ABSTRACT

Visual tracking is an important task in many computer vision applications. In this paper, we present a novel approach for visual tracking using a local sparse appearance model that exploits spatial information. The sparse information along with the spatial information of local patches within this representation is used to determine the motion of the object. When the match between the target and candidate patches are represented in matrix form, the translation motion of the object can be obtained by analyzing the diagonals of the mapping matrix. A novel local template update strategy is proposed to update relevant parts of object within the candidate undergoing changes. Along with the local patch update, we use an adaptive/permanent template update strategy which gives less update priority to the transient local patches accounting for partial occlusion. This approach performs comparably against state-of-the-art tracking techniques, in various challenging videos involving changes in scale, pose, illumination and partial occlusion. The proposed approach tracks at a processing speed of over 40 frames per second and it is suitable for various real-time applications.

1. INTRODUCTION

One of the leading research areas of computer vision is Visual Tracking with wide applications in surveillance, navigation, transportation monitoring, human computer interfacing, medical imaging and robotics. Tracking still poses challenges in conditions like partial occlusion, sudden illumination variation, pose change, view point change and scale change. Most of the current trackers sacrifice either accuracy or processing speed. Overcoming both of these problems simultaneously remains one of the biggest research challenges in computer vision.

Many attempts have been made in making the perfect visual tracker. A survey had been done to compare and analyze major algorithms [17]. Initially an algorithm was devised for minimizing SSD (sum of square differences) between possible matches [5] and, the gradient descent algorithm was commonly used to minimize the SSD. Mean-Shift Tracker [4] was developed using mean-shift iterations and a similarity measure based on the Bhattacharyya coefficient between the target and the candidate histograms. Incremental Tracker [14] and Covariance Tracker [13] soon presented other methods of tracking using appearance models. Mei et al presented $l_1$ tracker [11] which utilizes the particle filter to select candidate particles and sparsely represent in space spanned by the object templates using $l_1$ minimization. Several discriminative algorithms were introduced to distinguish the target from the background. Babenko et al. [2] use multiple instance learning to put all positive and negative samples into bags to learn a discriminative tracking model. Kalal et al [7] developed a P-N learning algorithm and classify the underlying positive and negative samples for tracking. Fragment based tracking algorithm [1] represented the template object by multiple fragments. Each patch votes on the possible positions of the object in current frame by comparing the corresponding image patch histogram. Then they minimize a robust statistic in order to combine the vote maps of the multiple patches.

Recently, sparse representation techniques have been proposed in the field of visual tracking [10, 9, 11, 12, 16]. Sparse representation involves representing a signal by linear combination of only a few basis vectors. Mei et al [11, 12] used $l_1$ minimization over a holistic representation of the object as the appearance model composed of dynamic and trivial templates. Liu et al. [10] employed histograms of sparse coefficients and the mean shift algorithm based on a static local sparse dictionary. It handles partial occlusion well but, fails due to insufficient spatial information present in histograms of the sparse coefficients. In [9] an appearance model was introduced based on dynamic group sparsity including spatial and temporal adjacency. One of the most recent of the sparse representation work in visual tracking by Yang et al [6] used local sparse representations over overlapped image patches with a fixed spatial layout along with an alignment-pooling method. They also update the dictionary with dynamic templates. The tracking is then done using a particle filter framework to generate possible candidates for sparse representation. This generative approach works only for a large number of particles and inspite of its high accuracy in dealing with major problems such as partial occlusion, scale, pose and illumination changes, it proves to be very slow.

Our work bares some resemblance with [6] in the use of local sparse representation along with a dynamic dictionary.
Instead of using a computationally expensive particle filter framework, we developed an intelligent algorithm by exploiting the spatial, temporal and local information provided by the sparse coefficients. The proposed algorithm extracts information by observing the structural block diagonal sparse coefficient matrix. It also implements a novel template update technique and local patch weighting based on spatial and temporal information. The update strategy involves selective local template patch updates and dictionary template updates for efficient learning.

