol{\omega_b})$ be the percentage of the occluded pixels in $\boldsymbol{U}'(\boldsymbol{\omega_b})$. Then the current block can be classified into three categories as is illustrated in Fig. 4. Each category has a specific procedure of further check. The idea behind it is that 1) if $\gamma(\boldsymbol{\omega_b})$ is non-zero, the corresponding region should be analyzed by the (smallest) blocks in the *final* scan pass and get double-checked in that pass to discover details and to ensure the detection of small target regions reappearing from behind the other side of an occluder; 2) if $\gamma(\boldsymbol{\omega_b})$ is zero, the block is double-checked in the current scan pass only when it is *not* large enough to yield reliable motion estimation. A block is believed to be large enough when both its dimensions exceed 15 pixels. All the blocks that need to be double-checked in the *current* scan pass is referred to as *uncertain blocks*, and we resort to further information to determine their occlusion statuses.

C. Second Information Source: Reference Target

The reference target is essentially a scaled version of the template and is updated through incremental interpolation and filtering *at the end of* processing each frame (see the normal mode of Table I): if the scale of the target is found to have changed in the current frame, the reference target is firstly interpolated to fit the size of the target, and then renewed to incorporate the variations of the target appearance. The renewed value of a certain pixel $\boldsymbol{x}$ in the interpolated reference target is calculated as

$$T_{ref}(\boldsymbol{x}, n+1) = T'_{ref}(\boldsymbol{x}, n) + G(\boldsymbol{x}_1, n) \cdot \{I_n(\boldsymbol{x}_2) - T'_{ref}(\boldsymbol{x}, n)\} \cdot [1 - U_n(\boldsymbol{x}_2)], \tag{37}$$

where $T_{ref}(\boldsymbol{x}, n+1)$ represents the pixel value of the reference target that will be used by the CAPOA at

frame $n+1$, $T'_{ref}(\boldsymbol{x},n)$ denotes the pixel value of the *interpolated* reference target of frame *n*, $G(\boldsymbol{x}_1,n)$ is the Kalman gain of the template pixel located at the same relative position and is obtained during the DIMKAF of the template, $I_n(\boldsymbol{x}_2)$ represents the corresponding image pixel in frame *n*, and $U_n(\boldsymbol{x}_2)$ is the corresponding pixel value of the final outlier map of frame *n*. (In this sub-section, "corresponding pixels" refer to the two pixels that have the same relative position within the target but reside in different images). Since the change in the scale of the target is very small over one frame interval, the incremental adjustment of the scale of the reference target enables much more accurate interpolation than directly interpolating from the template. Unlike the template, the reference target is the same size as the true target and hence contains more details than the template does. We therefore choose the reference target to help further determine the occlusion status of an uncertain block.

When an uncertain block belongs to the target, it must resemble the corresponding part of the reference target. In light of this, we search for the best match of the block around its corresponding position in the reference target and calculate the matching error measured by mean squared error (MSE). Then we compare this MSE with the matching error of the block in the previous frame (obtained through backward motion estimation). For the simplicity of notation, we denote the former error as $\bar{e}^2_{ref}$ and the latter one as $\bar{e}^2_{bwd}$. For an uncertain block located at $\boldsymbol{\omega_b}$ whose backward ME block is non-occluded, it can be determined as truly non-occluded [i.e. $\boldsymbol{U}_{out}(\boldsymbol{\omega_b})=\boldsymbol{U}'(\boldsymbol{\omega_b})$] if $\bar{e}^2_{ref}$ is smaller or only slightly larger than $\bar{e}^2_{bwd}$. Similarly, if the backward ME block of an uncertain block is partly or completely occluded, $\boldsymbol{U}_{out}(\boldsymbol{\omega_b})=\boldsymbol{U}'(\boldsymbol{\omega_b})$ holds if $\bar{e}^2_{ref}$ is significantly over $\bar{e}^2_{bwd}$. They are the decision criteria when using the second information source. The problem here is how to adaptively set the threshold of $\bar{e}^2_{ref} - \bar{e}^2_{bwd}$ involved in the decision criteria for individual blocks. This threshold is denoted as *t* in Fig. 4.

In fact, we can calculate the expected values of the two matching errors and therefore the expected value of $\bar{e}^2_{ref} - \bar{e}^2_{bwd}$ when the current block is non-occluded both in the current and the previous frame, and then use this value as the benchmark for the threshold. We denote the value of a *single* target pixel in frame *n* as $Y_F(n)$, and the relation between the values of the corresponding target pixels in frame *n*–1 and frame *n* is

$$Y_F(n) = Y_F(n-1) + \varepsilon_S(n-1) + \varepsilon'_M, \tag{38}$$

where $\varepsilon_S(n-1)$ denotes the variation of the pixel value from frame *n*–1 to frame *n* caused by the target itself, and ε'_M represents the measurement noise associated with the backward motion estimation. Let

$\mathrm{E}\{e_{bwd}^2\}$ denote the expected matching error of the pixel in the backward motion estimation. From (38), it can be expressed as

$$\mathrm{E}\{e_{bwd}^2\} = \mathrm{E}\{[Y_F(n) - Y_F(n-1)]^2\} = \sigma_S^2(n-1) + \sigma_M'^2, \tag{39}$$

where σ_S^2 and $\sigma_M'^2$ are the powers of ε_S and ε_M', respectively.

Now we calculate the expected matching error of the pixel in the reference target. The relation between the values of the corresponding target pixels in the reference target and frame *n* is

$$Y_F(n) = Y_T(n-1) + \varepsilon_E''(n-1) + \varepsilon_S(n-1) + \varepsilon_M'', \tag{40}$$

where $Y_T(n-1)$ denotes the value of the reference target pixel after the reference target has been updated at the end of processing frame *n*–1, and its estimation error is $\varepsilon_E''(n-1)$. ε_M'' is the measurement noise associated with the process of performing block matching in the reference target. As a result, the expected matching error of the pixel in the reference target $\mathrm{E}\{e_{ref}^2\}$ is

$$\mathrm{E}\{e_{ref}^2\} = \mathrm{E}\{[Y_F(n) - Y_T(n-1)]^2\} = \sigma_E''^2(n-1) + \sigma_S^2(n-1) + \sigma_M''^2, \tag{41}$$

where $\sigma_E''^2$ and $\sigma_M''^2$ are the powers of ε_E'' and ε_M'', respectively.

From (39) and (41), the expected difference between the two matching errors is

$$\sigma_D^2 = \mathrm{E}\{e_{ref}^2 - e_{bwd}^2\} = \sigma_E''^2(n-1) + \sigma_M''^2 - \sigma_M'^2. \tag{42}$$

where σ_D^2 is the expected error difference associated with the target pixel. As the context around the target pixel in frame *n*–1 is very close to that in the reference target, it is reasonable to simplify (42) by assuming $\sigma_M'^2 = \sigma_M''^2$, and (42) becomes

$$\sigma_D^2 = \sigma_E''^2(n-1). \tag{43}$$

Since $\sigma_E''^2(n-1)$ is the power of the estimation error of the pixel value in the reference target (which is a scaled version of the template), it can be effectively approximated by the power of the estimation error of the corresponding pixel in the template, that is,

$$\sigma_D^2(\boldsymbol{x}) = \sigma_E''^2(\boldsymbol{x}, n-1) = \sigma_E^2(\boldsymbol{x}_t, n-1), \tag{44}$$

where $\boldsymbol{x}$ and $\boldsymbol{x}_t$ represent the coordinates of the corresponding pixels located in the reference target and the template, respectively. σ_E^2 is the power of the estimation error of the template pixel and has been evaluated during the DIMKAF of the template when processing the previous frame.

After we acquire the expected difference of the two matching errors for a *single* target pixel, the expected difference of the two matching errors for an uncertain block in the ROI can be estimated by averaging over all the pixels located in the block:

$$\mathrm{E}\left\{\bar{e}_{ref}^2 - \bar{e}_{bwd}^2\right\} = \frac{1}{N_{BT}} \sum_{\boldsymbol{x}_t \in \boldsymbol{\Omega}_{BT}} \sigma_E^2(\boldsymbol{x}_t, n-1), \tag{45}$$

where $\boldsymbol{\Omega}_{BT}$ represents the corresponding block region in the template and N_{BT} is the number of pixels it contains. Equ. (44) has been considered when we derive (45).

Combining with the result given by the primary information source, we set the value of t for an uncertain block located at $\boldsymbol{\omega}_b$ as follows:

$$t(\boldsymbol{\omega}_b) = \mathrm{E}\left\{\bar{e}_{ref}^2 - \bar{e}_{bwd}^2\right\} \cdot \left[3 - 2\gamma(\boldsymbol{\omega}_b)\right] = \frac{3 - 2\gamma(\boldsymbol{\omega}_b)}{N_{BT}} \sum_{\boldsymbol{x}_t \in \boldsymbol{\Omega}_{BT}} \sigma_E^2(\boldsymbol{x}_t, n-1). \tag{46}$$

Smaller $\gamma(\boldsymbol{\omega}_b)$ implies higher probability of the current block belonging to the target, and therefore a larger threshold should be set. From (46), it can be seen that the value of t in Fig. 4 is adaptive to the contents of individual blocks: higher estimation error or lower occlusion percentage increases the value of t.

D. Third Information Source: Motion Constraint

For the uncertain blocks that fail to meet the decision criteria in last sub-section, we exploit motion constraint to further check their occlusion statuses. Blocks that belong to the target (occluder) should bear similar motion to the target (occluder). By virtue of this fact, we compare the motion vector of an uncertain block with the motion vector of the target (occluder) to look for additional cues that help decide on the occlusion status of the block.

The decision criteria for the third information source are illustrated in Fig. 4 and summarized as follows:

1) If $\gamma(\boldsymbol{\omega}_b) = 0$,

$$\boldsymbol{U}_{out}(\boldsymbol{\omega}_b) = \begin{cases} \boldsymbol{U}'(\boldsymbol{\omega}_b) & , \quad \left\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{tgt}\right\| \le 3\sigma_{tgt} \\ undetermined & , \quad \left\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{tgt}\right\| > 3\sigma_{tgt} \ \ and \ \ F_P = 0 \\ \boldsymbol{1} & , \quad \left\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{tgt}\right\| > 3\sigma_{tgt} \ \ and \ \ F_P = 1 \end{cases}. \tag{47}$$

2) If $0 < \gamma(\boldsymbol{\omega}_b) < 1$,

$$\boldsymbol{U}_{out}(\boldsymbol{\omega}_b) = \begin{cases} \boldsymbol{U}'(\boldsymbol{\omega}_b) & , \quad \left\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{otl}\right\| \le \left\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{tgt}\right\| \\ \boldsymbol{0} & , \quad \left\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{otl}\right\| > \left\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{tgt}\right\| \end{cases}. \tag{48}$$

3) If $\gamma(\boldsymbol{\omega}_b) = 1$,

$$\boldsymbol{U}_{out}(\boldsymbol{\omega}_b) = \begin{cases} \boldsymbol{U}'(\boldsymbol{\omega}_b) & , \quad \left\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{otl}\right\| \le 3\sigma_{otl} \\ \boldsymbol{0} & , \quad \left\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{otl}\right\| > 3\sigma_{otl} \end{cases}. \tag{49}$$

Here, $\boldsymbol{v}_{blk}$ denotes the motion vector of the uncertain block and is obtained through the backward motion estimation in Section IV-B. $\boldsymbol{v}_{tgt}$ ($\boldsymbol{v}_{otl}$)is the motion vector of the target (occluder) which is estimated by

averaging the motion vectors of all the pixels that have already been decided as belonging to the target (occluder). σ_{tgt} (σ_{otl}) is the root mean square (RMS) of the Euclidean distances from the motion vector of every target (occluded) pixel to the mean motion vector of the target (occluder). If no pixel has been determined as a part of the target yet, the Kalman filtered target speed [21] (see the bottom line of the pseudo codes in the normal mode of Table I) is used as $\boldsymbol{v}_{tgt}$, and σ_{tgt} is set to be 1. If no pixel has been determined to be a part of the occluder yet, the speed of the occluder in the last frame serves as $\boldsymbol{v}_{otl}$, and σ_{otl} is set to be 1. F_P = 1 if the current scan pass is the final one, otherwise F_P = 0. '*undetermined*' means the occlusion situation of the ROI region associated with the current block is left for subsequent scan passes to analyze. '***1***' ('***0***') denotes a matrix of ones (zeros) which indicates that the current block is determined as completely occluded (non-occluded).

After the occlusion situation of the entire ROI is determined (or equivalently, the outlier map within the ROI is updated), the CAPOA is completed. The pixel values of the outlier map outside the ROI are all ones. Each time the outlier map is updated, the template mask is also renewed by sampling from the outlier map as follows:

$$M_{out}(\boldsymbol{x}) = 1 - U_{out}\{\text{round}[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})]\}. \tag{50}$$

Here, round[·] is the operation that rounds the elements of a vector to their nearest integers. As an additional protection for the integrity of the template, we further perform image erosion operation [14] on M_{out} where the erosion width is 2 pixels.

