Robust Occlusion Handling in Object Tracking

Jiyan Pan
Fudan University
220 Handan Rd., Shanghai 200433, China
jiyanpan@fudan.edu.cn

Bo Hu
Fudan University
220 Handan Rd., Shanghai 200433, China
bohu@fudan.edu.cn

Abstract

In object tracking, occlusions significantly undermine the performance of tracking algorithms. Unlike the existing methods that solely depend on the observed target appearance to detect occluders, we propose an algorithm that progressively analyzes the occlusion situation by exploiting the spatiotemporal context information, which is further double checked by the reference target and motion constraints. This strategy enables our proposed algorithm to make a clearer distinction between the target and occluders than existing approaches. To further improve the tracking performance, we rectify the occlusion-interfered erroneous target location by employing a variant-mask template matching operation. As a result, correct target location can always be obtained regardless of the occlusion situation. Using these techniques, the robustness of tracking under occlusions is significantly promoted. Experimental results have confirmed the effectiveness of our proposed algorithm.

1. Introduction

Object tracking is an important aspect of computer vision and has a wide range of applications. Tracking algorithms can be classified into three categories: point tracking [1,2], kernel tracking [3-7,10], and silhouette tracking [8,9]. This paper focuses on kernel tracking, where an appearance model (or equivalently, a template) is used to represent the target and the geometric information of the target is characterized by affine parameters [10].

For kernel tracking algorithms, one of the toughest challenges comes from occlusions [3,5-7], in which the target is covered by outliers for an uncertain period of time. Failure to detect occluders would lead to significant loss in tracking precision [5] and, more seriously, the infiltration of occluders into the template which typically leads to tracking failure [3]. The situation is further complicated by the following 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.

In the literature a lot of efforts have been devoted to detecting and handling occlusions. A mixture of three distributions is used in [3] to model the observed target appearance, where occluders are characterized by the “lost” component which has a uniform distribution. Another approach declares outlier pixels by examining whether the measurement error exceeds a certain value [5-7]. 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 occlude the target for a long time. In [12], occlusion situation is analyzed by comparing motion characteristics between the target and the image blocks that cannot be well motion-compensated. This algorithm is more effective in detecting occluders because temporal context is utilized. However, error propagation is frequently observed as a consequence of lacking the check by an appearance model as a reference. As for the chicken and egg problem, few solutions are provided in the literature.

This paper proposes a content-adaptive progressive occlusion analysis (CAPOA) algorithm to handle occlusions robustly. Instead of relying on the statistics of the observed target appearance, our algorithm *explicitly* detects the outliers in a progressive manner by using the spatiotemporal context information around the target. The context information is further subject to the scrutiny of the reference target and motion properties as a double check. As a result, the CAPOA algorithm makes a much clearer distinction between the target and the occluder. We solve the aforementioned chicken and egg problem by performing the variant-mask template matching (VMTM), where the non-occluded portion of the target is exploited to align the target from the initial erroneous location to its true location. Using the techniques above, our object tracker is found to be much more robust against various types of occlusions.

The paper is organized as follows. In section 2, we give the overall structure of our proposed algorithm. Section 3 details the CAPOA algorithm. The VMTM method is discussed in Section 4. Experimental results are presented in

1-4244-1180-7/07/$25.00 ©2007 IEEE

Section 5, and Section 6 concludes this paper.

2. Overall structure

Our proposed object tracking algorithm utilizes grayscale features, yet it can readily be adapted to color features. The entire structure of our algorithm is illustrated in Figure 1. The initial target is manually selected by defining the region of interest (ROI). When a new frame comes in, the *approximate* target region (ROI_1) is obtained through the first masked template matching. 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 *non-occluded* part of the template using a Kalman appearance filter [13]. 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.

In our algorithm, template matching is performed by using coordinate transformation to map the estimated template to the frame and finds the frame region that agrees best with the estimated template. Occluders are completely masked out, not just down-weighted [6,7], in finding the target location, because occluders are explicitly detected using the CAPOA algorithm. The masked template matching can be expressed as

$$\hat{\boldsymbol{a}} = \arg\min_{\boldsymbol{a}} \frac{1}{\text{sum}(M)} \sum_{x \in \boldsymbol{\Omega}_T} \left(\frac{I_n[\phi(\boldsymbol{x};\boldsymbol{a})] - \hat{T}(\boldsymbol{x})}{\sigma_E(\boldsymbol{x})} \cdot M(\boldsymbol{x}) \right)^2 , \quad (1)$$