The rest of the paper is organized as follows: Section 2 provides the overview of the proposed tracker. Section 3 to 6 describes the proposed approach in detail. Section 3 includes the dictionary formation technique, section 4 describes the dynamic sparse representation framework used, section 5 gives details for motion estimation and section 6 comprises of the novel adaptive template strategy used. Section 7 discusses the results and Section 8 concludes the paper.

2. OVERVIEW

Through Sparse representation, local spatial information of the object can be easily used for various challenging problems in visual tracking. The proposed tracking algorithm exploits the partial spatial information of image patches in the video. The sparse representation creates a relation between the patches in a frame of the video with the patches of the object’s target templates using Orthogonal Matching Pursuit[3]. The main idea for tracking lies in observing the pattern changes in the block diagonal sparse representation matrix. The proposed algorithm handles partial occlusion well due to the presence of local information which proves to be sufficient. The matrix is observed for pattern changes indicating scale change and the scale is changed appropriately. To improve the robustness of the tracker, the proposed algorithm adapts its template dictionary with the changing appearance of the target object. The updates involve local patch updates, complete template update and patch confidence weighing. The algorithm tackles major illumination, pose and other appearance changes through this update strategy. As will be shown in section 7, without compromising high accuracy in tracking, the algorithm achieves tracking frame rate over 40 frames per second.

3. DICTIONARY CREATION

Visual tracking of an object in a video involves comparison of a certain frames of the video with respect to a reference source of object’s information. This source information can be stored in the form of a dictionary, $D$, where the dictionary elements are features of patches sampled from target image. Similarly, a candidate dictionary, $Y$, can be created for each candidate.

At the initial frame, a bounding box is defined by the user representing the window of the object to be tracked. The windowed image is cropped out of the frame and resized to an image of a predefined size $h_{im} \times w_{im}$ called the target template. Similarly, there are multiple target templates obtained throughout the tracking process denoted by, $T = [T_1, T_2, ... , T_n]$.

Overlapping image patches of size $h_p \times w_p$ are sampled from these target template images. The sampled image patches are mapped to a set of feature vectors. Multiple methods of feature extraction were used. Most naive feature vector is made by converting an image patch to a vector. An image patch of dimensions $[m\times n\times p]$ is mapped to a column vector of dimension $[mnp]$\([1]\) where $m$ = height, $n$ = width, $p$ = no. of channels of the image patch. The feature vectors obtained from each image patch are then normalized using the $l_2$ norm. The obtained dictionary is illustrated in Figure 1.

These column vectors, also known as the feature vectors, are then arranged row-wise to comprise the Target Dictionary i.e., $D = [d_1, d_2, ... , d_{n \times N}] \in \mathbb{R}^{d \times (n \times N)}$ where $d$ is the dimension of the feature vector, $n$ is the number of templates and $N$ is the no. of local patches sampled. Most feature vectors uniquely represent specific parts of the target and are sorted sequentially keeping the spatial information of the local patches intact. A dictionary, comprising of these features represent the whole target distributed into local parts. Multiple templates enable the tracker to intelligently capture important information of the object as it undergoes various illumination changes and pose variation. Similarly, for a candidate, we extract local patches that are co-located in the next frame and turn them into vectors to form the candidate dictionary, $Y = [y_1, y_2, ... , y_M] \in \mathbb{R}^{d \times M}$ where $M$ is the number of patches in the candidate image.

The target Dictionary $D$ is augmented with a set of trivial templates denoted by an identity matrix $I \in \mathbb{R}^{d \times d}$ to give $D = [D \ I_d]$ \([16]\) As will be shown in the next section, this trivial dictionary will prove useful to account for noise and occlusion and avoid incorrect information being put into the Sparse Representation Framework.