To sum up, the CAPOA algorithm exploits the spatiotemporal context (i.e. associates image blocks between the current and the previous frame) to acquire a primary belief of the occlusion situation. This belief is further double-checked by the reference template and motion constraints. This strategy significantly promotes the discriminating power against outliers. More importantly, even if errors occur to the outlier map, they can be corrected by the CAPOA algorithm in time before they ever have a chance to undermine tracking performance. As a result, those errors seldom propagate away. We will verify this claim in the experimental section.

V. Target Location Rectification Using Variant-Mask Template Matching

A. Purpose of the Target Location Rectification

After a new frame comes in, the template mask used by the first masked template matching is in

accordance with the occlusion situation of the previous frame rather than the current frame. As a result, the target location yielded by the first masked template matching is often inaccurate, especially when the occlusion percentage of the current frame is higher than in the previous frame. Accumulation of these errors will ultimately result in tracking failure. This is one of the main reasons why [7], [8] and [23] turn off tracking when more than 30% of the target is occluded. In order to remedy this problem, we propose the Variant-Mask Template Matching (VMTM) operation to rectify the target location.

B. *Variant-Mask Template Matching*

As the target location yielded by the first template matching is inaccurate, part of the target might stay *outside* the ROI (as is depicted by the shadowed area upon "A" in the lower left image of Fig. 5). Therefore, the outlier map generated after the first analysis of the occlusion situation might be incorrect, and is referred to as a *preliminary outlier map* (see Table I). Nevertheless, the portion of the target that lies *within* the ROI is still correctly identified. We can therefore utilize this information to align the target to its precise location using the VMTM operation.

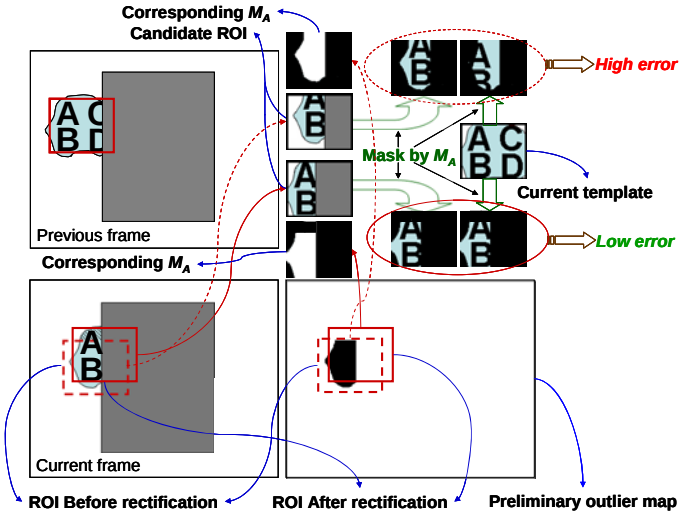


Fig. 5. An illustration of the VMTM operation. Letters "A", "B", "C" and "D" are marked on the target to indicate different portions of it. In the current frame, the ROI found by (1) is enclosed by a dashed red box, and the ROI found by (52) is enclosed by a solid red box. Note how M_A changes with the location of the candidate ROI and yields different errors between the masked candidate ROI and the masked template. The error reaches its minimum when the candidate ROI is just the region that the target occupies.

The VMTM operation is based on (1), but instead of being constant throughout the search, the template mask is a function of the transformation parameters under test by relating with the preliminary outlier map through

$$M_A(\boldsymbol{x};\boldsymbol{\alpha}) = 1 - U_{prlm}\{\text{round}[\phi(\boldsymbol{x};\boldsymbol{\alpha})]\}. \tag{51}$$

Here, $\boldsymbol{\alpha}$ is the transformation parameter vector under test, M_A is the dynamic template mask which varies with $\boldsymbol{a}$, and U_{prlm} is the preliminary outlier map. Equ. (1) is therefore modified as

$$\hat{\boldsymbol{a}}_A = \arg\min_{\boldsymbol{a}} \frac{1}{\text{sum}(M_A)} \sum_{\boldsymbol{x}\in\boldsymbol{\Omega}_T} \left| I_n[\phi(\boldsymbol{x};\boldsymbol{a})] - \hat{T}(\boldsymbol{x}) \right| \cdot M_A(\boldsymbol{x};\boldsymbol{\alpha}), \tag{52}$$

where $\hat{\boldsymbol{\alpha}}_A$ is the rectified transformation parameter vector which defines the rectified ROI (ROI_2 in Table I).

The underlying mechanism of the VMTM is that in the parameter search performed by (52), the unmasked part of the template and the unmasked part of the candidate ROI are always dissimilar *unless* the candidate ROI is located exactly where the target is. This fact is illustrated in Fig. 5. For the sake of stability, only translational transformation parameters are involved in the VMTM.

VI. Handling Complete Occlusion

The tracking algorithm discussed above can track the target accurately even when up to 85% of the target is occluded. When the occlusion percentage is even higher, however, the tracking becomes unstable, because only extremely little information of the target can be utilized to perform tracking. We therefore define 85% as the threshold separating partial and complete occlusions. In the case of the complete occlusions, our proposed solution switches from the normal mode to the complete-occlusion mode. The flow of the complete-occlusion mode is shown in Table I.

The most challenging task of handling a complete occlusion is how to effectively detect the end of it without any restriction on its maximal duration. We propose an algorithm that successfully solves this problem by verifying whether the best match in *every K*-frame period (referred to as an inspection period) is truly the target. The end of the complete occlusion is declared only when the best match of a certain inspection period passes the authentication. As a result, this algorithm is named the local best match authentication (LBMA) algorithm. Attributed to the mechanism of authentication, K can be set rather small (in our algorithm it is set to be 3) even when the duration of the complete occlusion well exceeds K frames. A smaller K has the desirable benefits of capturing the reemergence of the target in time to prevent possible interferences and reducing unnecessary computation.

In the LBMA algorithm, the coordinate transformation parameters and the frame index of the local best match is acquired by

$$(\boldsymbol{a}_m, n_m) = \arg\min_{\boldsymbol{a},n} \frac{1}{N} \sum_{\boldsymbol{x}\in\boldsymbol{\Omega}_T} \left| I_n[\phi(\boldsymbol{x};\boldsymbol{a})] - \hat{T}(\boldsymbol{x}) \right|, \tag{53}$$

where $\boldsymbol{a}_m$ and n_m are the transformation parameters and the frame index of the local best match, and N is the number of pixels in the template. The searching range of $\boldsymbol{a}$ is enlarged compared with (1) because the target location might be farther away from the initial searching location in the complete-occlusion mode. The initial searching location is estimated by assuming that the target moves at a constant velocity during the complete occlusion. The velocity is estimated as the Kalman filtered target speed in the normal mode.

By doing so, we raise the chance of recapturing the target as soon as it reemerges from behind the occluder. Of course, if the motion of the target deviates too much from the constant-velocity model, it is impossible to recapture it unless we search the entire frame. In (53), the frame index n is searched within the current inspection period, that is,

$$n \in \{n_c + (k-1)K + i\}_{i=1}^{K}. \tag{54}$$

Here, n_c denotes the frame index when our proposed solution enters the complete occlusion mode, and k indicates that the current inspection period is the k-th period.

The local best match T_{LB} can be expressed as

$$T_{LB}(\boldsymbol{x}) = I_{n_m}[\phi(\boldsymbol{x}; \boldsymbol{a}_m)]. \tag{55}$$

Let e_t be the matching error between T_{LB} and the template, that is,

$$e_t = \frac{1}{N} \sum_{\boldsymbol{x} \in \boldsymbol{\Omega}_T} \left| T_{LB}(\boldsymbol{x}) - \hat{T}(\boldsymbol{x}) \right|. \tag{56}$$

In order to verify the authenticity of T_{LB} indeed being the target, we match T_{LB} (not $\hat{T}$) backwards to the previous P frames and calculate the mean matching error $\bar{e}_b$ as follows:

$$\bar{e}_b = \frac{1}{P} \sum_{n=n_m-P}^{n_m-1} \left\{ \min_{\boldsymbol{a}} \frac{1}{N} \sum_{\boldsymbol{x} \in \boldsymbol{\Omega}_T} \left| I_n[\phi(\boldsymbol{x}; \boldsymbol{a})] - T_{LB}(\boldsymbol{x}) \right| \right\}. \tag{57}$$

The value of P is determined according to

$$P = \min\{n_m - n_c,\ 3K\}. \tag{58}$$

Whether the local best match is truly the target can be judged by comparing e_t with $\bar{e}_b$. On one hand, if the target is still under occlusion, the local best match actually falls on the occluder or other clusters which are most probably in the foreground in the previous P frames. Therefore, T_{LB} can always find good matches in the previous P frames. Meanwhile, as T_{LB} is *not* the target, its match with the template is much worse. As a result, when the complete occlusion is not over, e_t is always greater than $\bar{e}_b$.

On the other hand, if the local best match is indeed the target, we can infer that the target must be under occlusion in most of the previous P frames, as the inspection period is only K frames. Consequently, T_{LB} *cannot* find a good match in most of the previous P frames, resulting in a significant increase in $\bar{e}_b$. Meanwhile, e_t is rather low as long as the appearance of the target does not change too much during the complete occlusion. Therefore, $\bar{e}_b$ would exceed e_t when the local best match in the current inspection period is truly the target.

By virtue of the discussion above, T_{LB} is authenticated to be truly the target when the following inequality holds:

$$e_t(k) - \bar{e}_b(k) < \delta \cdot \log[n_m(k) - n_c], \tag{59}$$

where $\bar{e}_b(k)$ and $e_t(k)$ denote the two matching errors in the k-th inspection period, $n_m(k)$ is the frame index of the local best match in the k-th inspection period, and δ is a small positive number which accounts for possible appearance change of the target during the complete occlusion. It is adjusted according to the expected intensity of the appearance change: more intensive variation of the target appearance requires larger δ.

The end of the complete occlusion is declared after T_{LB} passes the authentication. At this time, a small part of the target might still be under occlusion. Therefore, it is necessary to insert a *transition period* before switching to the normal mode. The purpose of this period is to wait until the target gets fully disoccluded. Setting the length of the transition period as 5 frames is suitable for most tracking scenarios. In the transition period, we perform *non-masked* template matching to acquire the location of the target, because 1) we do not know the current occlusion situation yet and 2) template mask is not crucial here because the occlusion percentage is generally very small when the LBMA algorithm declares the end of a complete occlusion. At the last frame of the transition period, we reinitialize the template mask as an array of ones and the outlier map as a binary array in which only the pixel values within the current ROI are zeros. Then we return to the normal mode when the next frame arrives.

VII. Experimental Results

In this section, we test the robustness and accuracy of our proposed object tracking solution. The experimental results concerning computational complexity are also presented. We perform experiments on real-world video sequences containing a wide range of tracking scenarios. A total number of 66 sequences are under test, half of which are downloaded from two standard datasets and the other half are taken by us using a SONY EVI-D100P video camera. The standard datasets are available at CAVIAR (http://homepages.inf.ed.ac.uk/rbf/CAVIAR) and Birchfield's head tracking sequences with occlusions (http://www.ces.clemson.edu/~stb/research/headtracker/seq). As a complement to the experiments on the real-world sequences, we also use synthetic test sequences where target motion is precisely controlled and ground-truth data is accurately known, so that tracking performances can be analyzed quantitatively. In

TABLE II
The Numbers of Successfully-Tracked Sequences for Different Algorithms under Various Tracking Scenarios

Scenarios	RAF[8]	WSL[24]	MS[28]	ABM[12]	P-VMTM	P-AdpThd	P
SP [15(10+5)]	12(9+3)	9(7+2)	7(5+2)	10(7+3)	13(10+3)	9(7+2)	14(10+4)
LP [14(8+6)]	0(0+0)	3(2+1)	4(3+1)	5(3+2)	8(4+4)	7(5+2)	13(7+6)
SC [12(8+4)]	5(4+1)	0(0+0)	8(4+4)	0(0+0)	5(3+2)	6(4+2)	10(7+3)
LC [9(6+3)]	0(0+0)	0(0+0)	0(0+0)	0(0+0)	3(3+0)	2(2+0)	7(5+2)
NO[16(1+15)]	11(1+10)	11(1+10)	7(0+7)	10(1+9)	11(1+10)	10(1+9)	12(1+11)
Total [66(33+33)]	28(14+14)	23(10+13)	26(12+14)	25(11+14)	40(21+19)	34(19+15)	56(30+26)

SP: short-term partial occlusion; LP: long-term partial occlusion; SC: short-term complete occlusion; LC: long-term complete occlusion; NO: no occlusion. P: our proposed solution; P-VMTM: our proposed solution without VMTM; P-AdpThd: our proposed solution without adaptive threshold. The convention of the numbers are [all(us+std)], where "us" means the number of sequences shot by us, "std" means the number of sequences from standard datasets, and "all" means the total number.

order to allow for a fair comparison with other algorithms which do not necessarily have the prior knowledge of the camera noise, we set the camera noise power to be zero in our DIMKAF.

A. Robustness of Object Tracking

In our proposed tracking solution, the only parameters that need to be manually adjusted are the λ in (33) and the δ in (59). Here we set $\lambda = 0.3$ and $\delta = 2.7$ throughout our experiments.

