Robust Object Tracking via Key Patch Sparse Representation

Abstract—Many conventional computer vision object tracking methods are sensitive to partial occlusion and background clutter. This is because the partial occlusion or little background information may exist in the bounding box, which tends to cause the drift. To this end, in this paper, we propose a robust tracker based on key patch sparse representation (KPSR) to reduce the disturbance of partial occlusion or unavoidable background information. Specifically, KPSR first uses patch sparse representations to get the patch score of each patch. Second, KPSR proposes a selection criterion of key patch to judge the patches within the bounding box and select the key patch according to its location and occlusion case. Third, KPSR designs the corresponding contribution factor for the sampled patches to emphasize the contribution of the selected key patches. Comparing the KPSR with eight other contemporary tracking methods on 13 benchmark video data sets, the experimental results show that the KPSR tracker outperforms classical or state-of-the-art tracking methods in the presence of partial occlusion, background clutter and illumination change.

Index Terms—visual object tracking, patch sparse representation, occlusion prediction scheme, template update, particle filter

I. INTRODUCTION

OBJECT tracking, which plays an indispensable role in motion analysis, activity recognition, video surveillance and traffic monitoring, continually attracts attention in the computer vision community. Though numerous tracking methods have been proposed for object tracking in the past decades, it still remains a challenging problem because of many environmental factors in video datasets, such as illumination variation, background clutter and occlusions.

Generally, object tracking methods can be classified into discriminative and generative methods. The discriminative methods [1–8] aim to discriminate the target from the background by training a classifier according to the information from both the target and the background. The generative methods [9–18] aim to search for regions, which are extremely similar to the target, based on templates or subspaces.

In discriminative methods, support vector tracking (SVT) [7] is proposed by integrating Support Vector Machines (SVM) [6] into an optic-flow-based tracker and maximizing the classification score. Multiple instance learning (MIL) [2] is trained with instances, which are included in the bags. The MIL problem can be cast as a maximum margin problem and solved by SVM. P-N learning (PN)[3], which is guided by positive and negative constraints on the unlabelled data, is proposed to exploit the underlying structure of positive and negative samples to learn effective classifiers for object tracking. However, all of these existing methods based on classification rely on a heuristic intermediate step for producing labelled binary samples, which is often a source of error during tracking.

Therefore, a new adaptive tracking-by-detection framework based on structured output prediction [8] is proposed, which is able to avoid the need for an intermediate classification step and incorporate image features and kernels.

In generative methods, it is necessary and difficult to solve partial occlusion. Adam et al. [11] adopt a patch-based tracking method to allow every patch vote on the possible positions and scales of the target and then locate it by combining the vote maps of the multiple patches. The patch-based tracking methods can solve the partial occlusion to some extent. The sparsity-based tracking methods [16, 19–21], inspired by face recognition [22, 23] play an important role in object tracking. Mei et al. [16] first formulate the tracking problem as a sparse approximation problem. And then Wu et al. [20] propose a data fusion approach via sparse representation, where a flexible framework is provided and the information from different data sources can be easily integrated. The incremental visual tracking (IVT) [15] is robust to illumination and pose variation but sensitive to partial occlusion and background clutter. Naturally, inspired by [16], [11] and [15], visual tracking via adaptive structural local sparse appearance model (ASLSA) [18] is proposed that uses the patch sparse representations to deal with the partial occlusion to some extent.

Although the aforementioned methods have achieved a prominent performance in many cases, their performances can also be further improved since these methods do not consider the local information or do not consider the difference among the patches sampled from a target candidate and treats them equivalently. To this end, we propose a robust tracker based on key patch sparse representation (KPSR). The proposed method is based on patches and treats them differently. Its main contributions are twofold. First, we propose a selection criteria based on the key patch according to occlusion prediction and patch location. Second, we propose a contribution factor design for key patch and non-key patch regions, and emphasize the contribution of key patch for robust tracking.