where $\hat{\boldsymbol{a}}$ is the estimated coordinate transformation parameters, I_n represents frame n, $\hat{T}$ denotes the estimated template, σ_E is the expected estimation error obtained in the process of Kalman appearance filtering [13], $\phi(\boldsymbol{x};\boldsymbol{a})$ is an arbitrary coordinate transformation characterized by the parameter $\boldsymbol{a}$ which typically reflects the translation and deformation of the target, $\boldsymbol{\Omega_T}$ is the ensemble of the template pixels in the template coordinate system, M denotes the template mask which is the same size as the template. It 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. Only the translational parameters are involved in the first template matching for stability, while all the parameters are under search in the second template matching.

```
Initialize the outlier map $U_1$ by manually selecting the target region (ROI). $U_1(\boldsymbol{x})=0$ if $\boldsymbol{x}$ belong 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.
For frame index $n = 2,3,\ldots$
    Run $ROI_1 = \text{FTM}(\hat{T}, I_n, M_{n-1})$ to obtain the approximate target region $ROI_1$. FTM denotes “first template matching”. $I_n$ is frame n.
    Run $[U_{prlm}, dummy]=\text{CAPOA}(ROI_1, T_{ref}, I_{n-1}, U_{n-1})$ to obtain the preliminary outlier map $U_{prlm}$. “dummy” means a dummy variable.
    Run $ROI_2 = \text{VMTM}(\hat{T}, I_n, U_{prlm})$ to rectify the target region from $ROI_1$ to $ROI_2$. $U_{prlm}$ is used to generate the variant mask $M_A$.
    Run $[dummy, M']=\text{CAPOA}(ROI_2, T_{ref}, I_{n-1}, U_{n-1})$ to get a new template mask $M'$.
    Run $ROI_3 = \text{STM}(\hat{T}, I_n, M')$ to obtain the final target region $ROI_3$. STM denotes “second template matching”.
    Run $[U_n, M_n]=\text{CAPOA}(ROI_3, T_{ref}, I_{n-1}, U_{n-1})$ to acquire the final outlier map and template mask.
    Update the non-occluded part of $\hat{T}$ and $T_{ref}$ using $ROI_3$, $U_n$ and $M_n$.
END
```

Figure 1: The entire structure of our proposed algorithm.

The update of the estimated template is as follows:

$$\hat{T}(\boldsymbol{x}) \leftarrow \hat{T}(\boldsymbol{x}) + G \cdot \left\{ I_n[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})] - \hat{T}(\boldsymbol{x}) \right\} \cdot M(\boldsymbol{x}) . \quad (2)$$

Here, G is the Kalman gain in the appearance filter [13]. The occluded part of the template is excluded from the update. By updating the template, the tracking algorithm is robust against non-rigid deformation of the target and gradual changes in lighting conditions.

The CAPOA and the VMTM algorithms are highlighted in gray in Figure 1 and will be detailed in Section 3 and Section 4, respectively.

3. Content-adaptive progressive occlusion analysis (CAPOA) algorithm

The overall scheme of the CAPOA algorithm is shown in Figure 2. The function block in gray is further expanded in Figure 3. The two figures will be detailed in the subsequent sub-sections. The CAPOA algorithm takes four inputs: the image region to be analyzed (ROI), the reference target (T_{ref}), the previous frame (I_{prev}), and the previous outlier map (U_{prev}). The two outputs of the CAPOA algorithm are the updated outlier map (U_{out}) and the updated template mask (M_{out}). Their detailed definitions will be given later.

3.1. Progressive scanning of the region of interest

In our algorithm, occlusion detection is based on image blocks, not individual pixels as is done in [3,5-7], because spatial context plays an important role in deciding whether the 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 region of interest (ROI) undergoes multiple scans. In each new scan, the sizes of the blocks under analysis are halved, and we only analyze the blocks within which the occlusion situation has not been decided by the previous scans. The progressive scanning terminates when the occlusion situation of the entire ROI is determined (see Figure 2). Let D_1 and D_2 be the length of the two sides of the ROI, the total number of scans N_S is

$$N_S = \min\{\lfloor \log_2(\min(D_1, D_2)/5)\rfloor,\ 3\}, \qquad (3)$$

so that the minimum size of any block under analysis is 5 and the maximum number of scans is 3. The determination of the occlusion status of a block is illustrated in Figure 3 and described in the sub-sections below.

3.2. Exploiting spatiotemporal context

As is mentioned in Section 1, 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. Any occluder (including those appearing *after* the initialization) that moves from the non-target region into the ROI will be detected. The spatiotemporal context around the target region is thus exploited in this process.