In the 66 real-world video sequences, 50 contain various types of occlusions: short-term partial, long-term partial, short-term complete and long-term complete ones. The other 16, which do not involve occlusions, are also included for completeness. The 50 sequences containing occlusions are further classified into four categories associated with the four types of occlusions. Our proposed solution is compared with some state-of-the-art tracking algorithms including Robust Appearance Filter (or RAF) [8], Adaptive Block Matching (or ABM) [12], WSL Appearance Model (or WSL) [24], and Mean Shift (or MS) [28]. Our (modified) solutions with some key features off are also tested to examine the role those features take. They will be discussed later. In order to facilitate comparison, pixel grayscale is used as the target feature for tracking, and the search for target location is conducted within a searching radius of 30 pixels in an exhaustive manner so that the interference from local extrema can be eliminated. The tracking results are summarized in Table II, where the numbers of successfully-tracked sequences for different algorithms under each type of tracking scenarios are listed. The overall statistic is also given in the bottom row of Table II.

In the experiments, it is observed that RAF performs well when there is no occlusion or only short-term partial occlusions occur. However, RAF is not effective in handling long-term occlusions, because the outliers penetrate its appearance filter when an occlusion lasts relatively long. The *S* component of WSL is rather robust against outliers and it makes much contribution to stabilizing the tracking. Nevertheless, we observe that the WSL-based tracker is still frequently distracted by occluders. Most outliers are found to be

modeled by the *W* component rather than the *L* component. As the *W* component also participates in seeking the target location, it drags the WSL-based tracker away. MS tracks the histogram of a target and is therefore robust when the histogram of the non-occluded part of the target is closer to the model histogram than that of the occluder. When this condition is not satisfied, MS performs poorly. In addition, MS tracking is less stable than the other approaches, because histogram alone is not discriminating enough. ABM is more powerful in detecting occluders than the aforementioned algorithms, but it fails in many sequences as a result of error propagation in the outlier map and being ineffective in identifying target regions reemerging from behind the other side of occluders. Our proposed solution significantly outperforms the other algorithms in each type of tracking scenarios. We observe that targets and outliers are always effectively distinguished apart by our algorithm, and the integrity of template is rarely undermined.

In order to provide an intuitive impression of the performance of our proposed solution, we display the tracking process of typical sequences for each type of occlusions. For partial occlusions, we also show the tracking process of the WSL-based tracker [24] as a comparison because it employs a sophisticated appearance model. For short-term complete occlusions, the performance of the RAF-based tracker [8] is displayed for comparison because it also has a mechanism to handle this type of occlusions. For long-term complete occlusions, we do not show the tracking process of any other tracker, because none of them could deal with such occlusions.

(a_1) (a_2) (a_3) (a_4)

(b_1) (b_2) (b_3) (b_4)

Fig. 6. Illustration of the tracking process of our proposed tracker and the WSL-based tracker under a short-term partial occlusion at frames 409, 457, 466, and 500. The sequence under test is *seq_mb*. The tracked target (ROI) is indicated by a white rectangle with a cross at the center. (a_1)-(a_4) The tracking process of our proposed tracker, where the lower right corner of each image is overlapped from left to right by the current Kalman gain, the current template mask and the current template. The Kalman gain is displayed on a scale of 0 to 1, where brighter pixels indicate higher Kalman gains. In the template mask, the masked portion is indicated in black. (b_1)-(b_4) The tracking process of the WSL-based tracker, where the lower right corner of each image is overlapped from left to right by the mixing probabilities of the *W*, *S*, and *L* components, and the means of the *W* and *S* components.

The tracking process under a short-term partial occlusion is displayed in Fig. 6. The sequence under test, named *seq_mb*, is taken from Birchfield's head tracking sequences (the second standard dataset mentioned above). The boundaries of the original frames are padded so as to overlap the sub-images in the lower right corner. In this sequence, the girl's face (selected as the target) is partially occluded by a similar occluder (a man's face). Our proposed tracker stays firm on the target throughout the sequence. Note the sub-images of Kalman gain overlapped in the lower right corner (the leftmost sub-image). The Kalman gains are low in regions packed with features, because regions with stronger fluctuations of grayscale contribute more to

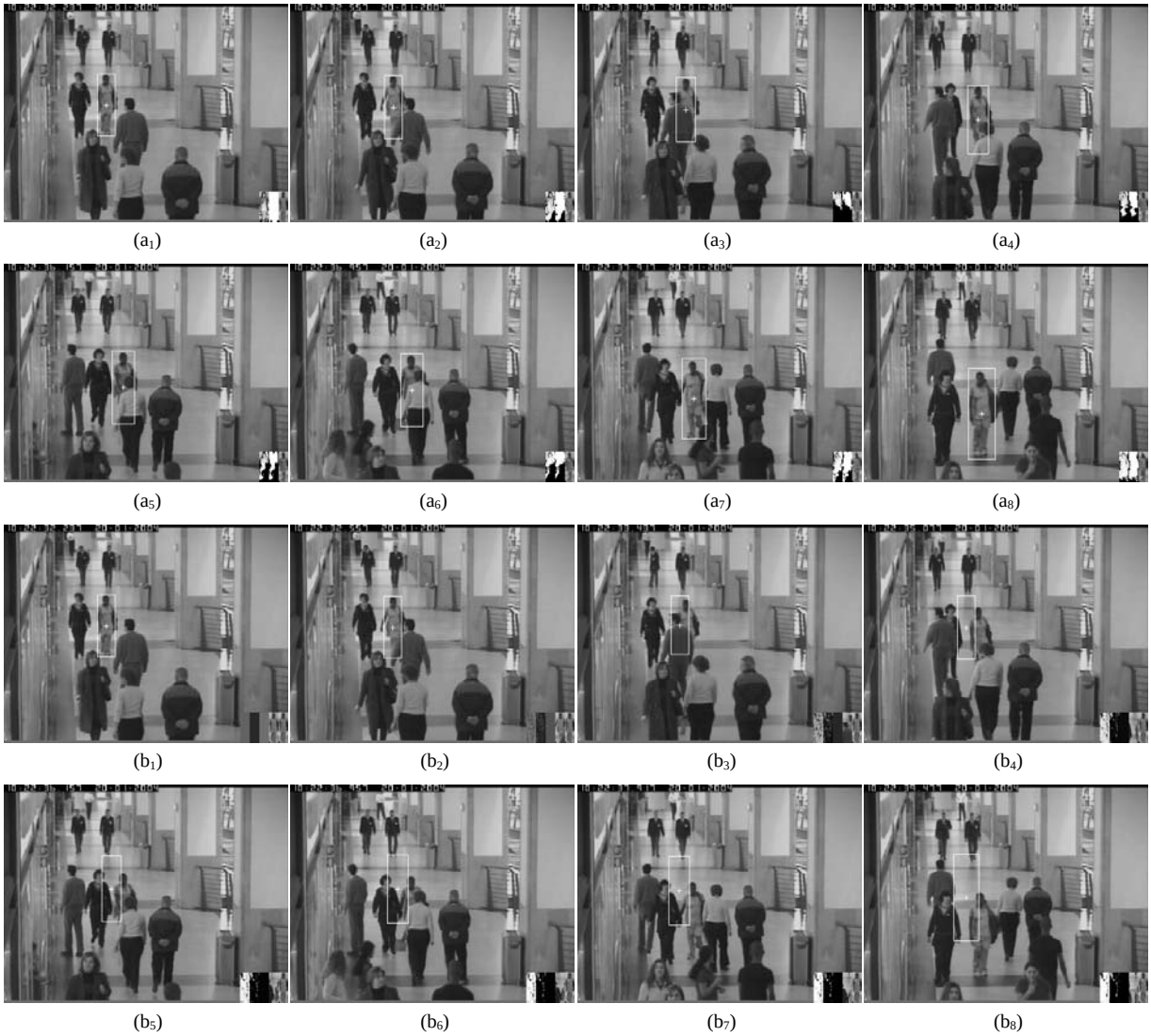


Fig. 7. Illustration of the tracking process of our proposed tracker and the WSL-based tracker under a long-term partial occlusion at frames 202, 210, 232, 273, 300, 320, 344, and 383. The sequence under test is *OneStopMoveEnter1cor*. The tracked target (ROI) is indicated by a white rectangle with a cross at the center. (a_1)-(a_8) The tracking process of our proposed tracker, where the lower right corner of each image is overlapped from left to right by the current Kalman gain, the current template mask and the current template. The Kalman gain is displayed on a scale of 0 to 1, where brighter pixels indicate higher Kalman gains. In the template mask, the masked portion is indicated in black. (b_1)-(b_8) The tracking process of the WSL-based tracker, where the lower right corner of each image is overlapped from left to right by the mixing probabilities of the *W*, *S*, and *L* components, and the means of the *W* and *S* components.

template drift than plain regions do. Also note the overlapped template mask (the middle sub-image). The analysis result of the occlusion situation is never completely right (see Fig. 6-a_3). However, errors are corrected very soon and seldom accumulate or propagate away (see Fig. 6-a_4). We would discuss this feature in more detail later. The WSL-based tracker also performs well. However, instead of being modeled by the *L* component (which does not participate in tracking), the occluder is regarded by the tracker as the *W* component. As a consequence, the tracker is biased away by the occluder (see Fig. 6-b_3), but it manages to recover in this sequence.

Fig. 8. Illustration of the tracking process of our proposed tracker and the RAF-based tracker under a short-term complete occlusion at frames 53, 64, 68, 71, 74, 77, 82, and 90. The sequence under test is shot by us. The tracked target (ROI) is indicated by a white rectangle with a cross at the center. (a_1)-(a_8) The tracking process of our proposed tracker, where the lower right corner of each image is overlapped from left to right by the current Kalman gain, the current template mask and the current template. The Kalman gain is displayed on a scale of 0 to 1, where brighter pixels indicate higher Kalman gains. In the template mask, the masked portion is indicated in black. (b_1)-(b_8) The tracking process of the RAF-based tracker, where the lower right corner of each image is overlapped from left to right by the current outlier situations of the template and the current template. Outlier pixels are indicated in black.

Fig. 7 shows the tracking process under a long-term partial occlusion, where the sequence under test is called *OneStopMoveEnter1cor* from CAVIAR (the first standard database mentioned above). The target in this experiment is a woman in white (see Fig. 7-a_1). She is firstly occluded by a man, which immediately followed by the occlusion from another woman (also in white). The entire occlusion lasts for nearly 50 frames. Our proposed tracker effectively protects the template from the occluders and successfully tracks the target all along. The WSL-based tracker loses the target during the first occlusion (see Fig. 7-b_3) and fails to lock back on the target again this time.

Tracking under a short-term complete occlusion is illustrated in Fig. 8. The sequence shot by us includes a girl whose head is completely occluded by another girl for about 10 frames. The occlusion percentage exceeds 85% at frame 65, where our tracker enters the complete-occlusion mode. Then it performs the LBMA algorithm on inspection periods whose sizes are 3 frames. At the end of each inspection period (i.e. at frames 68, 71, 74, and 77), it checks the authenticity of the local best match within the inspection period. Although some of the first three local best matches are quite similar to the target, they fail the authentication and the tracker remains in the complete-occlusion mode. The fourth local best match passes the authentication, and the tracker returns to the normal mode at frame 82 after the transition period. The values of e_t, $\bar{e}_b$ and $\bar{e}_b + \delta \cdot \log(n_m - n_c)$ [see (59)] over the entire process are plotted in Fig. 9-a. By contrast, the occluder successfully fools the RAF-based tracker which mistakenly takes the other girl's head as the newly-disoccluded target.

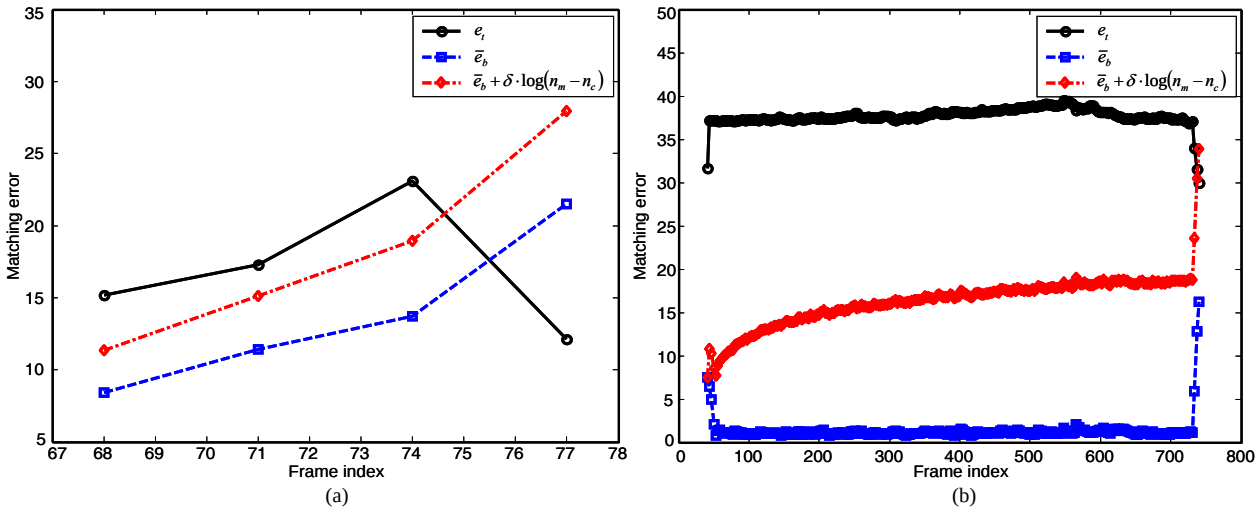


Fig.9. Plots of the values of some key variables involved in the LBMA algorithm during two complete occlusions. The variables are calculated every 3 frames at the end of each inspection period. Please refer to (59) for the meanings of the variables. (a) Plot for the short-term complete occlusion displayed in Fig. 8. (b) Plot for the *second* long-term complete occlusion displayed in Fig. 10.