The remainder of this paper is organized as follows. In Section II, we first introduce patch sparse representation to get the score of each patch, and then propose the key patch sparse representation for tracking in Section III. In Section IV, we propose the robust tracker based on key patch sparse representation. In Section V, we make quantitative and qualitative evaluations, and compare the KPSR with eight tracking methods including the classical and state-of-the-art tracking methods. Finally, conclusions are drawn in Section VI.

II. PRELIMINARY

In this section, we first give the general formulation of patch sampling and then generalize the process of patch sparse representation.
A. Patch Sampling

Given the target candidate with $B \times G$ pixels, the patch with $b \times g$ pixels, the column step length $S_c$ (an integer), $0 < S_c \leq b$ and the row step length $S_r$ (an integer), $0 < S_r \leq g$, the desired patches are sampled sequentially and used to represent the complete structure of the target candidate (see Fig. 1). In detail, if $mod(B - b, S_c) = 0$ and $mod(G - g, S_r) = 0$, we can get $(\frac{B-b}{S_c}+1)$ patches in row orientation and $(\frac{G-g}{S_r}+1)$ patches in column orientation; respectively. Let $r = \frac{B-b}{S_c} + 1$ and $c = \frac{G-g}{S_r} + 1$, then, the number of total patches of a target candidate is $N = rc$. Now, the serial number of each patch $k, k \in \{1,2,\ldots,N\}$ can be denoted as $k = (j-1)r+i$ where $i, i = 1,2,\ldots,r$ is row index and $j, j = 1,2,\ldots,c$ is column index. To sum up, given the target size, the sampled patch size, and the step length; key patch sampling will result in $N$ patches.

Note that patch sampling with different step length differ. When $S_c = b$ and $S_r = g$, it becomes the unoverlapped patch sampling, otherwise it becomes the overlapped patch sampling. Usually, the overlapped patch sampling is adopted because it is able to better capture local structure. In this paper, the bounding boxes of the target candidates are firstly resized to $32 \times 32$ pixels, i.e., $B = G = 32$. The patch size is $b \times g = 16 \times 16$ pixels, and the step length is $S_c = S_r = 8$, which will lead to an overlapped patch sampling strategy and finally get $N = 9$ overlapped patches.

In [11], the authors divide the target bounding box into several unoverlapped vertical and horizontal patches. On the contrary, our patch sampling strategy is similar to that of [11, 18], which utilizes an overlapped patch sampling strategy, thus can keep more spatial structural information between the adjacent patches. Finally, we should notice that the number of patches depends more on the experience and the consideration of the balance between accuracy and speed.

B. Patch Score via Sparse Representation

First, getting a dictionary is necessary. We manually label the target in the first frame and use K-dimensional tree (KD-tree) [24] to track the target from the second frame to the $n$-th frame. After we use the tracked target up to $n$ frames to initialize a template set $\hat{T}_n = \{T_1,T_2,\ldots,T_n\}$, where $T_i$ with $B \times G$ pixels is obtained by the tracked target in the $i$-th frame and $n$ is the template number in template set. Then, each template in $\hat{T}_n$ is divided into $N$ patches with a spatial layout (see Fig. 1), and these patches are assembled into a patch-dictionary $\hat{D} = \{D_1,D_2,\ldots,D_{n\times N}\} \in \mathbb{R}^{d\times(n\times N)}$, where $d = b \times g = 256$ is the dimension of each patch after turning into vector.

Next, the target is tracked in the $n+1$-th frame. $M$ target candidates are sampled surrounding the tracked target in the $n$-th frame. For a target candidate, we divide it into $N$ patches and turn them into vectors, which are denoted as $Y = \{y_1,y_2,\ldots,y_N\}$. With sparse representation, the candidate patch $y_k \in \mathbb{R}^{d\times 1}$ can be represented by $\hat{D}$, and the corresponding coefficient vector $z_k$ can be obtained by solving the following Lasso problem using the corresponding Lasso [25–27] method,

$$\min_{z_k} \| y_k - Dz_k \|^2_2 + \lambda \| z_k \|_1, \quad k = 1,2,\ldots,N$$