4. SPARSE REPRESENTATION

Once feature vectors of the $n$ target images and candidate images are extracted and stored in target dictionary $D$ and candidate dictionary $Y$, we represent each image patch vector in the candidate image, as a sparse linear combination of the image patch vectors in the target templates $D$.

Consider a candidate feature vector $y_k \in Y$. Then, $y_k = D b_k$, where $b_k$ is the coefficient vector for the feature vector $y_k$. In tracking, a portion of the object represented as a candidate feature vector shall be represented as a linear combination of a few patches in the target image. Thus, the required $b_k$ coefficient vector must be sparse in nature. Now
the problem reduces to the most fundamental problem of linear algebra $Db = y$. In sparse representation framework, the given equation is an under-determined system, since $D$ is a fat matrix. We are looking for a sparse solution i.e., a solution with minimal $l_0$ pseudo-norm. Though $l_0$ pseudo-norm does not follow the strict definition of a norm, it is defined as the number of nonzero entries in the vector. The identity of $y$ is determined by the sparsity structure of $b$.

Thus the problem can be defined as: $b_k^* = \arg \min ||b_k||_0$ subject to $Db = y$. This can be implemented using the Orthogonal Matching Pursuit [3] implemented in MATLAB using mexOMP[15]. We consider the orthogonal Matching Pursuit (OMP) algorithm to recover a high-dimensional sparse signal based on a small number of noisy linear measurements. OMP is an iterative greedy algorithm that selects at each step, the column which is most correlated with the current residuals. The idea is to identify significant patches’ sparse coefficients in the dictionary in the case where some of the nonzero sparse coefficients of dictionary patches are possibly small. The OMP algorithm will select the most important components before possibly selecting less important ones.

$$\min_{b_k} ||y_k - Db_k||_2^2, \text{ s.t. } ||b_k||_0 \leq \lambda$$

where $\lambda$ is the maximum limit for the $l_0$ norm of vector $b_k$, $y_k$ denotes the $k^{th}$ vectorized local image patch, $b_k \in \mathbb{R}^{(n \times N) \times 1}$ is the corresponding sparse coefficient of that local patch, and $||b_k||_0 \leq \lambda$ means that there may be a maximum of $\lambda$ number of positive coefficients.

Note that $B = [b_1, b_2, ..., b_M]$ represents the sparse codes of one candidate. The sparse coefficients of each local patch are divided into several segments, according to the template that each element of the vector corresponds to, i.e., $b_k^T = [b(1)_T, b(2)_T, ..., b(n)_T, b_p^T]$, where $b(i)_p \in \mathbb{R}^{N \times 1}$ denotes the $i$-th segment of the coefficient vector $b_k$ and $b' \in \mathbb{R}^{k \times 1}$ denotes coefficients corresponding to the trivial dictionary elements. Thus the matrix $B$ can be considered as a confidence match matrix between $Y$ and $D$ where the $Y$ candidate feature vectors are along the columns and the $D$ target feature vectors are along the rows. The value of $B_{i,j}$ represents the confidence value of match between $y_j$ with $d_i$.

The coefficients corresponding to the trivial dictionary vectors help in reducing the coefficients from other dictionary vectors in case of partial occlusion or noise. The trivial dictionary can more easily represent the candidate vector as a linear combination if other dictionary elements fail to produce good matches.

In our experiment, candidate templates are taken to be larger than the size of the template images. The candidate image acts as a search window and the greater it is, tracking of fast moving object becomes easier at the cost of computation. A margin on all four sides can be set resulting in greater number of patches in the candidate. Number of a vectorized image patches of target templates are $N$ while that of the candidate is considered to be $M$.

Say, patch size is $h_p \times w_p$, image size is $(h_{im} + 2 \times h_m) \times (w_{im} + 2 \times w_m)$

where $h_m, w_m$ are margin (in pixels) set on either sides along the height and width respectively.