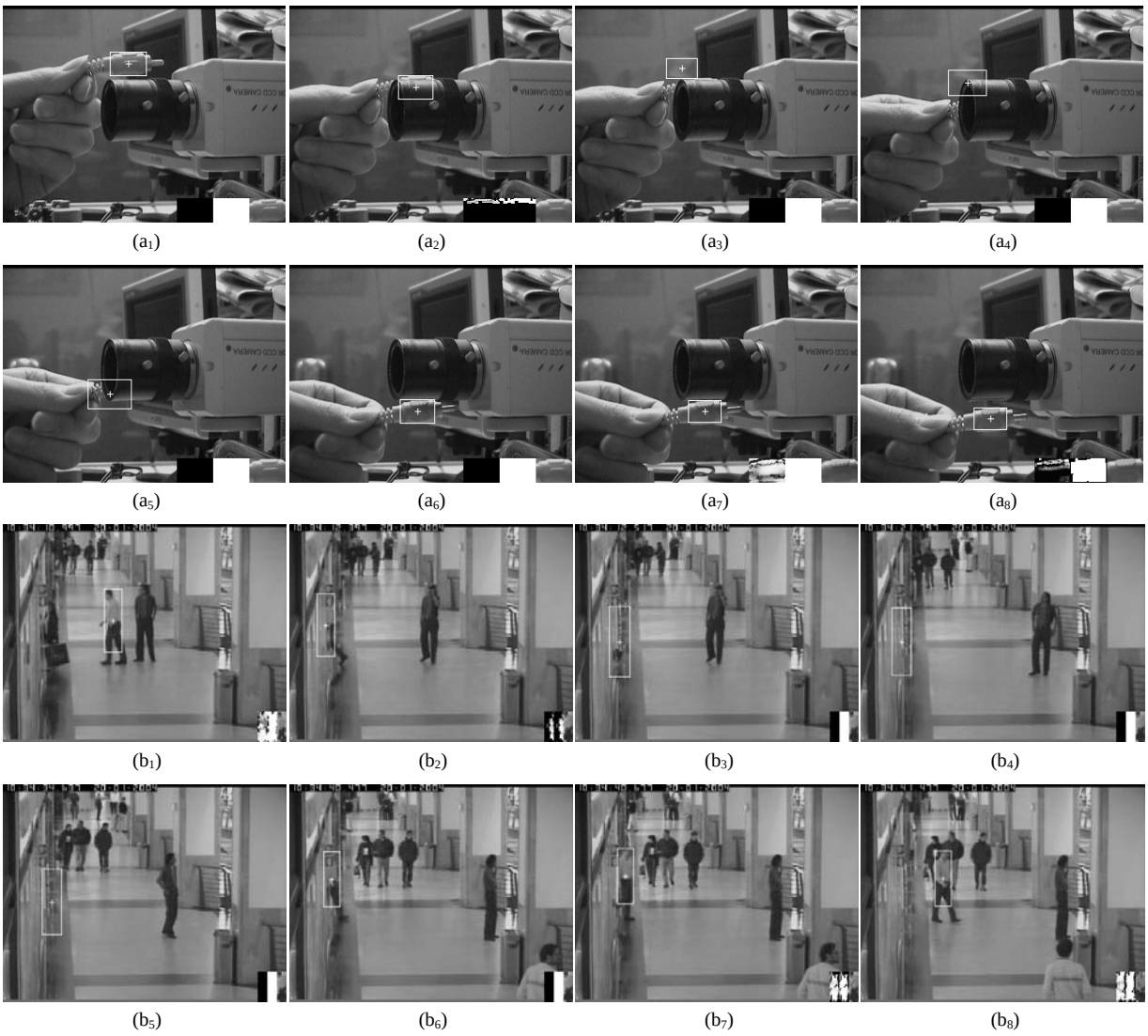
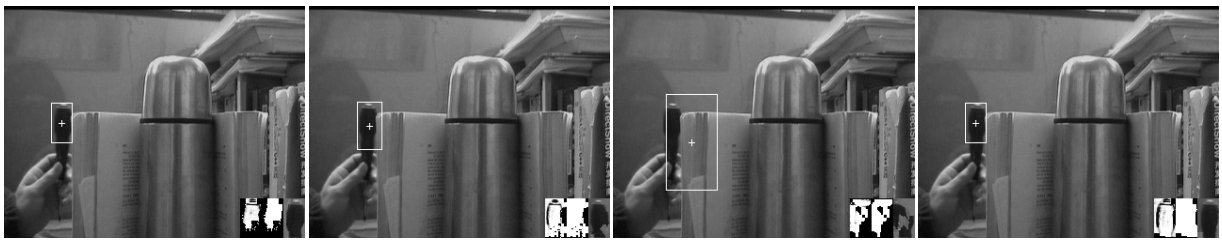


Fig. 10. Illustration of the tracking process of our proposed tracker under two long-term complete occlusions. The tracked target (ROI) is indicated by a white rectangle with a cross at the center. The lower right corner of each image is overlapped from left to right by the current Kalman gain, the current template mask and the current template. The Kalman gain is displayed on a scale of 0 to 1, where brighter pixels indicate higher Kalman gains. In the template mask, the masked portion is indicated in black. (a_1)-(a_8) The tracking process of the first sequence captured by us at frames 1, 100, 104, 167, 215, 294, 299, and 362. (b_1)-(b_8) The tracking process of the sequence *OneShopOneWait2cor* at frames 320, 355, 359, 590, 911, 1057, 1062, and 1082.

In Fig. 10, we demonstrate the tracking performance of our proposed tracker on two sequences containing long-term complete occlusions. The first sequence is captured by us and illustrated in the first two rows, where a plug moves downwards behind a video camera. The plug is complete occluded for about 150 frames. Our tracker enters the complete-occlusion mode at frame 101 and declares the reemergence of the target at frame 294. It returns to the normal mode 5 frames later. During the complete occlusion, although many background clutters exist, our tracker is never fooled. It should be noted that we use a constant-velocity model in this sequence to set the initial searching position for every new frame, in a hope to be in the vicinity

of the target when it is reemerging. However, if the motion of the target deviates too much from the constant velocity model, our tracker will fail to recapture the target. This case could have happened in the second sequence named *OneShopOneWait2cor* from CAVIAR. But fortunately, we have the prior knowledge in this sequence that if any target gets completely occluded behind a shop window (whose location is known and fixed), the best way to recapture it is to keep searching around the position where it first disappears. Using this prior information, our tracker fixes the initial searching point after entering the complete-occlusion mode (at frame 356), until it finds the target again (at frame 1057). The fluctuations of the key variables in the LBMA algorithm over this 700-frame complete occlusion are plotted in Fig. 9-b. In this plot, we could see the importance of the term $\delta \cdot \log(n_m - n_c)$. This term accounts for the potential appearance change of the target during complete occlusions. Our tracker would not have detected the reemergence of the target in time if this term had not been included.

As is discussed in Section IV-C, the adaptive threshold t is crucial for the CAPOA algorithm. In order to verify the importance of making t adaptive, we fix t to be a typical value (100) and run our modified tracking solution on the 66 sequences again. The results are shown in the column entitled "P-AdpThd" of Table II. As we can see, the tracking performance considerably degrades when t is forced to be a non-adaptive yet still reasonable value. Fig. 11 illustrates the tracking results using different strategies of setting t on a sequence containing a long-term partial occlusion. From the discussion in Section IV-C, we could infer that a low threshold would cause more false outlier alarms. This inference is confirmed by Fig. 11-a, where t is fixed as 2, a rather low threshold. When t is fixed at an average value (250), both false and missed outlier alarms can be observed, as is shown in Fig. 11-b. Note that a small part of the template has been damaged, and the tracking precision is lowered. In Fig. 11-c, setting t to be 400 leads to a disastrous consequence: the template is severely corrupted by pervading missed outlier alarms and the tracking fails. This is not



(a) (b) (c) (d)

Fig. 11. Comparison of the tracking performances using various strategies for setting the threshold t. (a) t = 2; (b) t = 250; (c) t = 400; (d) t is adaptive. All the images display the last frame of the sequence. The tracked target (ROI) is indicated by a white rectangle with a cross at the center. In the lower right corner of each image, the current Kalman gain, the current template mask and the current template are displayed from left to right. The Kalman gain is displayed on a scale of 0 to 1, where brighter pixels indicate higher Kalman gains. In the template mask, the masked portion is indicated in black.

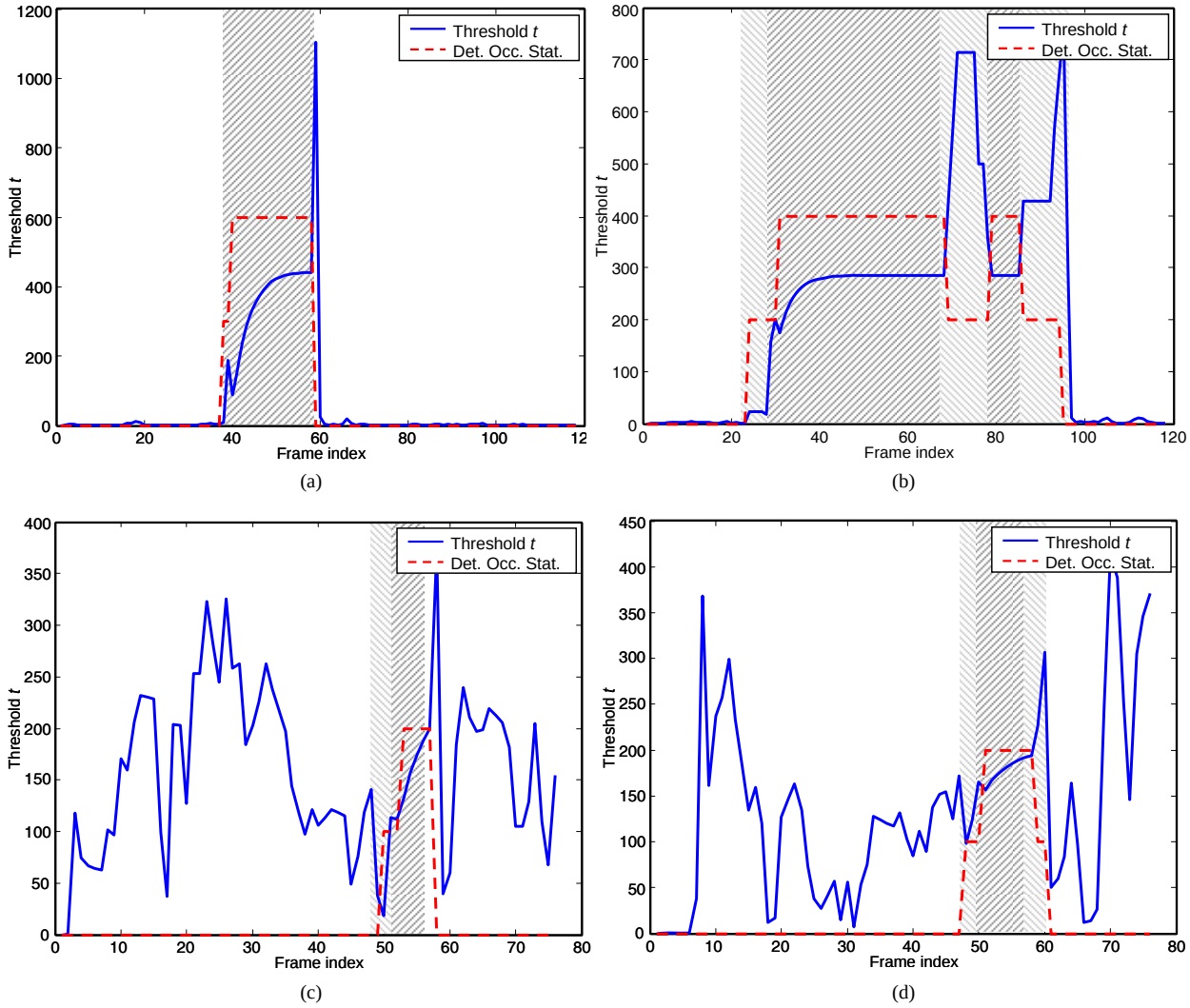


Fig. 12. Illustration of the fluctuation of the threshold t over time under various situations. "Det. Occ. Stat" means "detected occlusion status." (a) target location 1 for sequence segment 1; (b) target location 2 for sequence segment 1; (c) target location 1 for sequence segment 2; (d) target location 2 for sequence segment 2. Please refer to the text for details.

surprising because a high threshold tends to induce missed outlier alarms. The tracking result using the proposed adaptive threshold is demonstrated in Fig. 11-d, where few false or missed outlier alarms occur and the template is very well preserved.