$$s.t. \quad z_k \geq 0,$$

where the vector $z_k \in \mathbb{R}^{(n\times N)\times 1}$ is the corresponding sparse coefficients of $y_k$, and $z_k \geq 0$ means each element of $z_k$ is nonnegative. According to the different templates, $z_k$ is divided into $n$ group vectors, i.e., $z_k = [z_k^{(1)},z_k^{(2)},\ldots,z_k^{(n)}]$. Here, $z_k^{(i)} \in \mathbb{R}^{N\times 1}$ means the $i$-th group vector of $z_k$, $i = 1,2,\ldots,n$. Then, we compress $z_k$ and get $v_k \in \mathbb{R}^{N\times 1}$ as follows,

$$v_k = \frac{1}{C} \sum_{i=1}^{n} z_k^{(i)}, \quad k = 1,2,\ldots,N,$$

where $C$ is a normalization constant. Note that a square matrix $V$ is formed by $\{v_1,v_2,\ldots,v_N\}$.

In the same way, all $M$ target candidates get the corresponding $M$ square matrices. In order to distinguish between different target candidates, $v_k \in \mathbb{R}^{N\times 1}$ is denoted as the coefficient vector of the $k$-th patch for the $l$-th target candidate (patch $k$ in $l$), $l = 1,2,\ldots,M$, and $v_{kl}(k$-th element of vector $v_k$) is selected as the patch score of patch $k$ in $l$. The target candidate $l$ with the maximum sum is chosen as the target in the $n+1$-th frame as follows,

$$\hat{E}_l = \max_{l} \{ \sum_{k=1}^{N} v_{kl} \}$$

III. KEY PATCH SPARSE REPRESENTATION

Unfortunately, patch sparse representation in [18] does not consider the different contributions among these patches, and this may result in the drift when partial occlusion or some background information exist in the bounding box. In view of this, we propose key patch sparse representation to reduce the effect of occlusion or background clutter. Generally, the target lies in the center of the bounding box including target information and little background information. Therefore, it is reasonable that the middle patch (such as the green box in the right subfigure of Fig. 1) should account for a larger contribution and the peripheral patch (such as the red box in the right subfigure of Fig. 1) accounts for a smaller contribution. On the other hand, when occlusions exists, the corresponding patch should account for a smaller contribution. Therefore, our idea is to select the key patch according to the location and occlusion case of each patch, and then design the patch’s contribution factor for the key patch and non-key patch.

A. Selection of Key Patch

Consider that the target usually lies in the middle of the bounding box, we select the middle patch as key patch. In the below, we discuss how to select the key patch when the tracked target suffers from occlusion.
Input Prediction Scheme: We typically observe that occlusion phenomenon results from the background information. Here, background information is regarded as the image information except for the target information. That is to say, occlusion happens if some background information enters the bounding box. Fig. 2 gives an occlusion phenomenon, where we can observe that the shelter (i.e., a book) gradually enters the bounding box. Taking the 20-th frame for example, we first make the inner patch samples and define the patches in a row as a positive bag. Besides the inner patch sampling, we adopt the outer patch sampling that is to sample patches surrounding the bounding box. Taking Fig. 2 for example, we first make the inner patch samples and define the patches in a row as a positive bag. After we get the positive and negative bags which are used to label its including patches, MIL&SVM [6] is used to train the patches and a classifier (patch trainer) is obtained. With the classifier, we can predict the occlusion case of each patch, denoted by a binary indicator vector \( \delta_k, k = 1, 2, \ldots, N \). If patch \( k \) is not occluded, \( \delta_k = 1 \), otherwise \( \delta_k = 0 \). Therefore, when the current tracked target suffers from occlusions, we can find those occluded patches according to \( \delta_k = 0, k = 1, 2, \ldots, N \).

It is worth noting that the initial classifier is formed by exploring the tracked target across the initial \( n \) frames. As the appearance of the target changes, the classifier update is necessary. In this paper, the classifier is updated every \( \theta \) frames when the target does not have a severe occlusion.

