Learning Occlusion with Likelihoods for Visual Tracking

Suha Kwak Woonhyun Nam Bohyung Han Joon Hee Han
Department of Computer Science and Engineering, POSTECH, Korea
{mercury3,xgene,bhhan,joonhan}@postech.ac.kr

Abstract

We propose a novel algorithm to detect occlusion for visual tracking through learning with observation likelihoods. In our technique, target is divided into regular grid cells and the state of occlusion is determined for each cell using a classifier. Each cell in the target is associated with many small patches, and the patch likelihoods observed during tracking construct a feature vector, which is used for classification. Since the occlusion is learned with patch likelihoods instead of patches themselves, the classifier is universally applicable to any videos or objects for occlusion reasoning. Our occlusion detection algorithm has decent performance in accuracy, which is sufficient to improve tracking performance significantly. The proposed algorithm can be combined with many generic tracking methods, and we adopt L_1 minimization tracker to test the performance of our framework. The advantage of our algorithm is supported by quantitative and qualitative evaluation, and successful tracking and occlusion reasoning results are illustrated in many challenging video sequences.

1. Introduction

Visual tracking is one of the most popular problems in computer vision since it is a crucial task for many real-world applications. Although visual tracking has been investigated intensively, there are still many challenges; it frequently suffers from occlusions, appearance changes, significant motions, background clutter, etc. Many computer vision researchers make efforts to overcome the challenges; various adaptive appearance modeling algorithms are proposed in [12, 15, 21], a couple of algorithms to track an object with significant motions are introduced in [16, 28], and discriminative features for tracking are identified and used to track the target in the presence of background clutter [3, 7, 20].

Among many challenges in tracking problem, occlusion is one of the most critical issues since it is not straightforward to generalize and model occlusions. Due to the importance of occlusion reasoning in visual tracking, there has been a large volume of research related to this problem in various aspects, but there is no general framework to identify occlusions explicitly. Some adaptive appearance modeling techniques attempt to solve the occlusion problem indirectly by statistical analysis [12, 15, 21], but the appearance models are susceptible to be contaminated by long-term occlusions due to their blind update strategies. The target is divided into several components or patches so that the occlusion is implicitly reasoned by robust statistics [1, 6, 13], by patch matching [27], or by spatially biased weights on target observations [5]. Using multiple cameras is a good option to handle occlusion problem [8, 9, 19], but it is not applicable to many videos in hand because it requires a special setup and additional cost for multi-camera system. On the other hand, several algorithms are proposed to overcome occlusions in limited conditions; [25] infers the occlusions among multiple targets in the context of multi-object tracking, [10] discusses self-occlusion for image registration in a controlled environment, and [17, 23] reason the occlusions related to well-known objects such as hands or upright human bodies based on predefined model constraints. Recently, a few attempts to manage occlusions and other exceptions in tracking are made based on the spatio-temporal context [11, 26], but they require non-trivial observation and tracking of objects or features outside target.

Most of the existing occlusion reasoning and handling techniques have critical limitations such as the need of multiple cameras, strong models, and environment understanding. More importantly, it is generally difficult to determine the occlusion status given an observation in tracking scenarios. Motivated by this, we propose an active occlusion detection and handling algorithm for tracking by learning with observation likelihoods. To detect occlusion, we learn the patterns of likelihoods based on the data collected during tracking with and without occlusions. Even though we train the classifier with several specific videos, the trained classifier for occlusion detection is universal for general videos and/or objects because the features for the classifier are observation likelihoods, not image features. However, training and testing should be performed in the same environment for the reliability of our algorithm, and the same tracking al-

gorithm needs to be employed for the both procedures. Note that the data collection for training is crucial for the performance of the classifier in a new sequence since the patterns of the data observed in testing should be similar to the patterns in training. In our algorithm, the target is divided into a regular grid, and we determine the state of occlusion for each cell using the trained classifier. For tracking, the likelihood of each observation is computed based on unoccluded cells given the occlusion mask, which is constructed by applying a classifier to the target window at each frame. Our method has several important advantages as follows:

- We can compute more reasonable observation likelihoods when occlusion is involved because we effectively disregard the occluded cells for observation.
- Our classifier is trained using patch likelihoods associated with the cells in the target, and can be used for any other videos and/or objects to detect occlusions; we do not train a specialized classifier for a sequence or a target, but construct a *single universal* classifier.
- Our classifier is not perfect, but our simulation and experiment support that the decent performance of the classifier improves tracking accuracy significantly.
- Our occlusion reasoning technique can be integrated in many generic tracking algorithms—especially, template-based or part-based algorithms as long as the observation model is identical.[1]

The rest of the paper is organized as follows. Our occlusion reasoning algorithm in the L_1 minimization tracking is presented in Section 2, and the classifier for occlusion detection is described in Section 3. We illustrate experimental results with performance evaluation in Section 4.

2. Tracking with occlusion reasoning

Our occlusion reasoning technique should be incorporated into a tracking algorithm; particle filter tracking with the L_1 minimization [18] is employed to estimate and propagate the state of the target sequentially. In this section, we first review the L_1 minimization tracking framework, and present how our occlusion reasoning algorithm is integrated into the tracking method.