We also record how the threshold t fluctuates over time at different locations of different targets. The values of t are plotted against frame indices in Fig. 12. The blue real line represents the values of t calculated by the CAPOA algorithm for a selected block. The red dashed step-shaped line is overlapped to indicate the *algorithm-detected* occlusion statuses of the block: the highest step indicates complete occlusion, the middle step indicates partial occlusion, and the lowest step indicates no occlusion. The ground-truth occlusion situations are shown by manually shading the plot areas. Lighter shade represents partial occlusion and

darker shade represents complete occlusion. Fig. 12-a and 12-b illustrate the variations of t for two 5-by-5 image blocks located at different positions of the same target in a sequence segment. The target appearances associated with the two blocks have little change over the sequence. The fluctuations of t in the two plots could easily be explained by (46) and (33). We could see that the CAPOA algorithm has different strategies of setting thresholds for different locations of the same target. In Fig. 12-c and 12-d, the two image blocks lie on a target in another sequence segment. The appearance of the target changes irregularly and intensively. In this case, what dominates the threshold is the intensity of the variations in the target appearance. When occlusions occur, the fluctuation pattern of the threshold is similar to the patterns in Fig. 12-a and 12-b. In this experiment it could be seen that the threshold in the CAPOA algorithm is fully adaptive to time, location, occlusion situation, and target appearance.

As is mentioned in Section IV, the CAPOA algorithm has a strong capability of correcting errors in the outlier map. We conduct the following experiment to verify this claim. In the experiment, the initial outlier map within the ROI is intentionally set to be a random binary matrix instead of being initialized as an array

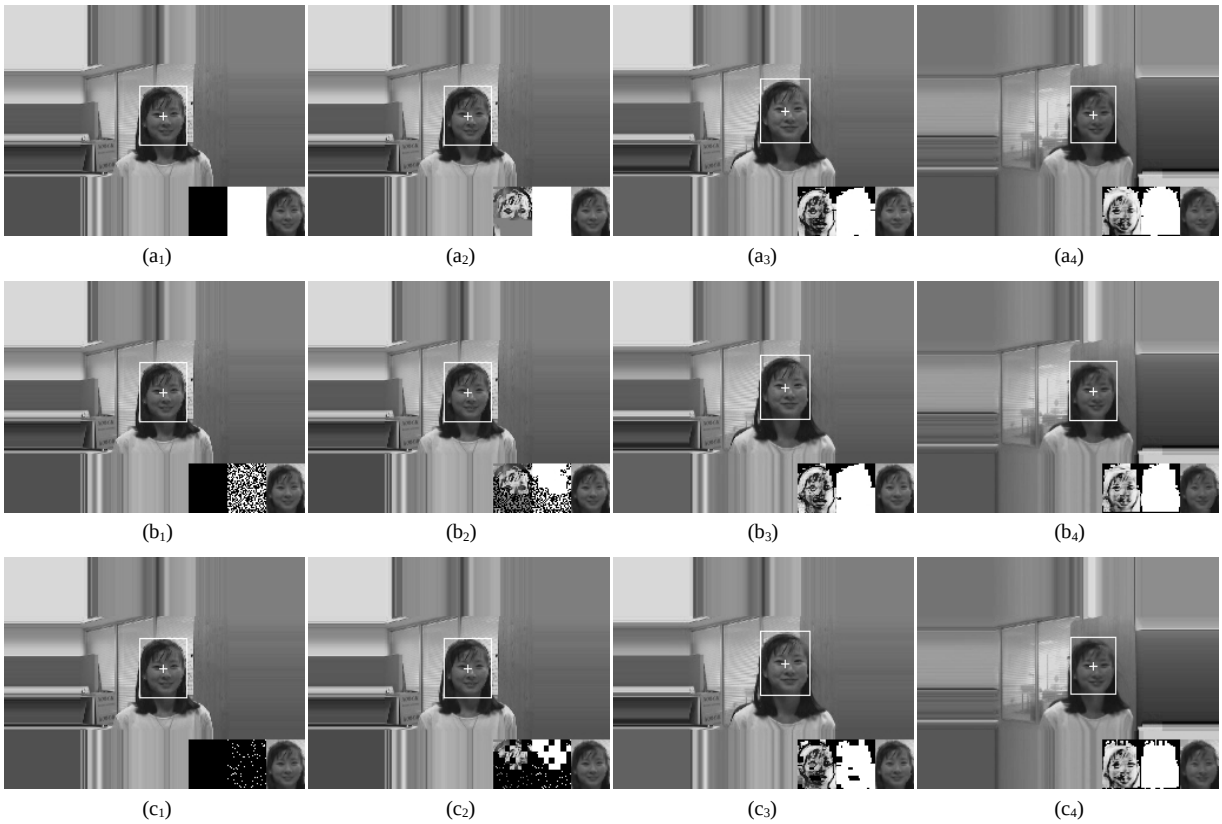


Fig. 13. Demonstration of the error-correction capability possessed by the CAPOA algorithm. The corruption degree of the initial outlier map increases from the top row to the bottom row. The four columns from left to right show frames 1, 2, 8, 16 of the sequence *seq_mb*, respectively. The tracked target (ROI) is indicated by a white rectangle with a cross at the center. In the lower right corner of each image, the current Kalman gain, the current template mask and the current template are displayed from left to right. The Kalman gain is displayed on a scale of 0 to 1, where brighter pixels indicate higher Kalman gains. In the template mask, the masked portion is indicated in black.

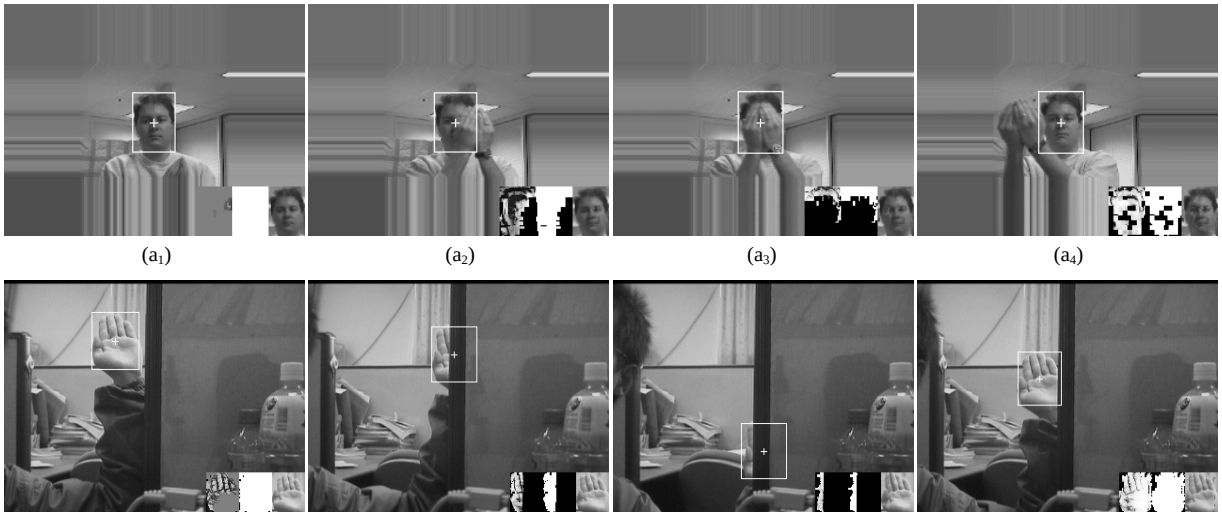
of zeros. By adjusting the probability of the pixels taking the value of one, we control how "wrong" the initial outlier map is. The sequence under test is *seq_mb* from Birchfield's head tracking sequences. The tracking results of four typical frames out of the first 16 frames are illustrated in Fig. 13. In the first row, we initialize the outlier map as normal. During the initialization, some background pixels are also included into the template. After frame 8, however, an approximate contour of the target is formed in the template mask when the background pixels are identified as outliers (see Fig. 13-a_3 and 13-a_4). In the second row, about 50% of the pixels in the ROI of the initial outlier map are corrupted (see Fig. 13-b_1), but roughly half of the errors are corrected in the frame immediately following the initialization (see Fig. 13-b_2). Six frames later, the template mask is nearly the same as in the first row (see Fig. 13-b_3 and 13-b_4). The corruption percentage in the third row is as high as around 95%. Still, as in the second row, the errors disappear very soon. At frame 16, the template mask is fully recovered. Note that background pixels are always recognized as outliers. This experiment confirms the fact that errors in the outlier map do not accumulate and propagate away in our proposed algorithm.

Now we examine the sensitivity of our tracking solution to the perturbation on the location of the initial ROI. We run our tracking solution again on the sequences that are successfully tracked before, but this time we randomly perturb the coordinates of the four vertices of the initial rectangular ROI by 2%, 4%, and 6% of the ROI size, respectively. The ratios of the numbers of successful sequences *after* the perturbation to the original numbers are listed in Table III, where the left column is the strength of perturbation and the right column contains the ratios. It is observed that our tracking solution is insensitive to the perturbation on the initial ROI location.

Our tracking solution enters the complete-occlusion mode only when the occlusion percentage reaches as high as 85%. This means our solution can still successfully track a target in the *normal* mode even when more than two-thirds of it is occluded. We give more examples related to this feature of our solution in Fig. 14. The first example is demonstrated in the first row, where the sequence under test is *seq_ms* from Birchfield's head tracking sequences. In the sequence, the man's head is severely occluded by his hands for multiple times, but our tracking solution is never distracted away by the occluder. In order to observe the performance of our solution when a severely-occluded target is also moving, we capture another sequence in which a hand moves up

TABLE III
The sensitivity of the proposed tracking solution to the perturbation on the initial ROI location

Perturb Strength	Ratio
2%	0.99
4%	0.94
6%	0.90



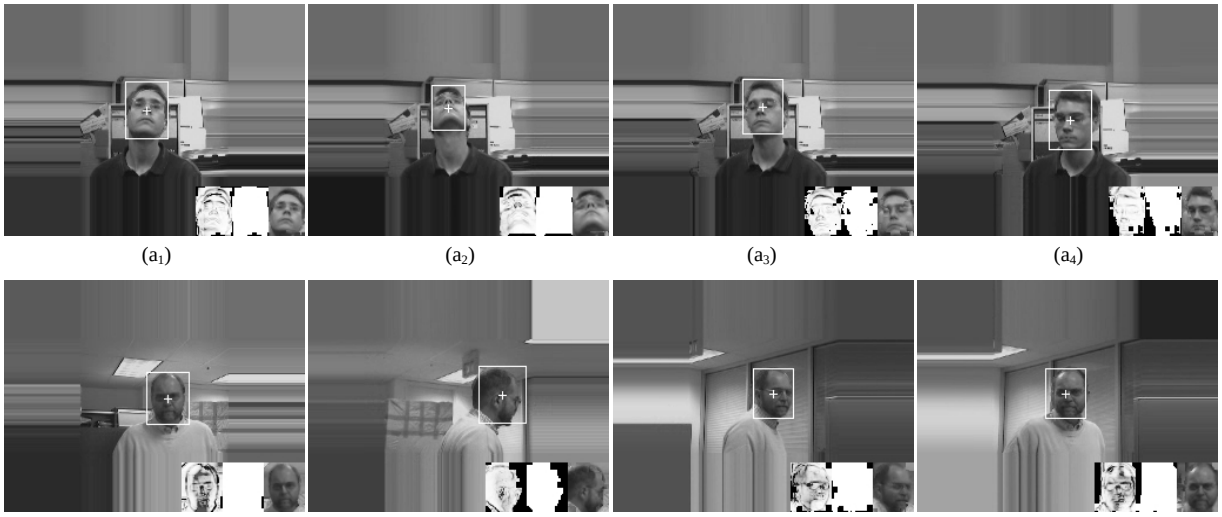
(a_1) (a_2) (a_3) (a_4)

(b_1) (b_2) (b_3) (b_4)

Fig. 14. Demonstration of the performance of our tracking solution under extremely severe occlusion. The first row displays the sequence *seq_ms* at frames 2, 8, 23, and 49. The second row illustrates a sequence shot by us at frames 2, 47, 91, and 178. The tracked target (ROI) is indicated by a white rectangle with a cross at the center. In the lower right corner of each image, the current Kalman gain, the current template mask and the current template are displayed from left to right. The Kalman gain is displayed on a scale of 0 to 1, where brighter pixels indicate higher Kalman gains. In the template mask, the masked portion is indicated in black.

and down while a majority of it is behind a board most of the time, as is displayed in the second row. We could see that our solution also performs very well by generating a suitable template mask to track only the non-occluded part.