### B. Contribution Factor Design

After key patch is selected, we propose a contribution factor design that assigns different contribution factors for the sampled patches and emphasizes the contribution of key patch. In order to keep consistence with occlusion indicator \( \delta_k \), we use \( \omega_k, k = 1, 2, \ldots, N \) to represent the contribution of total \( N \) patches. The contribution factor of patch \( k \), \( \omega_k \), is defined as follows,

\[
\omega_k = 1 + \delta_k e^{-\beta (|i - \frac{k}{c}| + |j - \frac{i}{r})}, \quad i = 1, 2, \ldots, r, \quad j = 1, 2, \ldots, c
\]

where \( \delta_k \) means the occlusion indicator of patch \( k \), \( \beta \) is a constant, \( r \) is the patch’s number in a row and \( c \) is the patch’s number in a column.
When the target does not suffer from occlusion (i.e., all total \( N \) patches do not suffer from occlusion), we design the contribution factor as \( \omega_k = 1 + e^{-\beta(i-\frac{l-1}{2})+j-\frac{l-1}{2})}, k = 1, 2, \ldots, N \), where the term \( e^{-\beta(i-\frac{l-1}{2})+j-\frac{l-1}{2})} \) indicates that different patch locations have different contribution factors. In detail, the patch (i.e., \( i = \frac{l+1}{2} \) and \( j = \frac{l+1}{2} \)) has a largest contribution factor \( 2 \). The peripheral patch (e.g., \( i = 1 \) and \( j = 1 \)) have such a contribution factor that is smaller than \( 2 \) and larger than \( 1 \). When the target suffers from a partial occlusion, the occluded patch \( k \) has \( \delta_k = 0 \) and hence its contribution factor is \( \omega_k = 1 \). When the target suffers from a complete occlusion, the contribution factor of each patch becomes \( 1 \). To sum up, the contribution factor of each patch considers not only its location but also its occlusion case. In this way, the contribution of the middle and unoccluded patch is emphasized. Fig. 3 gives the overview of key patch sparse representation.

IV. A ROBUST TRACKER BASED ON KEY PATCH SPARSE REPRESENTATION

We first combine key patch sparse representation with particle filter to construct the objective function of the tracker. Second, we give a template update to adapt to the appearance change of the target. Finally, we discuss the robust tracker under three cases including no occlusion, partial occlusion and complete occlusion.

A. Objective Function

We use the status variable \( E_i \) to represent the location and shape of the target to be tracked in the \( t \)-th frame and \( A^{12} = \{ A^1, \ldots, A^7 \} \) to represent the observation set of the target from the first frame to the \( t \)-th frame. Target tracking is to estimate a posterior probability \( p(E^t|A^{12}) \) that can be written as follows,

\[
p(E^t|A^{12}) = p(A^t|E^t) \int p(E^t|E^{t-1})p(E^{t-1}|A^{12-1})dE^{t-1},
\]  

(5)

\( p(A^t|E^t) \) means the observation model in the \( t \)-th frame and describes the similarity between a target candidate and the target templates. \( p(E^t|E^{t-1}) \) means the motion model in the successive frames and describes the temporal correlation of the target state. \( E^t = (x, y, \theta, s, \beta, \phi) \) is consisted of six parameters of the affine transformation, where \( x, y, \theta, s, \beta, \phi \) denote 2D translations, rotation angle, scale, aspect ratio and skew, respectively. In this paper, the motion model \( p(E^t|E^{t-1}) \) is modeled as \( p(x_t|x_{t-1}) = N(x_t|x_{t-1}, \sigma) \), where \( \sigma \) is a diagonal covariance matrix and the diagonal elements are the variances of the affine parameters. For each target candidate in the particle filter framework, estimating the posterior probability \( p(E^t|A^{12}) \) is converted into maximizing \( p(A^t|E^t) \).