The occlusion situation of a frame is represented by a so called *outlier map*. It is a binary matrix that has a value of 1 where the pixel does *not* belong to the target. For each block in the ROI (referred to as a *current block*), we perform backward motion estimation to find the image block in the previous frame that is the most similar to the current block, 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:

$$\boldsymbol{U}'(\boldsymbol{\omega_b}) = \boldsymbol{U}_{prev}(\tilde{\boldsymbol{\omega}}_{\boldsymbol{b}}), \qquad (4)$$

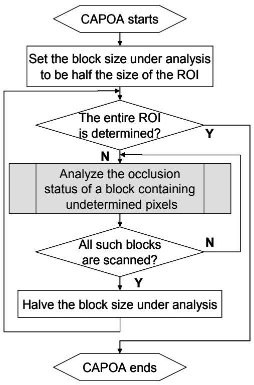
CAPOA starts
Set the block size under analysis to be half the size of the ROI
The entire ROI is determined?
Y
N
Analyze the occlusion status of a block containing undetermined pixels
All such blocks are scanned?
N
Y
Halve the block size under analysis
CAPOA ends

Figure 2: The flowchart of the overall CAPOA algorithm.

where $\boldsymbol{U'}$ denotes the temporary outlier map, $\boldsymbol{U}_{prev}(\tilde{\boldsymbol{\omega}}_{\boldsymbol{b}})$ represents the 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 Figure 3. 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. 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.

3.3. First check using the reference target

The reference target T_{ref} is essentially a scaled version of the target and is updated through incremental interpolation and filtering *at the end of* processing each frame (see Figure 1): if the scale of the target is found to have changed in the 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 in the interpolated reference target is calculated as

$$T_{ref}(n+1) = T'_{ref}(n) + G\cdot\{ROI(n) - T'_{ref}(n)\}\cdot(1-u_n), \qquad (5)$$

where $T_{ref}(n+1)$ is a pixel value of the reference target to be used at frame $n+1$, $T'_{ref}(n)$ denotes the corresponding pixel value of the *interpolated* reference target at frame n, G is the same Kalman gain as is used in the template update, $ROI\ (n)$ is the corresponding pixel value of the ROI, and u_n is the occlusion status of the corresponding pixel in frame n. Since the change in the scale of the target is very small over one frame interval, the incremental interpolation of the reference target allows for smaller interpolation error and enables the

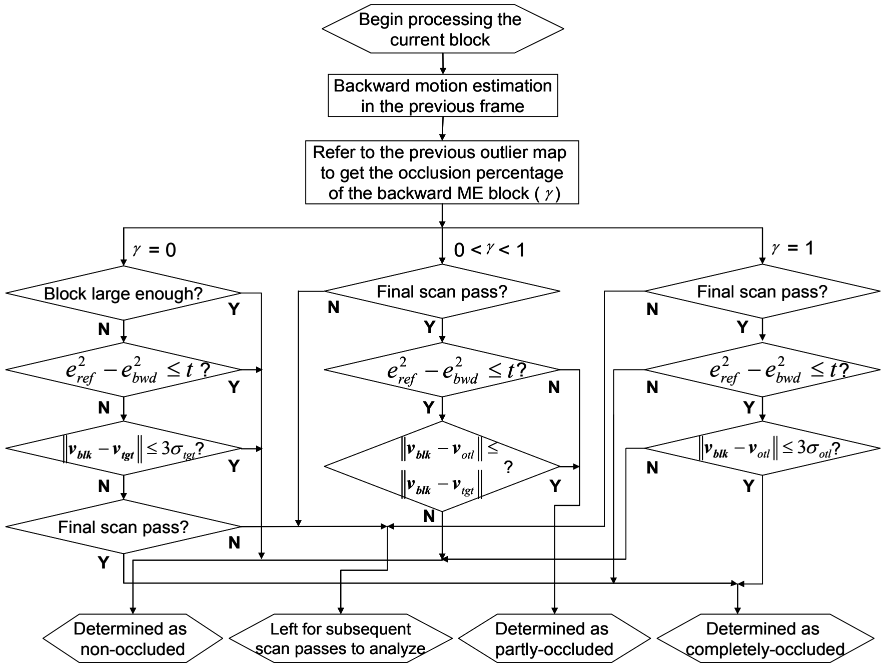
Begin processing the current block
Backward motion estimation in the previous frame
Refer to the previous outlier map to get the occlusion percentage of the backward ME block (γ)
γ = 0
0 < γ < 1
γ = 1
Block large enough?
Y
N
$e_{ref}^2 - e_{bwd}^2 \le t$?
Y
N
$\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{tgt}\| \le 3\sigma_{tgt}$?
Y
N
Final scan pass?
N
Y
Final scan pass?
N
Y
$e_{ref}^2 - e_{bwd}^2 \le t$?
N
Y
$\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{otl}\| \le \|\boldsymbol{v}_{blk} - \boldsymbol{v}_{tgt}\|$?
Y
N
Final scan pass?
N
Y
$e_{ref}^2 - e_{bwd}^2 \le t$?
N
Y
$\|\boldsymbol{v}_{blk} - \boldsymbol{v}_{otl}\| \le 3\sigma_{otl}$?
N
Y
Determined as non-occluded
Left for subsequent scan passes to analyze
Determined as partly-occluded
Determined as completely-occluded

Figure 3: The flowchart of analyzing the occlusion status of a current block.

reference target to contain more details than directly interpolating the template to the size of the current target at each frame. As a result, the reference target is more effective than the template as a benchmark.