Then total number of patches along the width,

$$np_w = (w_{im} + 2 \times w_m)w_p + 1, \quad (2)$$

patches along the height

$$np_h = (h_{im} + 2 \times h_m)h_p + 1. \quad (3)$$

Hence, total number of patches would be

$$NP = np_w \times np_h = [(w_{im} + 2 \times w_m)w_p + 1] \times [(h_{im} + 2 \times h_m)h_p + 1] \quad (4)$$

For the target templates, number of vectorized image patches for the target templates is

$$N = [w_{im}w_p + 1] \times [h_{im}h_p + 1] \quad (5)$$

and for the candidate image is

$$M = [(w_{im} + 2 \times w_m)w_p + 1] \times [(h_{im} + 2 \times h_m)h_p + 1] \quad (6)$$

4.1 Robust Matching

Similar to how candidate vectors were represented by dictionary elements ($D \rightarrow Y$ mapping), the dictionary elements can be inversely represented by the candidate vectors ($Y \rightarrow D$ mapping). The two representations are combined together to remove any false positive representations. This means that we consider the pair of patches from target and candidate templates only if they significantly contribute to the construction of the other patch through sparse representation. This leads to a more robust matching technique which is illustrated in Figure 2 showing the two kinds of mapping and the final result.

Consider the Target and Candidate Dictionaries:

$$D' = [d_1, d_2, \ldots, d_{N \times N}] \in \mathbb{R}^{d \times (n \times N)} \quad (7)$$

$$Y = [y_1, y_2, \ldots, y_M] \in \mathbb{R}^{d \times M} \quad (8)$$

$$y_k = Db_k \implies B = [b_1, b_2, \ldots, b_M] \in \mathbb{R}^{(n \times N) \times M} \quad (9)$$

$$d_q = Yc_q \implies C = [c_1, c_2, \ldots, c_{n \times N}] \in \mathbb{R}^{M \times (n \times N)} \quad (10)$$

$$V = B.C^T \quad (11)$$
4.2 Coefficient Weighing

In tracking problems, there is a high chance of background information being present at the edges of the windowed candidate image. Thus, the coefficients denoting the match confidence obtained from the sparse representation need to be weighed down. A Gaussian radial filter of the size of the number of patches sampled from a target template is produced. This filtered image is reshaped to a column vector. It is appropriately repeated along the rows to account for multiple target templates and along the columns for all the candidate vectors. This matrix is used as a mask for the sparse representation matrix. The resultant coefficients of the patches are weighed based on the location of their matched dictionary element with respect to the centre.

4.3 Coefficient Weighing Using Object Patch Matching Confidence

During a video, the target template patches having multiple high representation coefficients are weighed. The template patches having negligible match are considered as part of the background. A weighing matrix is used to weigh the sparse coefficients based on the confidence of the patch belonging to the object or the background. Weight of a patch at \((i,j)\)th location for \(t\)th frame is

\[
    w_{i,j}^t = \frac{((t-1) \times u_{i,j}^{(t-1)} + w')/t}{w'}
\]  

where \(t\) = frame number, \(w_{i,j}\) = equivalent weight assigned for \((i,j)\)th coefficient from previous weighing, \(w' \in [0,1]\) = Confidence of patch belonging to the object (from current coefficient). This classifies candidates from belonging to the object or to the background. Hence, object need not occupy the whole of initial window in the proposed algorithm.

5. MOTION ESTIMATION

The obtained sparse coefficient matrix \(V\) has all the information required for precise tracking under all major challenges like translation, partial occlusion, scale changes, sudden illumination changes, pose variation and temporary complete occlusion. By observing the representation matrix for a particular frame, the correct motion of the object can be deduced.

In an ideal situation where the candidate image is as big as a template image and also forms a perfect match, we would get the matrix \(V\) as an identity matrix \(I\). In other words, every patch \(y_k\) of candidate dictionary \(Y\) can be perfectly represented by single patch \(d_k\) belonging to template dictionary \(D\).

\[
y_k = d_k, \forall k
\]

For translation, scale change, occlusion or light intensity change of the object between frames, this diagonal structure is affected.