In our tracker based on key patch sparse representation, we replace \( p(A^t|E^t) \) with \( p(A^t|\delta^t, E^t) \). In detail, we use \( E_i \) to represent the state of the target candidate \( l \) in the \( t \)-th frame and \( E_i^t \) to represent the state of patch \( k \) in \( l \), \( l = 1, 2, \ldots, M \). Then, for each state of target candidate \( l \), we have \( p(A^t|\delta^t, E^t) = \prod_{k=1}^N p(A^t|\delta^t, E_i^t) \), where \( \delta^t \) is the occlusion prediction indicator of patch \( k \) in \( l \). Without loss of generality, we remove the frame index \( t \) and have \( p(A^t|\delta^t, E_t) = \prod_{k=1}^N p(A^t|\delta^t, E_k^t) \). Consequently, our objective function \( \hat{E}_t \) can be written as

\[
\hat{E}_t = \max_l p(A^t|\delta_t, E_t).
\]  

(6)

After taking the logarithm, Eq. (6) becomes

\[
\hat{E}_t = \max_l \{ \sum_{k=1}^N \log p(A_k^t|\delta_k, E_t) \},
\]  

(7)

where \( p(A_k^t|\delta_k, E_t) \) means the observation likelihood of patch \( k \) in \( l \). Let \( p(A_k^t|\delta_k, E_t) \propto e^{\delta_k e^{\alpha_k}} \). Then, our objective function is finally defined as

\[
\hat{E}_t = \max_l \{ \sum_{k=1}^N \omega_k v_k \},
\]  

(8)

where \( \omega_k = 1 + \delta_k e^{-\beta(i-\frac{l-1}{2})+j-\frac{l-1}{2})} \) denotes the contribution factor of patch \( k \) in \( l \) and \( v_k \) means the score of patch \( k \) in \( l \).

B. Template Update

It is necessary to update the templates in tracking, because fixed templates can not capture the appearance change of the target. IVT [15] is proposed to update both eigenbasis and mean to faithfully model the appearance change of the target. Although IVT is robust to illumination and pose variation, it is sensitive to partial occlusion. The template update method in [18], which combines subspace learning with sparse representation, is proposed to deal with partial occlusion,

\[
p = Uq + e = [U \ I][q \ e]^T.
\]  

(9)

where \( p \) denotes the observation vector, \( U \) is the matrix composed by eigenbasis vectors, \( q \) is the coefficient of eigenbasis vectors, and \( e \) indicates the pixels in \( p \) that are occluded. Let \( \hat{U} = [U \ I] \) and \( \hat{q} = [q \ e]^T \), assuming the error caused by occlusion is sparse, Eq. (9) can be solved by

\[
\min_q \| p - \hat{U} \hat{q} \|^2 + \lambda \| \hat{q} \|_1,
\]  

(10)

where \( \lambda \) is the regularization parameter. The goal of Eq. (10) is to update \( Uq \) into the template set.

Because the number of templates is fixed, old template sets need to discarded low-weight templates for balance. Usually, the selection of the discarded template is complied with the rationale that the earlier tracking results are more accurate and should be stored longer than the latter tracking results. In detail, a cumulative probability sequence \( \{0, \frac{1}{2^0-1}, \frac{1}{2^1-1}, \frac{1}{2^2-1}, \ldots, 1\} \) is first generated and its each element means the update probability from the first template to the \( n \)-th template. Then, a random number is generated according to the uniform distribution on the unit interval \([0,1]\). According to the random number, the corresponding template is discarded.

To some extent, sparse representation can avoid that the shelter is updated into template set. However, the shelter could possibly be updated into template set when the target suffers from a large occlusion. In view of this, we introduce occlusion ratio \( r_{oc} \) to describe the occlusion degree, which is denoted as,
The template is updated or not. Algorithm 1 gives the procedure by 9 patches in different color (shown in the second row of our prediction scheme, the predicted result is composed unoverlapped ones, see the most left part of Fig. 4. According we spread the overlapped patches’ contribution factors as the patch under three cases, namely the case without occlusion, partial occlusion, and complete occlusion, as shown in Fig. 4.

We regard \( r_{occ} \leq \eta \) as a constraint to determine whether the template is updated or not. Algorithm 1 gives the procedure of template selection.