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 e^2_{ref} and the latter one as 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 e^2_{ref} is smaller or only slightly larger than 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 e^2_{ref} is significantly over e^2_{bwd}. They are the decision criteria when performing the first check. The problem here is how to adaptively set the threshold of $e^2_{ref} - e^2_{bwd}$ involved in the decision criteria for individual blocks. This threshold is denoted as *t* in Figure 3.

As the the expected estimation error σ_E [see (1)] reflects the general deviation of the observed value of a *target* pixel from the estimated value in the template (or the reference target), it is reasonable to make the threshold dependent on the average σ^2_E of the template pixels related with the uncertain block. In fact, it can be theoretically proved that the expected value of $e^2_{ref} - e^2_{bwd}$ for an uncertain block belonging to the target is

$$\mathrm{E}\left\{e_{ref}^2 - e_{bwd}^2\right\} = \frac{1}{N_{BT}} \sum_{\boldsymbol{x}_t \in \boldsymbol{\Omega}_{BT}} \sigma_E^2(\boldsymbol{x}_t), \qquad (6)$$

where $\boldsymbol{\Omega_{BT}}$ represents the template region corresponding to the uncertain block, and N_{BT} is the number of pixels $\boldsymbol{\Omega_{BT}}$ contains. The proof is omitted here for conciseness.

Combining with the result given by the spatiotemporal context, we set the threshold *t* for an uncertain block located at $\boldsymbol{\omega_b}$ as follows:

$$t(\boldsymbol{\omega}_b) = \mathrm{E}\left\{e_{ref}^2 - e_{bwd}^2\right\}\cdot\left[3 - 2\gamma(\boldsymbol{\omega}_b)\right] = \frac{3 - 2\gamma(\boldsymbol{\omega}_b)}{N_{BT}} \sum_{\boldsymbol{x}_t \in \boldsymbol{\Omega}_{BT}} \sigma_E^2(\boldsymbol{x}_t) \,. (7)$$

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 (7), it can be seen that the *t* in Figure 3 is adaptive to the contents of individual blocks: higher estimation error or lower occlusion percentage increases the value of *t*.

3.4. Second check by motion constraint

For the uncertain blocks that fail to meet the decision criteria in Section 3.3, we exploit motion constraint to further check their occlusion situations. 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 detailed decision criteria are illustrated in Figure 3 based on the discussion above. Here, $\boldsymbol{v}_{blk}$ denotes the motion vector of the uncertain block and is obtained through the backward motion estimation in Section 3.2. $\boldsymbol{v}_{tgt}$ ($\boldsymbol{v}_{otl}$) is the motion vector of the target (occluder), and is estimated by averaging the motion vectors of all the already-determined target (occluded) pixels. σ_{tgt} (σ_{otl}) is the root mean square of the Euclidean distances from the motion vectors of every target (occluded) pixel to the mean motion vector of the target (occluder).

After the occlusion situation of the entire ROI is determined, 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:

$$M_{out}(\boldsymbol{x}) = 1 - U_{out}\{\text{round}[\phi(\boldsymbol{x};\hat{\boldsymbol{a}})]\}, \qquad (8)$$

where round[·] is the operation that rounds the elements of a vector to their nearest integers.

4. Variant-mask template matching (VMTM)

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, not 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.

In order to remedy this problem, we propose the variant-mask template matching (VMTM) algorithm to rectify the target location. 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 Figure 4). 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 Figure 1). However, 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 algorithm.