As our tracking solution can adaptively update the template, we only use translational and scaling parameters to model target motion for the sake of computational simplicity and stability. Other types of



(a_1) (a_2) (a_3) (a_4)

(b_1) (b_2) (b_3) (b_4)

Fig. 15 Demonstration of the performance of our tracking solution when targets undergo non-rigid deformation and out-of-plane rotation. The first row displays the sequence *seq_sb* at frames 174, 179, 182, and 184. The second row illustrates the sequence *seq_dhb* at frames 3, 16, 37, and 51. The tracked target (ROI) is indicated by a white rectangle with a cross at the center. In the lower right corner of each image, the current Kalman gain, the current template mask and the current template are displayed from left to right. The Kalman gain is displayed on a scale of 0 to 1, where brighter pixels indicate higher Kalman gains. In the template mask, the masked portion is indicated in black.

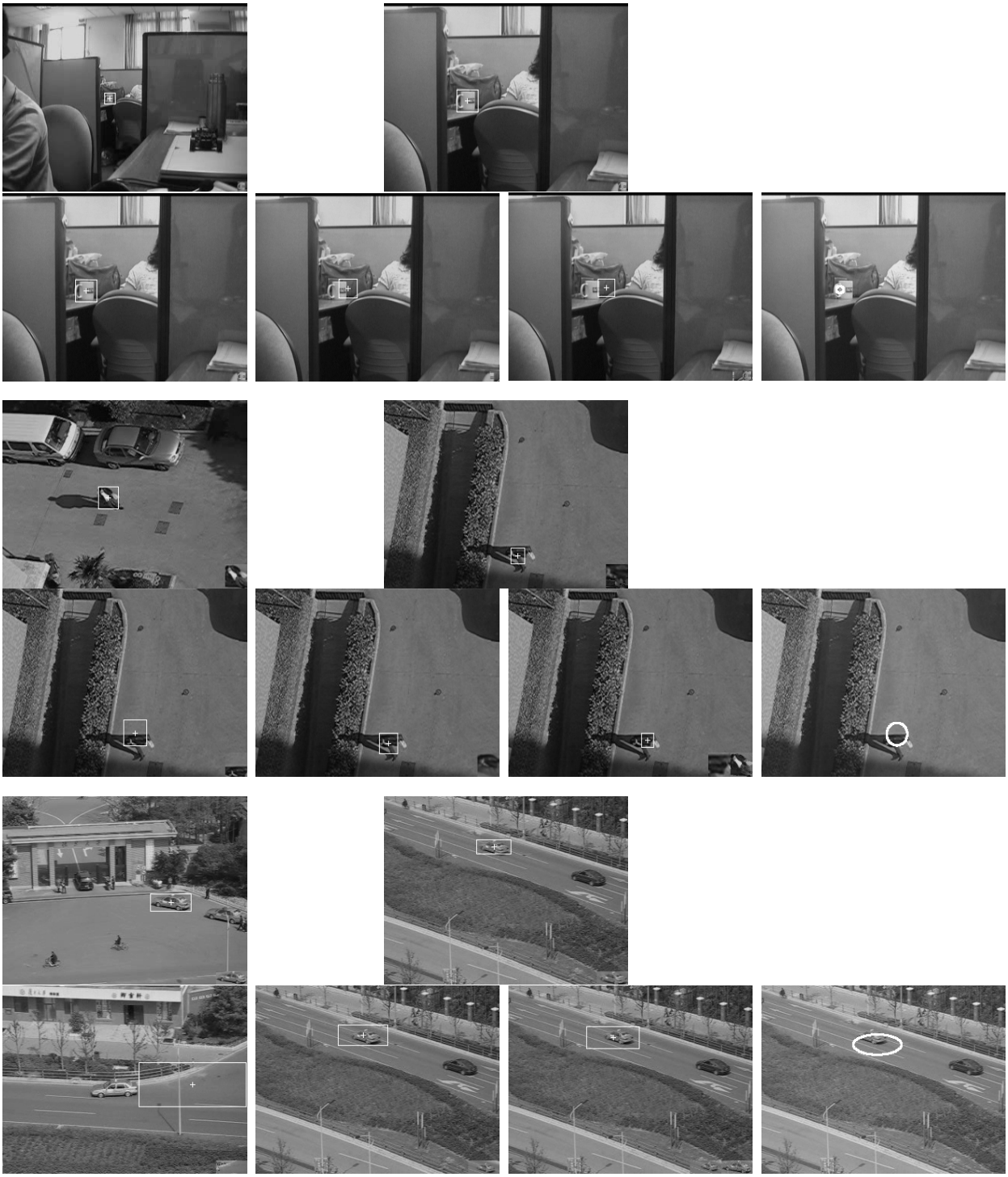


Fig. 16. Comparison of the performances of different algorithms in terms of reducing template drift. The three *groups* of rows illustrate different sequences taken by us. For each *group*, the six images from left to right and top to bottom are: common initialization, the results of DIMKAF, TUDC [2], RAF [8], WSL [24], and MS [28], respectively. The tracked target is indicated by an ellipse for MS. The current template (or appearance model) is overlapped in the lower-right corner of each image where applicable. The three sequences from top to down end at frames 200, 162, and 554, respectively. In the third sequence, TUDC loses the target and the tracking is therefore terminated at frame 346.

target motion, such as non-rigid deformation and out-of-plane rotation, are regarded as the variations of

target appearance and learned by the template. Therefore, they seldom pose a threat to our tracker. Two typical examples are illustrated in Fig. 15. The first row contains frames of the sequence *seq_sb* from Birchfield's head tracking sequences. We could see that the man's head undergoes very rapid motion that is hard to be modeled by any affine transformation. Nevertheless, by updating the template in time, our proposed solution has no difficulty in tracking it. In the second row, our-of-plane rotation occurs in the sequence *seq_dhb* from the same standard dataset. Again, our tracking solution succeeds.

B. Accuracy of Object Tracking

To begin, we verify the contribution that the DIMKAF makes to reducing template drift and enhancing tracking accuracy. We conduct lots of experiments on real-world sequences and compare the tracking accuracies of different tracking algorithms. Three typical cases are illustrated in the three *groups* of rows of Fig. 16. For each *group*, the six images from left to right and top to bottom are: common initialization, the tracking results of DIMKAF, TUDC [2], RAF [8], WSL [24], and MS [28], respectively. Here, TUDC is the short for "template update with drift correction", a drift-correction technique proposed in [2]. From the top sequence to the bottom one, the variations of the target appearance become increasingly intensive. It is observed in our experiments that when the target appearance varies little, RAF is prone to suffer from template drift as a result of unnecessarily high Kalman gains (see the second image of the second row). On the other hand, when the target appearance undergoes much variation, the tracking accuracy of RAF is also unsatisfactory (see the second image of the sixth row) because it does not update the template in time. The performance of TUDC is very good with a stationary target appearance (see the first image of the second row). However, it degrades a lot when facing a changing target appearance (see the first image of the fourth row) due to the invalidity of the first template. It even loses the target in some cases (see the first image of the sixth row). WSL suffers from some template drift almost in all cases because it does not have an explicit model for the drift noise. The tracking accuracy of MS is found to be rather low. Nevertheless, this is not caused by template drift, as MS does not update its histogram model. What leads to the low tracking precision is that histogram itself is not very suitable for accurately locating a target. Our proposed solution tracks the target accurately under all circumstances with the aid of the DIMKAF.

In order to evaluate the degree of template drift *quantitatively*, we perform further experiments on synthetic test sequences so that the ground-truth values of the transformation parameters are known in advance and the experiments can be conducted in a controlled manner. We generate test sequences using

(a) (b)

Fig. 17. Sample frames from synthetic sequences. The target is highlighted by a red ellipse. (a) A sample frame from the sequence for comparing template drift. (b) A sample frame from the sequence for evaluating the contribution of VMTM.

the 512-by-512 standard test image "lake", in which the image block containing the sailboat (lines 367 to 447, columns 296 to 350) is extracted as the target and overlapped on the original image after being scaled and altered in appearance. The trajectory of the target is a spiral with the form

$$\begin{cases} x(n) = (10 + r \cdot n)\cos(\pi \cdot r \cdot n / 20) + 256 \\ y(n) = (10 + r \cdot n)\sin(\pi \cdot r \cdot n / 20) + 256 \end{cases}, \quad n = 0,1,\cdots, \tag{60}$$

where r continuously changes to ensure that the target is moving at a constant speed of 2 pixels per frame. The scale of the target varies between 0.5 and 1.5 at a constant rate of 0.03 per frame. The appearance of the target is altered by modifying pixel values according to

$$I'(\boldsymbol{x}, n) = \begin{cases} [I(\boldsymbol{x}, n) - 128] \cdot k + 128, & \boldsymbol{x} \in \boldsymbol{\Omega}_A \\ I(\boldsymbol{x}, n), & else \end{cases}, \quad n = 0,1,\cdots, \tag{61}$$

where $I(\boldsymbol{x}, n)$ and $I'(\boldsymbol{x}, n)$ represent the original and the modified pixel value, respectively. k is a time-variant parameter which takes values between -1 and 1 at a constant changing rate. $\boldsymbol{\Omega}_A$ is the region where the target appearance is under modification. The left, upper, right, and lower half of the target is selected as $\boldsymbol{\Omega}_A$ alternately, where k finishes one cycle for each half. By varying the changing rate of k, we control how fast the target appearance changes. A sample frame is illustrated in Fig. 17-a.

Fig. 18 shows the results of the first experiment in which the test sequence has 300 frames and the target appearance remain fixed throughout the sequence. Various algorithms with adaptive template update are tested to compare their mean tracking errors which are measured by the average Euclidean distances between the estimated and the true values of the transformation parameter vectors at all frames. In Fig. 18, "ATT"

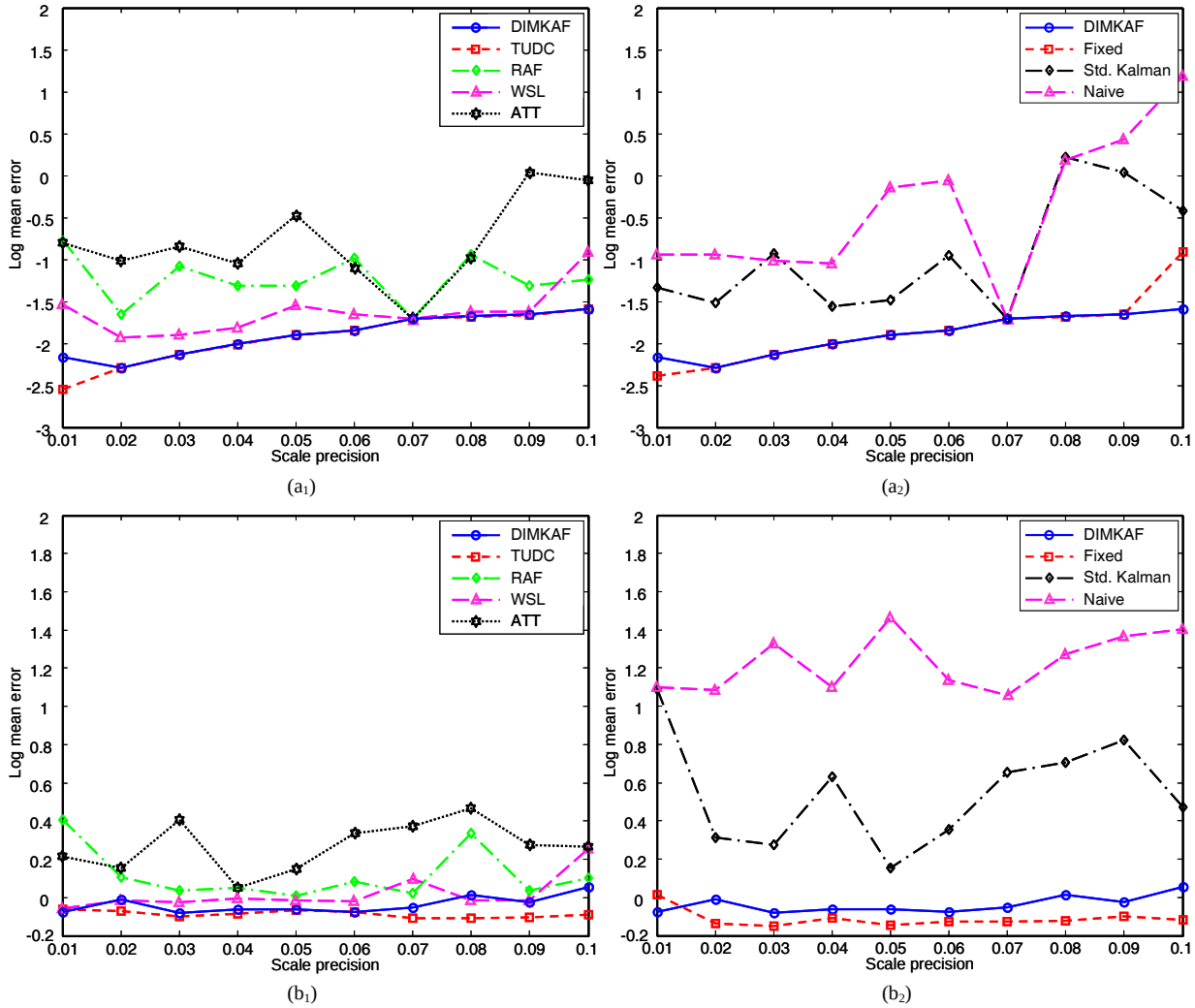


Fig. 18. Comparison of template drift with a fixed target appearance. (a_1)-(a_2) Precision of translational parameters is 1. (b_1)-(b_2) Precision of translational parameters is 2. Please refer to the text for details.