**Algorithm 1. Template Selection**

**Input:** Old template set \( \hat{T}_{f-1} \) including \( n \) templates, observation vector \( p \), eigenbasis vectors \( U \), occlusion ratio \( r_{occ} \), threshold \( \eta \), the current frame \( f \) (\( f > n \));

1. Take \( M \) candidates surrounding the \( f \)-th frame, and use (Eq. (1)) to obtain patch score (see Section II-B);
2. Use Eq. (8) to ascertain the optimal candidate \( \hat{E}_f \) as the target in the \( f \)-th frame;
3. Update template to get \( \hat{T}_f \);
   - If \( \text{mod}(f,5) = 0 \) and \( r_{occ} \leq \eta \)
     - Generate a random number between 0 and 1 and decide the discarded template \( T_{d_f} \), then \( \hat{T}^{\text{disc}}_f = \hat{T}_{f-1} - T_{d_f} \);
     - Solve Eq. (10) to obtain \( q \);
     - Add \( p = Uq \) into template set \( \hat{T}^{\text{disc}}_f \) and output the new template set \( \hat{T}_f = \hat{T}^{\text{disc}}_f + p \);
   - Else \( \hat{T}_f = \hat{T}_{f-1} \);
   - End

**Output:** New template set \( \hat{T}_f \).

**C. Discussion**

In this section, we visualize the contribution factor of each patch under three cases, namely the case without occlusion, partial occlusion, and complete occlusion, as shown in Fig. 4. Here, to facilitate the display of patches’ contribution factors, we spread the overlapped patches’ contribution factors as the unoverlapped ones, see the most left part of Fig. 4. According to our prediction scheme, the predicted result is composed by 9 patches in different color (shown in the second row of Fig. 4).

**The case without occlusion.** Under the case without occlusion, in theory, we have \( \delta_k = 1, \omega_k = 1 + e^{-\beta (|\hat{t}_i - t_i| + |\hat{t}_j - t_j|)}, k = 1,2,\ldots,N \), and \( r_{occ} = 0 \). Therefore, the prediction result should be in a ‘+’ style theoretically. Experimentally, from Fig. 4(a), we can see that our method only predict the second patch wrong (corresponding to the head part of the target). Unlike ASLSA, our tracker differently treats the patch according to its location. Therefore, our tracker can restrain the importance of the non-key patch while ASLSA can not.

**The case with partial occlusion.** Under the case with partial occlusion, in theory, we have different values of \( \delta_k \),
\( \omega_k = 1 + \delta_k e^{-\rho (|r_k^t - r_k^{t-1}| + |j_k - j_k^{t-1}|)}, k = 1, 2, \ldots, N, \) and \( 0 < r_{occ} < 1. \) In this case, we believe that the unoccluded patch is key patch (i.e., \( \delta_k = 1 \)), and should give a larger contribution factor for the key patch. Taking Fig. 4(b) for example, we can see that the right part of the man with black clothes is occluded by the red man. Therefore, in this case, the theoretical prediction result should be in a \( \ell^t \)-style. Experimentally, from the down figure of Fig. 4(b), we can see that our method only predict the second patch wrong (corresponding to the head part of the target). Therefore, our tracker can better deal with partial occlusion than ASLSA to some extent.

The case with complete occlusion. Under the case with complete occlusion, in theory, we have \( \delta_k = 0, \omega_k = 1, k = 1, 2, \ldots, N, \) and \( r_{occ} = 1. \) \( r_{occ} = 1 \) indicates that each patch is occluded. From the upper figure of Fig. 4(c), we can see that the red man occlude the target completely. In this case, the theoretical prediction result should be that all patches are occluded. Obviously, our method predict them with a higher accuracy from the down figure of Fig. 4(c). Considering the occluded target is not supposed to update into template set, our tracker is able to reject the occluded target to update into the template set while ASLSA can not. Therefore, our tracker may have a better tracking than ASLSA.