In the VMTM algorithm, the accurate target location is acquired by performing the following parameter search:

$$\hat{\boldsymbol{a}}_A = \arg\min_{\boldsymbol{a}} \frac{1}{\text{sum}(M_A)} \sum_{\boldsymbol{x}\in\Omega_T} \left(\frac{I[\phi(\boldsymbol{x};\boldsymbol{a})] - \hat{T}(\boldsymbol{x})}{\sigma_E(\boldsymbol{x})} \cdot M_A(\boldsymbol{x};\boldsymbol{a}) \right)^2, \qquad (9)$$

where $\hat{\boldsymbol{a}}_A$ is the rectified transformation parameter vector which defines the rectified ROI (ROI_2 in Figure 1), M_A is a dynamic template mask which varies with the $\boldsymbol{a}$ under test, and is generated from the preliminary outlier map by

$$M_A(\boldsymbol{x};\boldsymbol{a}) = 1 - U_{prlm}\{\text{round}[\phi(\boldsymbol{x};\boldsymbol{a})]\} \qquad (10)$$

where U_{prlm} denotes the preliminary outlier map.

The underlying mechanism of the VMTM algorithm is that in the parameter search performed by (9), the unmasked part of the template and unmasked part of the ROI are always dissimilar *unless* the ROI is located exactly where the target is. This fact is illustrated in Figure 4. For the sake of stability, only translational transformation parameters are involved in the VMTM.

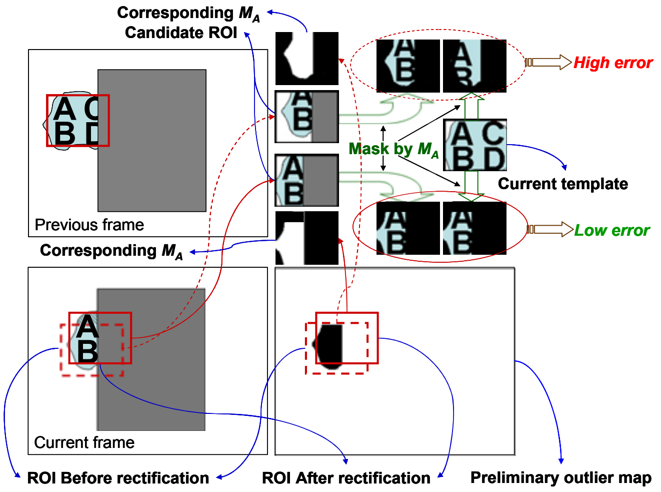

Figure 4: An illustration of the VMTM algorithm. Letters "A", "B", "C" and "D" are marked on the target to indicate different portions of it. The matching error reaches its minimum when the candidate ROI is just the region that the target occupies.

5. Experimental results

We perform experiments on a wide range of video sequences both taken by ourselves and downloaded from the standard datasets available at http://homepages.inf.ed.ac.uk/rbf/CAVIAR and http://www.ces.clemson.edu/~stb/research/headtracker/seq. Those real-world sequences contain various types of targets and different scenarios of occlusions. Cameras do *not* need to be stationary. We classify the occlusion scenarios of all the 30 test sequences into two categories: short-term and long-term occlusions. An occlusion that lasts more than 25 frames is classified as a long-term occlusion here. We compare the performance of our algorithm with that of some state-of-the-art algorithms ([4, 7, 12]). In order to observe the contribution of the VMTM method, the performance of our algorithm *without* the VMTM is also examined. In our experiments, only translational and scaling parameters are involved as the template is adaptive.

The experimental results are listed in Table 1, where P and P' denote our algorithm with and without the VMTM, respectively. It is observed in the experiments that the algorithm of [7] is much more effective in handling short-term occlusions than long-term ones. In the latter case, the outliers penetrate into the appearance model when the occlusion lasts relatively long. This phenomenon exist for all

Table 1: Performances of various object tracking algorithms in dealing with occlusions.

Occlusion Type	[4]	[7]	[12]	P'	P
Short-term (15)	7	12	10	13	14
Long-term (15)	4	0	5	7	13
Total (30)	11	12	15	20	27