denotes the "occlusion robust adaptive template tracking" algorithm proposed in [7]; "Fixed" represents keeping the template unchanged; "Std. Kalman" means using standard Kalman filter to update the template where noise parameters are given and fixed; "Naïve" means replacing the template every frame with the previous ROI. Our algorithm DIMKAF is displayed in all the plots to facilitate comparison. In each plot, the x-axis is the searching precision of the *scaling* parameter, and the y-axis is the logarithm of the mean tracking error mentioned above. The top row and the bottom row show the results when the searching precisions of the *translational* parameters are 1 and 2, respectively. From Fig. 18, we could see that tracking with a fixed template has the minimal tracking error. This is not surprising, because when the target appearance does not change, the best strategy to update the template is not updating it at all. The performance of TUDC is similar to the best one, as the first template used to rectify the target location is

always valid in the case of a fixed target appearance. It is observed in the experiment that although the target appearance remains the same, the innovation is much larger when the target is smaller in scale and thus compact with features. Evidently, such large innovation results from matching inaccuracy, not the variation of the target appearance. The DIMKAF takes this into account by raising the measurement noise power and keeping the Kalman gain low. As a result, the performance of the DIMKAF is still very close to the best one and the tracking errors are virtually indiscernible by the naked eye. RAF, ATT and standard Kalman filter incur much greater tracking errors as a result of suboptimal calculation of the Kalman gain. WSL also has greater template drift than our algorithm because of its W component. When the "naïve" approach is taken, severe template drift is observed.

The results of tracking a target with a changing appearance are displayed in Fig. 19. In this test

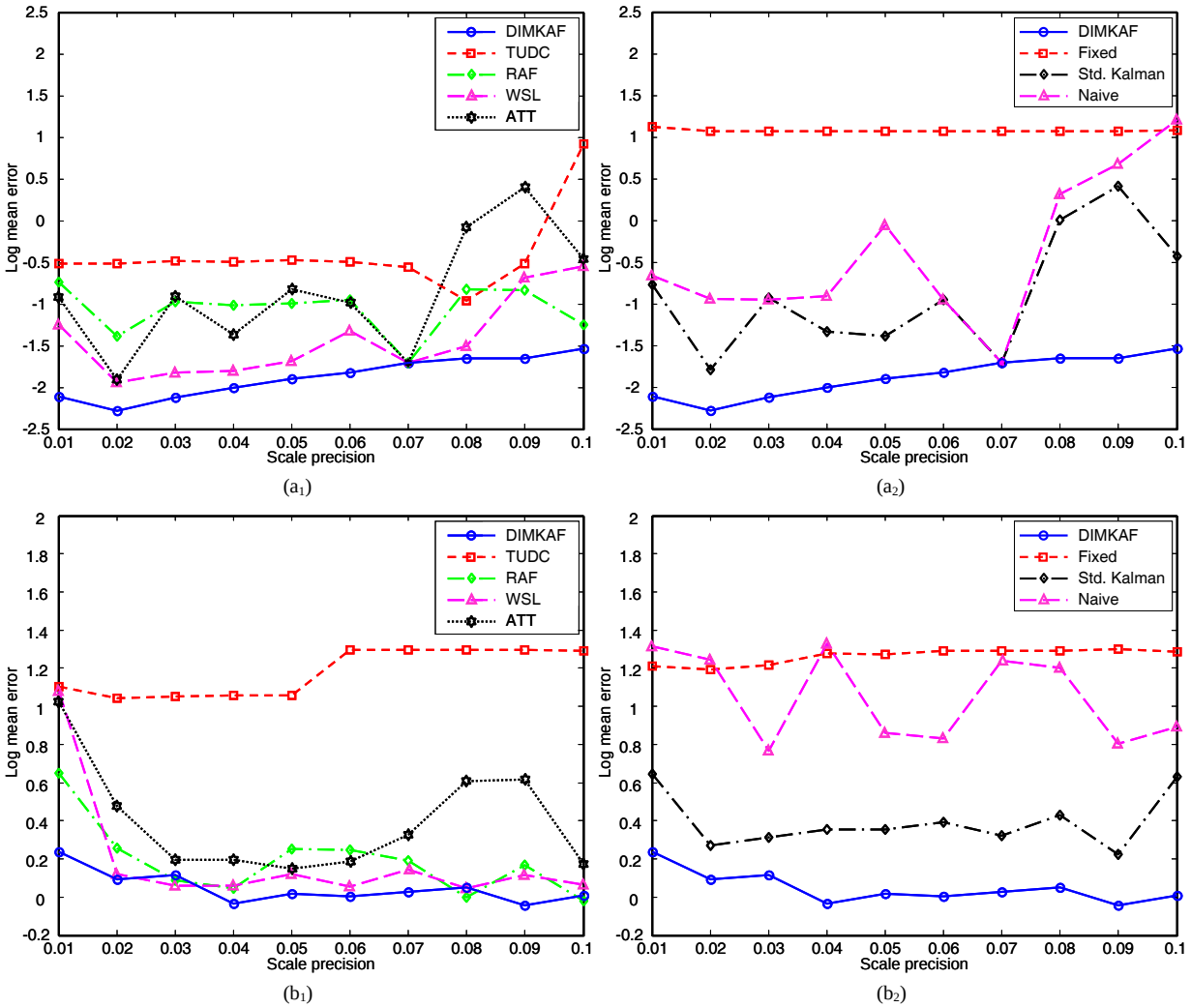


Fig. 19. Comparison of template drift with a changing target appearance. (a_1)-(a_2) Precision of translational parameters is 1. (b_1)-(b_2) Precision of translational parameters is 2. Please refer to the text for details.

sequence, we set the changing rate of the k in (61) to be 0.01 per frame. The displaying conventions of Fig. 19 are exactly the same as Fig. 18. In Fig. 19 we could find that the tracking error of keeping the template unaltered rises significantly. So does the performance of TUDC. The "naïve" approach still suffers from severe template drift. Among the remaining algorithms, the DIMKAF achieves much better tracking accuracy than any other approach. From the experimental results, we can see that whether the target appearance changes or not, the DIMKAF always effectively reduces tracking error by updating the template just in time and just in place to keep up with the variation of the target appearance while refraining from over-updating the template.

It is also interesting to note that when the searching precision for the translational parameters is 1 and for the scaling parameter is 0.07, almost all the algorithms achieve the best tracking accuracy. This is because such a combination of searching precisions happens to allow a certain searching point to coincide with the ground truth data. As a result, little template drift would exist no matter how the template is updated. This phenomenon also validates our discussion about the ultimate cause of template drift.

Secondly, we examine how the VMTM operation guarantees accurate object tracking when facing severe occlusions. The experimental results on the real-world sequences are shown in the column entitled "P-VMTM" in Table II. Without performing the VMTM, our tracking solution fails for some sequences it could have successfully tracked otherwise. In the other sequences for which the tracking still succeeds, it is observed that the tracking accuracy drops as a result of leaving out the VMTM. Two typical cases are illustrated in Fig. 20. The first two rows show a case in which the target is lost if the VMTM is omitted, and the last two rows display a case where the tracking accuracy is undermined if the VMTM is not performed.

In order to quantify the contribution of the VMTM and acquire a deeper insight into the underlying factors, we perform experiments on synthetic sequences as well. Those sequences are also based on the standard test image 'lake', and are generated by moving the image block of the sailboat (mentioned above) diagonally from (178,178) to (370,370) at a constant velocity. Along the way, the target is partially occluded by a 61-by-61 original image block centered at (276,276). The highest occlusion percentage exceeds 75%. The scale of the target varies between 0.5 and 1.0 at a rate of 0.01 per frame. A sample frame is illustrated in Fig. 17-b. We measure the tracking accuracy of the proposed tracking solution with and without the VMTM under various target speeds.

The experimental result is illustrated in Fig. 21. It can be observed that when the relative target speed with

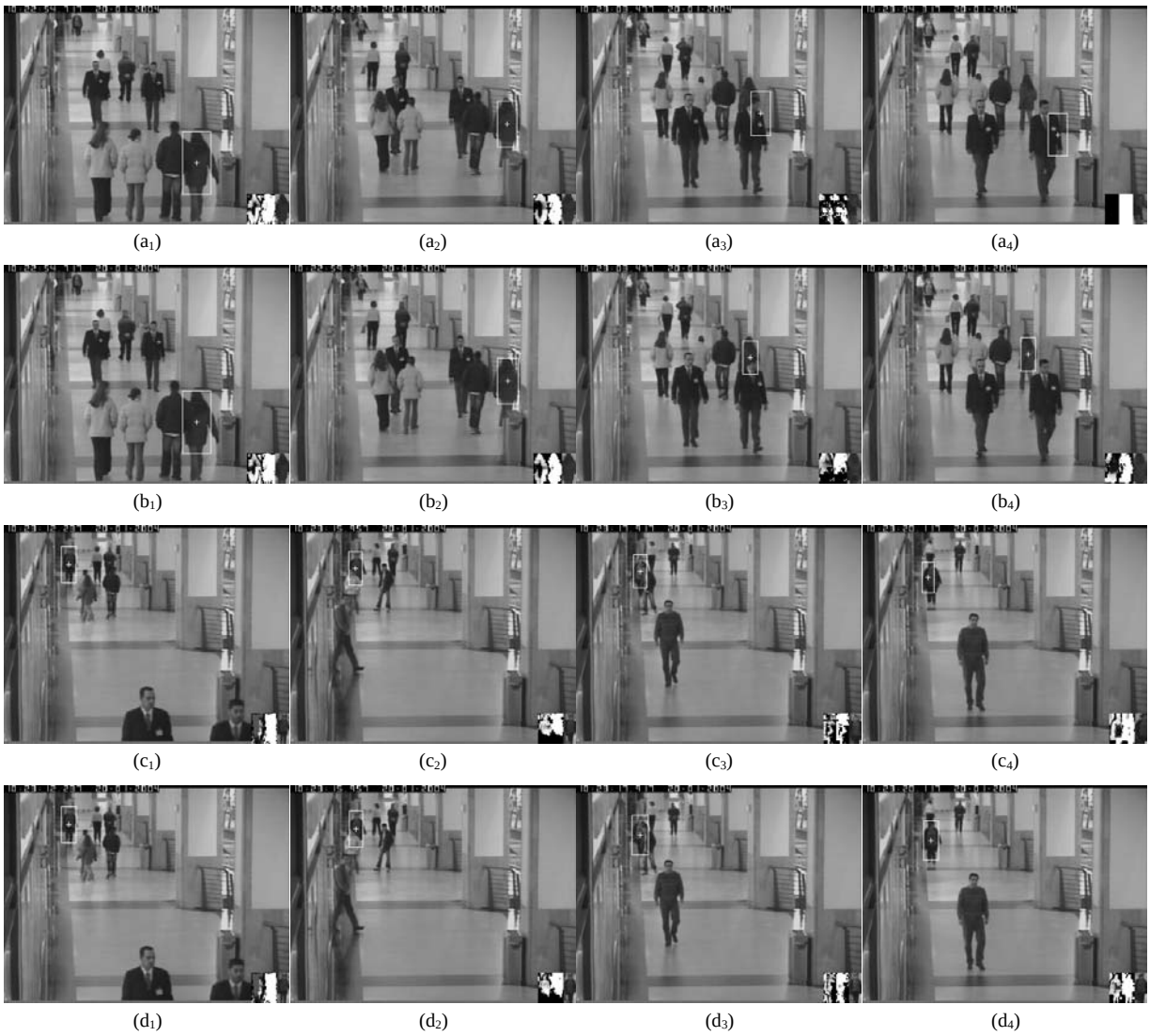


Fig. 20. Comparison of the tracking performances when using our proposed solution with and without performing the VMTM. (a_1)-(a_4) The target is lost when the VMTM is *not* performed. (b_1)-(b_4) The target is tracked well when the VMTM is performed. (c_1)-(c_4) The tracking precision is low when the VMTM is *not* performed. (d_1)-(d_4) The tracking precision is high when the VMTM is performed. The sequence under test is *OneStopMoveEnter1cor*. In the top two rows, we choose a girl as the target, and the four images from left in each row display frames 765, 877, 983, and 1004, respectively. In the bottom two rows, we choose a man as the target, and the four images from left in each row display frames 1202, 1295, 1344, and 1399, respectively.

respect to the occluder is low, the proposed tracking solution achieves high tracking accuracy no matter whether the VMTM is employed. When the target moves faster, however, the tracking error with the VMTM almost remains the same, while the tracking error without the VMTM drastically increases. This is because with the rise of the relative target speed, the non-occluded part of the target in the current frame becomes increasingly dissimilar to the non-occluded part in the previous frame. As a result, the template mask generated according to the previous occlusion situation becomes increasingly imprecise in guiding the search for the target location in the current frame, thus leading to the soaring tracking error. By performing the

VMTM, the target location is always effectively rectified and the tracking error is therefore always kept low.