5.1 Translation/Displacement

The Sparse Coefficient Matrix, \(V \in \mathbb{R}^{(n \times N)+(y \times d) \times (M)}\) comprises of \(n\) sub matrices \(V(i) \in \mathbb{R}^{N \times M}\) denoting the representation of the candidate patches in terms of the patches in the \(t\)th template \(T(i)\). Ideally if the candidate is exactly the same as the template \(T(i)\), we would obtain an identity matrix \(V(i)\).

Given that most of the patches belonging to the object would be displaced in the same direction, each patch in candidate belonging to the object say, \(p_i\) would now match to a patch shifted by \((d_x, d_y)\) given by \(p_i = p_i + d_x + (d_y \times n_p_w)\). This results in a shifted diagonal structure in the Sparse Coefficient Matrix.

The most prominent diagonal is chosen based on the effective weight of the diagonal. The shift of this diagonal is used to back calculate the Euclidean shift of the object in the candidate with respect to the template. Algorithm 1 describes the process of estimation the translation of the object.

5.2 Partial Occlusion

During partial occlusion, a part of the object’s information is lost. Only part of the diagonal structure is formed and the rest of the patches corresponding to the occlusion have dispersed coefficients with no coherent structure in the Sparse Coefficient Matrix. Thus, the most prominent diagonal contains most of the information about the object and relates to the spatial information accurately. The weight of the diagonal can be weighed to determine the confidence measure of the object’s translation under occlusion.

5.3 Scale Change

Various tracking situations involve scaling of the object. An object scaling down in ideal conditions would have its object patches converge within the candidate image, implying that a few patches in the candidate would match with most of the patches in the target template. Similarly, for an object scaling up, its object patches would diverge within the candidate and thus, most of the patches in the candidate would be matching with a few patches in the target template. This results in tilting of the diagonal in each block due to object stretching the \(x\) axis, and tilting of the block diagonal structure due to object stretching along the \(y\) axis. For scaled up objects, the resultant slope of the diagonal structure is less than 1 whereas, for scaled down objects, it is greater than 1. Hough transform is implemented to identify the most prominent straight line in the sparse coefficient matrix. The slope calculated is used to determine the amount of object stretching in either axes and then, scale the window size accordingly. Refer figure 3 for illustration.

This strategy is adopted for each block of the Sparse Coefficient Matrix to determine the scaling in \(x\) direction. Similarly, the same algorithm is used for the \(weightMat\) matrix obtained in Algorithm 1.

6. THE TEMPLATE UPDATE STRATEGY

Since the object’s appearance keeps changing throughout the video, the target templates must be updated accurately based on the object’s appearance changes. Updates involve intelligently using the object’s patches and down-weighting the background patches. The strategy proposed updates...
Figure 3: Scale change between the target and candidate results in change of slope of the block diagonal structure.

Algorithm 1: Translation Estimation

**Data:** Sparse Representation Matrix V, np, npw, ph, pw, search

**Result:** Displacement of the object \((\Delta x, \Delta y)\)

**initialization**

```plaintext
for i ← 1 to np do
    for j ← 1 to np + ph do
        eV(i,j) = V(npw × (i - 1) + 1 : npw × i, (npw + pw) × (j - 1) + 1 : (npw + pw) × j)
        v = diagvec(eV, 0)
        ref_weight = norm(v, 2)/(length(v > 0))
        for k ← pw - (npw × search) to pw + (npw × search) do
            v = diagvec(eV,k)
            new_weight = norm(v, 2)/(length(v > 0))
            if new_weight > ref_weight then
                shift = k
                ref_weight = new_weight
                el(v) = length(v > 0)
            end
        end
        dMat(i,j) = shift - ph
        weightMat(i,j) = ref_weight
        elem(i,j) = el
    end
end
dvel = diagvec(elem, 0)
v = diagvec(weightMat, 0)
wt = norm(v, 2)/sum(vel)
dshift = ph
for i ← ph - (np × search) to ph + (np × search) do
    dvel = diagvec(elem, i)
    v = diagvec(weightMat, i)
    new_weight = norm(v, 2)/sum(vel)
    if new_weight > wt then
        dshift = i
        wt = new_weight
    end
end
a = diag(weightMat, dshift)
b = diag(dMat, dshift)
\(\Delta y = dshift - ph\)
\(\Delta x = \text{mode}(b(a > 0))\)
```