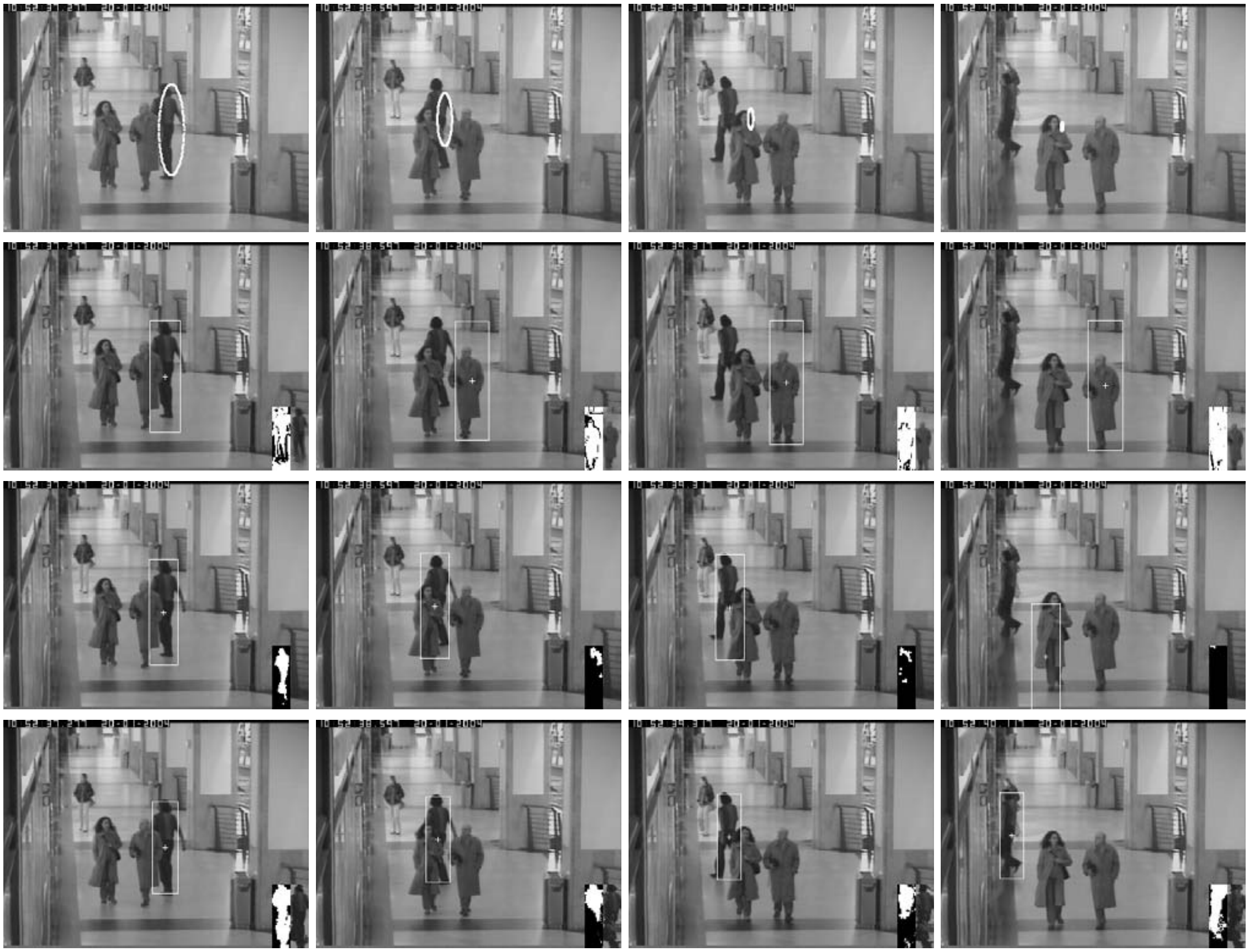

Figure 5: Comparison of different algorithms in handling a challenging long-term occlusion. The four rows from top display the results of the algorithms of [4], [7], [12], and our proposed algorithm, respectively. The four columns from left show frames 1980, 2013, 2032, and 2052, respectively. In the first row, the tracking result is represented by an ellipse. In the remaining rows, the tracking result is indicated by a rectangle with a cross at the center. For rows 2 and 4, the current occlusion situation of the target (or template mask) and the current template are displayed from left in the lower right corner of each image. For row 3, only the current occlusion situation of the target is displayed in the corresponding positions. Occluded pixels are indicated in black.

the other algorithms sharing similar occlusion detection mechanisms (e.g. [3, 5, 6]). The algorithm of [12] has a better performance, but it fails in many sequences as a result of the propagation of the errors in the outlier map and being ineffective in detecting target regions reemerging from behind the other side of occluders. As the algorithm of [4] (Mean Shift) does not possess a scheme to handle occlusions, it has the worst performance. Our proposed algorithm tracks the target very well in almost all the sequences for each type of occlusions. We observe that outliers are always effectively detected by our algorithm, and the integrity of the template is rarely undermined. Without performing the VMTM, our tracking algorithm fails for some sequences that it could have successfully tracked otherwise. In the other sequences for which the tracking still succeeds, it is observed that the tracking accuracy drops when we leave out the VMTM.