C. Computational Complexity

In order to ensure the robustness and accuracy of object tracking, our proposed solution involves many schemes other than the basic template matching operation. As a result, the computational complexity of our proposed solution is naturally higher than the algorithms that only contain single-pass template matching, such as RAF. However, as our proposed solution uses progressive scanning and restricts the scanning scope *within* the ROI when analyzing the occlusion situations, the computational burden is much lower than ABM where the entire frame undergoes analysis and no template matching is performed. After we conduct experiments on a wide range of video sequences, the average implementation time for various algorithms to process one frame on the MATLAB platform is shown in Table IV. Although the implementation speed of our proposed solution is lower than that of RAF and WSL, the tracking stability and accuracy of our proposed solution is much higher.

Now we examine the effectiveness of our proposed fast algorithm for the DIMKAF mentioned in Section III-C. Fig. 22 shows

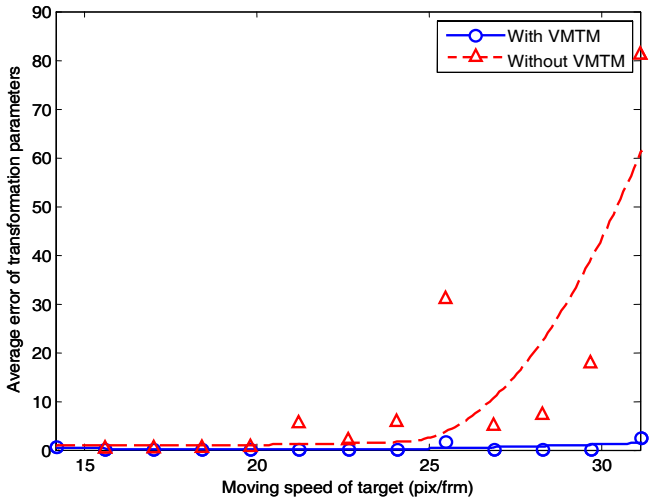


Fig. 21. The plot of tracking error against target speed when applying the proposed tracking solution with and without the VMTM. The curves are LS-fitted lines using piecewise quadratic functions.

TABLE IV
Average Implementation Time for Various Algorithms

Algorithm	Time (sec)
RAF	0.1076
WSL	0.3260
MS	1.9719
ABM	7.9063
Proposed	0.7442

The data is collected by running the algorithms on the MATLAB platform.

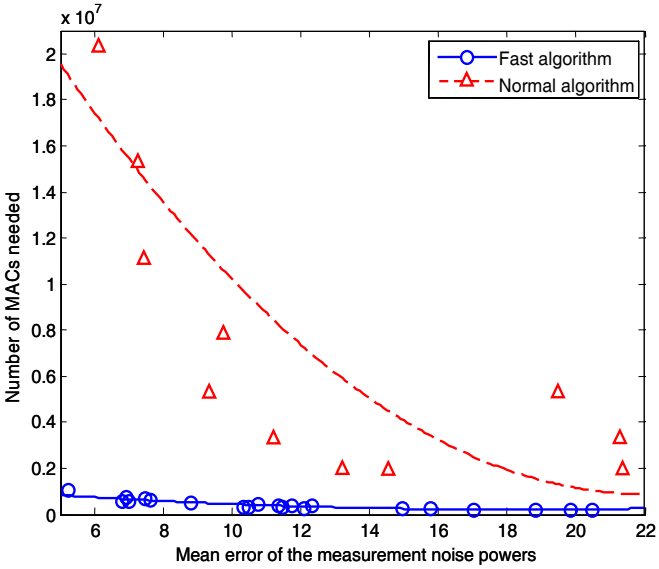


Fig. 22. A comparison of the computational complexity when performing the DIMKAF with and without employing the proposed fast algorithm. The curves are LS-fitted lines using piecewise quadratic functions.

the numbers of MAC (multiply-accumulate) operations needed for a single implementation of the DIMKAF using the normal algorithm and the fast algorithm, respectively. The *x*-axis is the mean error between the true and the estimated measurement noise powers of the pixels over the entire template. In the experiment, we vary the size of the summation units in (20) and (24) to yield a series of mean errors and their corresponding computational complexities. As is indicated in Fig. 22, the computational burden rises very little when the mean error approaches zero if the fast algorithm is employed. By contrast, the normal algorithm suffers from a dramatic increase in the computational complexity when the mean error reduces.

D. Failure Modes

There are two cases that could lead to the failure of our tracking solution: 1) When a human head rotates from a frontal view to a rear view, the dark hair that gradually becomes visible from one side of the face sometimes fails to be recognized as part of the target; 2) during the period of complete occlusion, the motion of the target is so irregular that the target is out of the searching range when it reemerges.

The first difficulty is encountered by most tracking algorithms, including those with adaptive template update (like ours). Without prior knowledge, there is really no reason for the machine to believe that the newly-appeared hair belongs to the head which is initially a white face. However, when the contrast is not so significant, our tracking solution can still succeed (see Fig. 15-b). In order to overcome this problem, multiple-appearance models could be used, yet at the expense of adding the need of training before tracking.

The second difficulty could be solved by enlarging the searching range. However, this would incur strong interference from background clutters and much higher computational complexity. Efficient detection of a reappearing target over the entire image is one of the focuses of our future work.

VIII. CONCLUSION

In this paper we propose an object tracking solution that contains a set of algorithms aiming at enhancing the robustness and accuracy of object tracking when various types of occlusions occur. In the proposed solution, the content-adaptive progressive occlusion analysis (CAPOA) algorithm effectively acquires the current occlusion situation by: scanning the ROI progressively, using the joint information provided by three sources and making decisions according to content-adaptive thresholds. The variant-mask template matching (VMTM) is performed to rectify the target location that initially might be

inaccurate due to the influence of occlusions. The drift-inhibitive masked Kalman appearance filter (DIMKAF) significantly reduces template drift by correctly evaluating the noise models and obtaining the optimal Kalman gains for all the template pixels at every frame. The local best match authentication (LBMA) method reliably detects the end of complete occlusions by authenticating the genuineness of local best matches. We verify the effectiveness of our proposed solution by conducting experiments on a wide range of real-world video sequences. Synthetic test sequences are also employed to evaluate the tracking performance quantitatively. The experimental results have confirmed the robustness and accuracy of our proposed object tracking solution under almost all types of tracking scenarios.

Our tracking solution could be further improved by "softening" the outlier map so that a smooth transition from non-occlusion to occlusion could be realized. This requires assigning a reasonable "belief of occlusion" for each pixel, which is another focus of our future research.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their comments and suggestions that have considerably improved the quality of this paper. They would also like to thank John L. Haller Jr. for his valuable revision of the paper.

REFERENCES

[1] S. Baker and I. Matthews, "Lucas-Kanade 20 Years on: A Unifying Framework," *Int'l J. Computer Vision*, vol. 53, no. 3, pp. 221-255, 2004.
[2] I. Matthews, T.Ishikawa, and S. Baker, "The Template Update Problem," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 810-815, 2004.
[3] T. Kaneko and Osamu Hori, "Template Update Criterion for Template Matching of Image Sequences," *Proc. IEEE Int'l Conf. Pattern Recognition*, vol. 2, pp. 1-5, 2002.
[4] A.M. Peacock, S. Matsunaga, D. Renshaw, J. Hannah, and A. Murray, "Reference Block Updating When Tracking with Block Matching Algorithm," *Electronic Letters*, vol. 36, pp. 309-310, 2000.
[5] C. Smith, C. Richards, S. Brandt, and N. Papanikolopoulos, "Visual Tracking for Intelligent Vehicle-highway Systems," *IEEE Trans. on Vehicular Technology*, vol. 45, no. 4, pp. 744-759, 1996.
[6] N. Papanikolopoulos, P. Khosla, and T. Kanade, "Visual Tracking of a Moving Target by a Camera Mounted on a Robot: A Combination of Control and Vision," *IEEE Trans. on Robotics and Automation*, vol. 9, pp. 14-35, 1993.
[7] H.T. Nguyen, M. Worring, and R. van den Boomgaard, "Occlusion Robust Adaptive Template Tracking," *Proc. IEEE Int'l Conf. Computer Vision*, vol. 1, pp. 678-683, 2001.
[8] H.T. Nguyen and A. W.M. Smeulders, "Fast Occluded Object Tracking by a Robust Appearance Filter," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 8, pp. 1099-1104, 2004.
[9] C. Haworth, A.M. Peacock, and D. Renshaw, "Performance of Reference Block Updating Techniques When Tracking with the Block Matching Algorithm," *Proc. IEEE Int'l Conf. Image Processing*, vol. 1, pp. 365-368, 2001.
[10] Z. Jia, A. Balasuriya, and S. Challa, "Target Tracking with Bayesian Fusion Based Template Matching," *Proc. IEEE Int'l Conf. Image Processing*, vol. 2, pp. II - 826-9, 2005.
[11] K. Ito and S. Sakane, "Robust View-based Visual Tracking with Detection of Occlusions," *Proc. IEEE Int'l Conf. Robotics and Automation*, vol. 2, pp. 1207-1213, 2001.
[12] K. Hariharakrishnan and D. Schonfeld, "Fast Object Tracking Using Adaptive Block Matching," *IEEE Trans. on Multimedia*, vol. 7, no. 5, pp. 853-859, 2005.
[13] R.G. Brown and P.Y.C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, John Wiley, 1992.
[14] R.C. Gonzalez and R.E. Woods, *Digital Image Processing, Second Edition*, pp. 598-600, Prentice Hall, 2002.
[15] J. Jain, and A. Jain, "Displacement Measurement and Its Application in Interframe Image Coding," *IEEE Trans. on Communications*, vol. 33, pp. 1799-1808, 1981.
[16] L.K. Liu, and E. Feig, "A Block-Based Gradient Descent Search Algorithm for Block Motion Estimation in Video Coding," *IEEE Trans. on Circuits and Systems for Video Technology*, vol.6, pp. 419-422, 1996.

[17] G. Sinevriotis, and T. Stouraitis, "A Computationally Efficient Gradient Search Block Matching Algorithm for the Motion Estimation of Image Sequences," *Proc. IEEE Int'l Conf. DSP*, vol. 2, pp. 1127-1130, 1997.
[18] Y. Wang, J. Ostermann, and Y.Q. Zhang, *Video Processing and Communications*, pp. 159-161, Prentice Hall, 2002.
[19] M.J. Black and Y. Yacoob, "Recognizing Facial Expressions in Image Sequences Using Local Parameterized Models of Image Motion," *Int'l J. Computer Vision*, vol. 25, no. 1, pp. 23-48, 1997.
[20] H. Sidenbladh, M.J. Black, and D.J. Fleet, "Stochastic Tracking of 3D Human Figures Using 2D Image Motion," *Proc. European Conf. Computer Vision*, vol.2, pp. 702-718, 2000.
[21] J. Pan, B. Hu, and J.Q. Zhang, "An Efficient Object Tracking Algorithm with Adaptive Prediction of Initial Searching Point," *Lecture Notes in Computer Science,* vol. 4319/2006, pp. 1113-1122, 2006.
[22] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani, "Hierarchical Model-Based Motion Estimation," *Proc. European Conf. Computer Vision*, pp. 237-252, 1992.
[23] S.K. Zhou, R. Chellappa, and B. Moghaddam, "Visual Tracking and Recognition Using Appearance-Adaptive Models in Particle Filters," *IEEE Trans. on Image Processing*, vol. 13, no. 11, pp. 1491-1506, 2004.
[24] A.D. Jepson, D.J. Fleet, and T.F. EI-Maraghi, "Robust Online Appearance Models for Visual Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 10, pp. 1296-1311, 2003.
[25] A.W. Senior, "Tracking with Probabilistic Appearance Models," *ECCV workshop on Performance Evaluation of Tracking and Surveillance Systems*, June 2002, pp. 48-55.
[26] A. Senior et al, "Appearance Models for Occlusion Handling," *Journal of Image and Vision Computing*, vol. 24, no. 11, pp. 1233-1243, 2006.
[27] M.S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Trans. on Signal Processing*, vol. 50, no. 2, pp. 174-188, 2002.
[28] D. Comaniciu, V. Ramesh, and P. Meer, "Kernel-Based Object Tracking," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 25, no. 5, pp. 564-577, 2003.
[29] D. Gatica-Perez, C. Gu, and M.T. Sun, "Semantic Video Object Extraction Using Four-Band Watershed and Partition Lattice Operators," *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 11, pp. 603-618, 2001.
[30] C. Chang, and R. Ansari, "Kernel Particle Filter for Visual Tracking," *IEEE Signal Processing Letters*, vol. 12, no. 3, pp. 242-245, 2005.