All the above three cases are involving a greater distinction between the target and the shelter. In these cases, our tracker can obtain a more robust and stable results than ASLSA in most of cases (see the section of Experiments). However, when the target is occluded by a similar disruptor (i.e., the shelter is very similar with the target), our key patch prediction scheme will be invalid (see Fig. 4(d)).

V. EXPERIMENTS

A. Implementation Details and Datasets

Our tracker based on key patch sparse representation (KPSR) is implemented in MATLAB and runs at around 4.4 frames per second (fps) on a PC with an Intel 3.6 GHz Dual Core CPU and 4GB memory, which is slower than ASLSA \((\approx 8 \text{ fps})\) while faster than \( \ell_1 \) tracker \((\approx 2 \text{ fps})\). In the experiments, we manually label the location of the target in the first frame for each dataset and set \( \lambda = 0.01, n = 15, N = 9, M = 600, \eta = 0.4, \tilde{\beta} = 1 \) and \( \theta = 13. \)

We evaluate the performance of KPSR on thirteen challenging datasets, which include Faceocc1, Faceocc2, Singer, Car4, Car11, Stone, Board, Woman, Face, Caviar1, Caviar2, Caviar3 and DavidIndoor. For each dataset, we resize the bounding box to 32 \( \times \) 32 pixels and sample patches with the patch size 16 \( \times \) 16 pixels and the step length 8 pixels.

B. Evaluation

We compare KPSR with eight tracking methods, i.e., incremental learning visual tracking (IVT) method [15], \( \ell_1 \) tracker [16], STRUCK [8], visual tracking decomposition (VTD) method [13], P-N learning (PN) tracker [3], multiple instance learning (MIL) tracker [2], FRAG [11], and ASLSA [18]. IVT, \( \ell_1 \), MIL and FRAG methods are classical and the remaining ones are state-of-the-art. In order to make the comparisons objective and persuasive, we obtain the results of these tracking methods by running the source codes provided by their authors.

1) Quantitative Evaluation: We employ two evaluation criteria, i.e., the position center error (in pixels) and the Pascal visual object classes (Pascal VOC) overlap criterion [18, 28], to quantitatively evaluate the performance of KPSR. In detail, given the tracked target and its ground truth, overlap rate \( OR \) and center error \( CE \) can be denoted as \( OR = \frac{R_T \cap R_G}{R_T \cup R_G} \) and \( CE = \text{norm}(C_T - C_G) \), respectively. Here \( R_T \), \( C_T \) are used to represent the region and the center of the tracked target, and \( R_G, C_G \) are used to represent the region and the center of the ground truth. It should be noted that only \( CE \) or \( OR \) can not ensure the accuracy of the tracking result. And we believe both \( CE \) and \( OR \) should be used for tracking evaluation. Fig. 5 shows the comparison results between eight tracking methods and our method using the position center error criterion and the average position center error is listed in Table I. For greater clarity, Fig. 5 removes the results of some tracking methods with large center error. Taking Car4 dataset for example, we remove two curve lines of MIL and FRAG. This is because these two methods deviate from the correct location after a few frames. Similarly, we remove STRUCK, VTD and MIL methods on Face dataset, IVT, L1, PN, MIL, FRAG and VTD methods on Woman dataset, L1 method on Caviar1 dataset, VTD, MIL, IVT, STRUCK and PN methods on Caviar3 dataset, MIL and FRAG methods on Car11 dataset, VTD, MIL and FRAG methods on Stone dataset, L1 method on Board dataset, and remove L1 and PN methods on Singer dataset. Fig. 6 shows the comparison results between eight tracking methods and our method using the Pascal VOC overlap criterion and the average overlap error is listed in Table II. From these figures and tables, we can clearly see that KPSR is the best method on Faceocc1, Faceocc2, Singer, Stone, Woamn, Face, Caviar1, Caviar2 and Caviar3 datasets. KPSR and ASLSA yield the similar results on DavidIndoor dataset, KPSR and IVT yield the similar results on Car4 dataset, and KPSR and STRUCK also yield the similar results on Car11 and Board datasets.