In order to provide an intuitive impression of the performance of our proposed algorithm, we show the tracking process of a challenging test sequence in Figure 5. This sequence named “WalkByShop1cor” is taken from the first standard dataset mentioned before. It contains a long-term occlusion, in which a man (target) is severely occluded by two pedestrians for over 50 frames. Since the man is walking, the target itself changes appearance. Some parts of the occluders are similar in color to the target. All the algorithms start tracking from frame 1920 with identical initialization. It can be seen that the algorithm of [4] fails in this sequence, because histogram is not discriminant enough when occlusions occur. The algorithm of [7] performs poorly in distinguishing between the occluders and the target in this

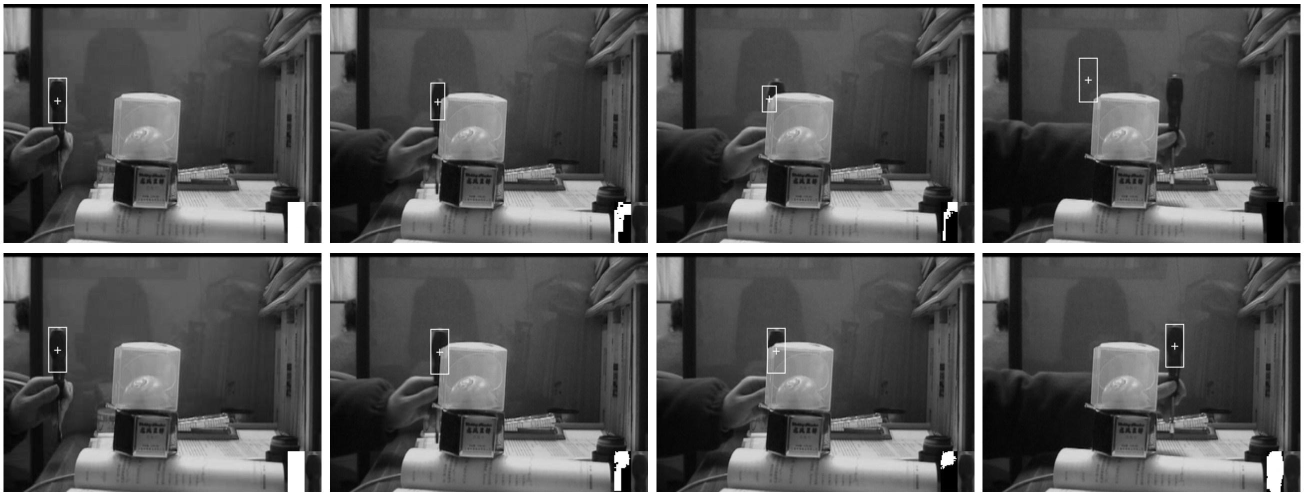

Figure 6: Illustration of the role that the VMTM plays in our algorithm. The first row displays the results of our algorithm without the VMTM. The second row shows the results of our complete algorithm. The four columns from left show frames 2, 27, 29, and 45, respectively. The tracking result is indicated by a rectangle with a cross at the center. The current occlusion situation of the target (or template mask) and the current template are displayed from left in the lower right corner of each image. Occluded pixels are indicated in black.

complex environment. The algorithm of [12] is more effective in detecting occlusions. Nevertheless, as it does not keep an appearance model to check the correctness of the motion-based analysis, a lot of errors occur in the outlier map. Our proposed algorithm has the best performance in analyzing occlusion situations. It also has the capability to discriminate against background outliers (see the template masks of the images in the fourth row). It is worth noting that the outlier map generated by our algorithm is never completely correct (see the template mask of the third image in the fourth row), yet the errors are soon corrected by the double checks in the CAPOA algorithm and they seldom propagate away (see the template mask of the fourth image in the fourth row). If we omit the double checks in the CAPOA algorithm, then we have the problems similar to the ones encountered by the algorithm of [12].

An intuitive example of the role that the VMTM plays in our algorithm is shown in Figure 6, where the test sequence is taken by us. In this sequence, a screwdriver (target) moves behind a box for about 20 frames. It can be seen that without the rectification of the target location by the VMTM, the tracking fails when the target gets occluded rapidly.

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, so that the ground-truth values of the transformation parameters are known in advance and the experiments can be conducted in a controlled manner. The synthetic sequences are generated using the 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. It moves along the diagonal of the image “lake” 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 also varies between 0.5 and 1.0 at a rate of 0.01 per frame. A sample frame in one of the synthetic test sequences is displayed in Figure 7. We measure the tracking errors of the proposed tracking algorithm with and without the VMTM under various target speeds. Tracking error is measured by the mean Euclidean distances between the estimated values and the ground truth values of the transformation parameter vectors at all frames.

The experimental result is illustrated in Figure 8. It can be observed that when the target speed relative to the occluder is low, the proposed tracking solution achieves high tracking accuracy no matter whether the VMTM is employed or not. 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 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.