Figure 4: Algorithm of the proposed Template Update Strategy

Information based on the confidence of matching, the frequency of matching and classification of patch (background or foreground). The strategy is illustrated as a flowchart in figure 4. conf(A,C) means confidence of match between A and C, and d(AC) means euclidean shift of C with respect to A. The strategy involves three parts:

1. Patch Template Update
2. Permanent Dictionary Update
3. Patch + Template Weighing

6.1 Patch Template Update

The patches belonging to the object must be updated and not those belonging to the background or occlusion. Consider \(n + 1\) number of templates. Templates \([A_1, A_2, ..., A_n]\) being the permanent templates and Template B which is updated every frame also known as Most Frequently Updated Template (MFU). After every frame, shift of the candidate C with respect to all templates and its confidence value is calculated. The confidence is related to the strength of the diagonal. Refer Algorithm 2 for patch update and refer Algorithm 3 for B=update(X,C) i.e., updating B by using representation between X and C. Figure 5 shows how a patch is selected for updation. The mean weight of the patch and its nearby patches must cross a threshold.

If mean coefficient value of the patch and the adjacent patches is greater than a minimum threshold, that patch is copied from C to the appropriate place in B determined by the shift observed between X and C.

6.2 Permanent Template Strategy

If the MFU template does not change for more than a certain minimum number of frames and the confidence(B,C) is very high, B template is updated to the permanent Templates \([A_1, A_2, ..., A_n]\). The Template in A with most update probability is replaced with B. In case of low confidence
Algorithm 2: Patch Template Update

Data: Sparse Representation Matrix $V$, $A_1, A_2, \ldots, A_n, B, C, n$, $\text{n}_{p_w}, p_w, \text{minDist}$

Result: $B$

for $i \leftarrow 1$ to $n$
  if $\text{confidence}(A_i, C) \geq \text{minConf}$ then
    $B = \text{update}(A_i, C)$
    $\text{updated} = \text{TRUE}$
  end
  if $\text{confidence}(B, C) \geq \text{minConf}$ AND $\text{distance}(\text{shift}(A_i, C) - \text{shift}(B, C)) \leq \text{minDist}$ then
    $B = \text{update}(B, C)$
    $\text{updated} = \text{TRUE}$
  end
  if $\text{updated} \neq \text{TRUE}$ AND $\text{confidence}(B, C) \geq \text{minConf}$ AND $\text{distance}(\text{shift}(A_i, C) - \text{shift}(B, C)) \leq \text{minDist}$ then
    $B = \text{update}(B, C)$
  end
end

Algorithm 3: $B = \text{update}(X, C)$

Data: Sparse Representation Matrix between $(X, C)$ $V$, $D$, $Y$, $B, C, \text{n}_{p_w}, p_w, \text{p}_{w}, \text{threshold}$

Result: $D$

$D' = \text{createDictionary}(X, D)$
$Y' = \text{createDictionary}(C, Y)$

for $i \leftarrow 1$ to $\text{n}_{p_w}$ do
  $eV(:, i) = V(\text{n}_{p_w} \times (i - 1) + 1 : \text{n}_{p_w} + 2 \times p_w) \times (i + \Delta y + p_h - 1) + 1 : (\text{n}_{p_w} + 2 \times p_w) \times (i + \Delta y + p_h)$
end