2) Qualitative Evaluation: Occlusion case. Fig. 7 demonstrates how accurate and robust KPSR performs when the target undergoes a heavy or long-time partial occlusion. In the Woman dataset, the woman is occluded when she passes by the black car window. MIL, IVT and FRAG suffer from a drift in this case because the color of the car window is extremely similar with that of the trousers of the woman. However, KPSR, STRUCK and ASLSA are more robust and stable through the whole dataset, this is because KPSR can solve it by increasing the contribution factor for the key patch, STRUCK method benefits robustness to noise by using a kernelized structured output SVM and ASLSA can reduce the partial occlusion to some extent by patch sampling and sparse representation. In the Faceocc1 dataset, the human’s face is occluded by one book. ASLSA fails to make a correct tracking during the 538-th frame to the 878-th frame, this is because the occlusion caused by this book not only occupies a large region but also lasts for a long time. However, KPSR is able to make a better tracking, which attributes to occlusion prediction scheme and the adding constraint of \( r_{occ} \leq \eta \) in
Fig. 5. Quantitative comparisons between KPSR and eight trackers in terms of position center error (in pixels).
Fig. 6. Quantitative comparisons between KPSR and eight trackers in terms of average overlap errors.
template update. In the Faceocc2 dataset, both KPSR and ASLSA can work well, this is because the occlusion has a short time and particle filter functions it.

**Illumination change case.** Fig. 8 demonstrates how accurate and robust KPSR performs when the target undergoes a large illumination variation. More specific, in the Singer dataset, many methods suffer from a drift when the target undergoes a large illumination variation. For example, the PN method drifts away in the 142-th frame and recovers a correct tracking due to its global search function. L1 also drifts away in the 142-th frame and recovers a correct tracking. This is because the template update in L1 can not capture the appearance variation of the target. However, KPSR, ASLSA and IVT have a more robust tracking. This is because the template update in both KPSR and ASLSA use IVT method and hence are robust to illumination. In the DavidIndoor dataset, KPSR and ASLSA have the approximated tracking result. In the Car11 dataset, KPSR, IVT, STRUCK and ASLSA can estimate the more accurate location of the target while the remaining methods fail to estimate the correct location.

**Background clutter case.** Fig. 9 demonstrates how accurate and robust KPSR performs when the target undergoes a heavy background clutter. More specific, in the Board dataset, KPSR is able to obtain a better tracking than ASLSA. This is because the bounding box includes amounts of background information and our contribution factor design helps to reduce the disturbance of the background clutter. Besides, ASLSA also keeps a more continuous tracking through the whole dataset. In the Stone dataset, when a similar big stone fully occludes the target (the small stone), KPSR can keep the minimal tracking error in a continuous tracking as shown in Fig. 5 while FRAG, MIL and VTD suffer from an extreme influence since the 376-th frame.

**VI. CONCLUSION**

In this paper, we propose a robust tracker based on key patch sparse representation. To better solve partial occlusion and background clutter problems, KPSR treats differently the sampled patch according to its location and occlusion case. A contribution factor design for all sampled patches utilizes a weighted approach to important patches. In order to obtain the occlusion case of each patch, we provide an occlusion prediction scheme by training a classifier. In addition, the occlusion degree \( r_{occ} \) is used for template update as a update condition. The experiments on challenging datasets demonstrate that the KPSR tracker not only is accurate and robust for occlusion and background clutter but also is effective for illumination change.

**ACKNOWLEDGMENT**

This study was supported by the Innovation Project of Scholars from Overseas of Shenzhen(KQCX20120801104656658), the Technology Innovation Project of Shenzhen (No. CXZZ20120618155717337, CXZZ20130318162826126) and the National Natural Science Foundation of China (Grant nos. 61203376 and 61375012). This research was also supported in part by Shenzhen IOT key technology and application systems integration engineering laboratory.
Fig. 7. Tracking results on video datasets with heavy or long-time partial occlusion.

Fig. 8. Tracking results on video datasets with illumination change.
REFERENCES