Finally, we discuss the failure modes of our algorithm. 1) When complete occlusions occur, our algorithm will fail. 2) If a part of an occluder has *exactly* the same appearance as a

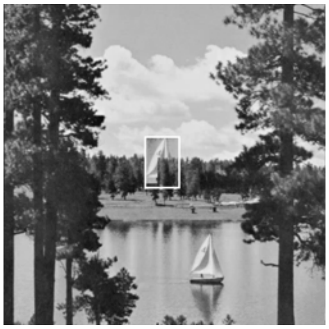

Figure 7: A sample frame in a synthetic test sequence. The partly-occluded target is highlighted by a white rectangle.

nearby part of the target, our algorithm will not be able to detect it. Although [9] can tackle the first problem, it cannot readily be adapted to kernel tracking. Solving the second problem might need more prior information. The solutions to the two problems are left for future research.

6. Conclusion

In this paper, we propose an object tracking algorithm which demonstrates high robustness against occlusions. This is achieved by effectively analyzing the occlusion situation to generate proper template mask and rectifying initial erroneous target location caused by occlusions.

In our proposed algorithm, we utilize the spatiotemporal context to analyze the occlusion situation of the target. The analysis results are further checked by the reference target and the motion information according to content-adaptive thresholds. The occlusion analysis is performed in a progressive manner, so that we achieve high reliability and high resolution simultaneously. Our proposed algorithm also performs variant-mask template matching (VMTM) to remove the error of the target location introduced by performing template matching when the template mask reflecting the current occlusion situation is unavailable yet. In the process of the VMTM, the non-occluded portion of the target serves as a benchmark for the alignment of the target location.

We verify the effectiveness of our proposed algorithm by conducting experiments on a wide range of real-world video sequences downloaded from the standard datasets and taken by ourselves. Synthetic test sequences are also employed to evaluate the role of the VMTM quantitatively. The experimental results have shown that our approach outperforms many state-of-the-art algorithms when faced with different types of occlusions.

References

[1] C. Rasmussen, and G. Hager. Probabilistic data association methods for tracking complex visual objects. IEEE Trans. on Pattern Analysis and Machine Intelligence, 23(6):560–576, 2001.

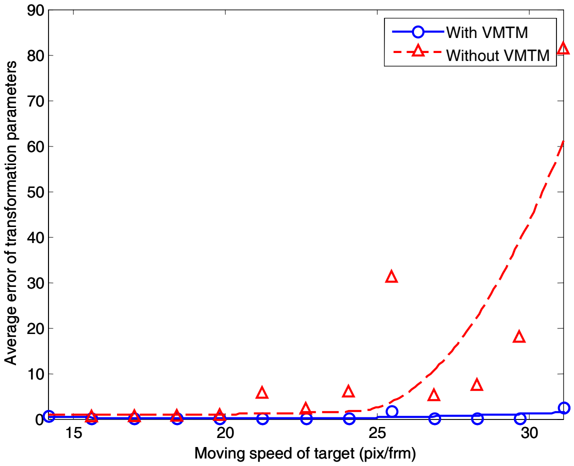

Figure 8: The plot of tracking error against target speed when applying our algorithm with and without the VMTM.

[2] C. Hue, J.L. Cadre, P. Prez. Sequential Monte Carlo methods for multiple target tracking and data fusion. IEEE Trans. on Signal Processing, 50(2):309–325, 2002 .

[3] A.D. Jepson, D.J. Fleet, and T.F. EI-Maraghi. Robust online appearance model for visual tracking. IEEE. Trans. on Pattern Analysis and Machine Intelligence, 25(10):1296-1311, 2003.

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

[5] S.K. Zhou, R. Chellappa, and B. Moghaddam. Visual tracking and recognition using appearance-adaptive models for particle filters. IEEE Trans. on Image Processing, 13(11):1491-1506, 2004.

[6] H.T. Nguyen, M. Worring, and R. van den Boomgaard. Occlusion robust adaptive template tracking. Proc. IEEE Int'l Conf. Computer Vision, 1:678-683, 2001.

[7] 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, 26(8):1099-1104, 2004.

[8] Y. Chen, Y. Rui, and T. Huang. Jpdaf based HMM for real-time contour tracking. Proc. IEEE Conf. on Computer Vision and Pattern Recognition, 1:543–550, 2001.

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

[10] S. Baker, and I. Matthews. Lucas-Kanade 20 years on: a unifying framework. Int'l Journal Computer Vision, 53(3): 221–255, 2004.

[11] I. Matthews, T.Ishikawa, and S. Baker. The template update problem. IEEE Trans. on Pattern Analysis and Machine Intelligence, 26(6):810–815, 2004.

[12] K. Hariharakrishnan, and D. Schonfeld. Fast object tracking using adaptive block matching. IEEE Trans. on Multimedia, 7(5):853-859, 2005.

[13] 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, 1:365-368, 2001.