for $i \leftarrow 1$ to $\text{n}_{p_w}$ do
  $\text{count} = 1$
  $\text{sum} = eV(j, j + p_w + xshift, i)$
  if $j > 1$ then
    $\text{sum} = \text{sum} + eV(j - 1, j - 1 + p_w + \Delta x, i)$
    $\text{count} = \text{count} + 1$
  end
  if $j < \text{n}_{p_w}$ then
    $\text{sum} = \text{sum} + eV(j + 1, j + 1 + p_w + \Delta x, i)$
    $\text{count} = \text{count} + 1$
  end
  if $i > 1$ then
    $\text{sum} = \text{sum} + eV(j, j + p_w + \Delta x, i + 1)$
    $\text{count} = \text{count} + 1$
  end
  if $i < \text{n}_{p_w}$ then
    $\text{sum} = \text{sum} + eV(j, j + p_w + \Delta x, i + 1)$
    $\text{count} = \text{count} + 1$
  end
  if $\text{sum} > \text{threshold}$ then
    $D(1 : \text{end}, j + (i - 1) \times \text{n}_{p_w}) = Y(1 : \text{end}, j + p_w + \Delta x + (i - 1 + p_h) + \Delta y \times (\text{n}_{p_w} + 2 \times \text{pad}(1))$;
  end
end

Figure 5: The mean weight of the coefficients of the patch to be replaced and its neighbouring patches is used to determine whether to update the patch.

7. RESULTS AND DISCUSSION

The proposed algorithm was implemented on MATLAB running at an average speed of over 40 frames per second on 64 bit Ubuntu-12.04 operating system with 24GB memory, 3.46GHz Xeon processor. All algorithms under comparison were run on the same system. For all tracking video sequences, window was resized to [15 x 15] with an additional margin 2 pixels on either sides. The Orthogonal Matching pursuit was implemented using the SPAMS package [15] with maximum number of elements in each decomposition ($L$) parameter set to 2 and run under a single thread mode. Limit for maximum shift estimated was set to 40% on either sides of the window between two consecutive frames. Motion estimation algorithm was implemented using mex with Armadillo Linear Algebra Library[8]. Vectorized RGB pixel values of single pixel shifted overlapping image patches were used as the feature vectors of the dictionaries. Template update strategy was implemented using two templates. One permanent template and one frequently updated template (MFU).

The proposed algorithm was compared against popular trackers including [6], L1 tracker [12] and Fragtrack [1]. The results were evaluated using the source codes of the above algorithms provided by the respective authors. The codes were run for four video sequences: ming, car44, david indoor and trellis.

7.1 Qualitative Analysis

The results of the specified trackers are are compared for each video sequence. In figure 6(a), 'ming' sequence, the person’s face (target) undergoes illumination changes, pose variation and scale change. All trackers besides L1 tracker manage to track the object in this video sequence almost perfectly. Due to the innovative template update strategy, the templates are updated with the different poses and light illumination.

In figure 6(b), 'car44' sequence, the car’s rear is marked as the object. As the car moves, its scale changes due to its motion away from the camera. This scale change is tackled with all the templates, permanent set of templates is not updated. Greater the life of particular template, lower are its chances of getting updated. In situations like temporary occlusion, the original tempaltes have more importance than the ones updated using the occluded candidate.
7.2 Quantitative Analysis

Figure 6: Tracking results for (a) ming, (b) car44, (c) david indoor, and (d) trellis video sequences.

Figure 7: The plots represent the position error of various tracking algorithms. Red: Proposed Algorithm, Green: Reference, Yellow: L1 tracker, Blue: Frag Track.

by the proposed and the reference algorithm, whereas the L1 and Frag Track algorithms fail to adapt to the scale changes. As the car moves under the bridge (c), it undergoes sudden and heavy illumination change. This shows the efficiency of the proposed algorithm as the object suffers sudden appearance changes. The proposed and reference algorithm manage to track the object accurately.

In figure 6(c), ‘david indoor’ sequence, the person’s face marked as the object undergoes illumination change and extreme pose and appearance variations. This video also involves viewpoint changes which the proposed and reference algorithm have no trouble in overcoming. During the most critical part of the video with extreme pose variation involving the face to rotate 90 degrees away from the camera, the reference algorithm fails, whereas the proposed algorithm continues to efficiently track the object till the end of the video.

Figure 6(d), ‘trellis’ sequence is one of the most challenging sequences, where the object marked as the person’s face, experiences major irregular illumination changes, pose variations and sudden jerks in motion. Due to the sufficient search region for the proposed algorithm, it is able to follow the object through rough motion where all other algorithms fail yet again. Moreover, the object keeps the correct trajectory throughout the illumination changes and sudden pose variations.
Quantitative Analysis is carried using two criteria namely, position error and tracker execution speed. Position error is calculated as the root mean square error (RMSE) for each tracker’s estimated central location of object with the ground truth. The results are plotted together and shown in figure 7.

Table 1 shows the position error of the results obtained from the four trackers. Table 2 shows the execution speed given in frame per second. Clearly, the proposed algorithm works better in three of four cases. Moreover, the speed obtained is greater than twice of real-time speed. The accuracy of the proposed algorithm is similar to the reference algorithm whereas, in terms of speed the reference algorithm lags behind due to its particle filter framework which is very slow.

<table>
<thead>
<tr>
<th>Video</th>
<th>Proposed</th>
<th>Reference</th>
<th>L1</th>
<th>Fragtrack</th>
</tr>
</thead>
<tbody>
<tr>
<td>ming</td>
<td>3.85</td>
<td>2.50</td>
<td>37.89</td>
<td>8.24</td>
</tr>
<tr>
<td>car44</td>
<td>1.91</td>
<td>2.52</td>
<td>161.75</td>
<td>140.93</td>
</tr>
<tr>
<td>david</td>
<td>6.89</td>
<td>32.67</td>
<td>71.45</td>
<td>95.29</td>
</tr>
<tr>
<td>trellis</td>
<td>5.70</td>
<td>35.60</td>
<td>90.32</td>
<td>33.15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Video</th>
<th>Proposed</th>
<th>Reference</th>
<th>L1</th>
<th>Fragtrack</th>
</tr>
</thead>
<tbody>
<tr>
<td>ming</td>
<td>66.98</td>
<td>3.38</td>
<td>0.277</td>
<td>12.55</td>
</tr>
<tr>
<td>car44</td>
<td>41.25</td>
<td>2.926</td>
<td>0.299</td>
<td>12.37</td>
</tr>
<tr>
<td>david</td>
<td>63.26</td>
<td>3.08</td>
<td>0.289</td>
<td>12.64</td>
</tr>
<tr>
<td>trellis</td>
<td>67.24</td>
<td>2.82</td>
<td>0.301</td>
<td>12.70</td>
</tr>
</tbody>
</table>

8. CONCLUSION

As evaluated, overall the proposed algorithm stands out in various difficult scenarios. Not far behind, the reference algorithm follows with minor decrease in accuracy. The L1 and fragtrack algorithm seem to fail easily when posed with difficult problems The proposed algorithm with its adaptive techniques of template updation, it tackles problems of pose variation, illumination change and viewpoint changes. Exploiting the spatial and local information provided through the sparse representation framework, it is insensitive to occlusion. Using temporal information along with the update strategy, it overcomes temporary complete occlusion. While adopting all these intelligent techniques through a novel sparse representation solution, it presents itself as one of the most robust tracking algorithms developed till date. Moreover, with a high accuracy in tracking various challenging videos, it is extremely fast and boasts average runtime speed of over 40 frames per second.

9. ACKNOWLEDGEMENTS

This work was carried out during the first author’s internship, at Video Analytics Lab, at SERC, Indian Institute of Science, Bangalore, through the Summer Internship Program (SIP) of SERC.

10. REFERENCES