2.1. L_1 Minimization tracking

Suppose that $\mathbf{T} = [\mathbf{t}_1\ \mathbf{t}_2 \dots \mathbf{t}_n] \in \mathbb{R}^{d\times n}$ and $\mathbf{I} = [\mathbf{i}_1\ \mathbf{i}_2 \dots \mathbf{i}_d] \in \mathbb{R}^{d\times d}$ are a set of target templates and trivial templates[2], respectively. In the original L_1 minimization tracking, the target template, $\mathbf{T}$, as well as the positive and negative trivial templates, $\mathbf{I}$ and $-\mathbf{I}$, are stored to compute likelihoods and update appearance model. The observation denoted by $\mathbf{y}$ is decomposed as

$$\mathbf{y} = [\mathbf{T}\ \ \mathbf{I}\ -\mathbf{I}] \begin{bmatrix} \mathbf{a} \\ \mathbf{e}_+ \\ \mathbf{e}_- \end{bmatrix} = \mathbf{B}\mathbf{c} \quad \text{subject to} \quad \mathbf{c} \geq 0, \quad (1)$$

where $\mathbf{c}^\top = [\mathbf{a}^\top\ \mathbf{e}_+^\top\ \mathbf{e}_-^\top]$ is a non-negative coefficient vector. Note that $\mathbf{a} \in \mathbb{R}^n$ is a coefficient vector for the target template, and $\mathbf{e}_+ \in \mathbb{R}^d$ and $\mathbf{e}_- \in \mathbb{R}^d$ are coefficient vectors for the positive and negative trivial templates, respectively. The good observation of the target can be approximated with a sparse representation of the stored templates, $\mathbf{B} = [\mathbf{T}\ \ \mathbf{I} - \mathbf{I}]$. The solution is obtained by the regularized L_1 minimization with non-negative constraint [24] as

$$\mathbf{c}^* = \arg\min_{\mathbf{c}} ||\mathbf{B}\mathbf{c} - \mathbf{y}||_2^2 + \tau||\mathbf{c}||_1, \quad (2)$$

where $||\cdot||_1$ and $||\cdot||_2$ denote the L_1 and L_2 norms, respectively, and τ is a regularization constant.

The L_1 minimization technique is integrated into a particle filter tracking framework [14]. In particle filtering, the posterior density of the target state is represented with a set of weighted samples, where the weight of each sample is computed by an observation model. The optimization process in Eq. (2) is performed for each sample to obtain observation likelihood. Given a set of particles generated by importance sampling in each frame, the likelihood of the observation $\mathbf{y}_i$ corresponding to the i-th sample is obtained by computing the similarity between $\mathbf{y}_i$ and the reconstructed image of $\mathbf{y}_i$ with non-trivial templates, which is given by

$$\ell(\mathbf{y}_i) = \exp\left(-\frac{\lambda||\mathbf{y}_i - \mathbf{T}\mathbf{a}_i||^2}{d}\right), \quad (3)$$

where $\ell(\cdot)$ denotes the likelihood for a given observation, $\mathbf{a}_i$ is computed based on Eq. (2), and λ is a constant. The basic idea in Eq. (3) is that a good observation can be reconstructed effectively with a set of non-trivial target templates only. If the appearance of the best sample denoted by $\mathbf{y}^*$ is not sufficiently similar to the existing target templates, the least important template in $\mathbf{T}$ is replaced by $\mathbf{y}^*$. For the details about the L_1 minimization tracking, refer to [18].

2.2. Occlusion detection and handling

Although the original L_1 minimization tracker can handle the occlusions and the corruptions of the observation conceptually, the performance of occlusion handling is not so satisfactory in practice, which is partly because the likelihood computation and the template update do not consider the occlusion status. We now tackle the issues and propose an active occlusion handling algorithm in the existing L_1 minimization tracking framework.

[1] We will mainly describe our occlusion detection framework in the L_1 minimization tracking, but also illustrate the applicability to other tracking methods such as incremental subspace learning [21] in our experiments.

[2] The trivial templates are special templates with only one element in each vector (template) one to reduce the error in reconstruction. Therefore, $\mathbf{I}$ is an identity matrix. For the details, see [18, 24].

When the state of the target is estimated in each frame, the occlusion mask is constructed by applying the trained classifier to the target window. Suppose at the moment that we already have a trained classifier C for occlusion detection, $C : \mathbb{R}^p \rightarrow \{0, 1\}$, where p is the dimensionality of the feature vectors, 0 and 1 denote *non-occlusion* and *occlusion*, respectively. The target region is divided into a 4×4 regular grid and the occlusion is detected for each cell using the classifier C. Note that a single occlusion mask per frame is obtained from the estimated target window.

The occlusion mask constructed in frame $t-1$ is utilized to detect the occlusion status of the target and compute the likelihood of each observation at frame t. Only unoccluded cells are considered for reconstruction and likelihood computation. Let $\hat{\mathbf{y}}_i \in \mathbb{R}^k$ ($k \leq d$) be a vector of unoccluded elements in $\mathbf{y}_i$. Note that the same occlusion mask is applied to the all samples in a time step and the dimensionality of $\hat{\mathbf{y}}_i$ ($i = 1, \ldots, m$), where m is the number of the particles, is fixed in each frame. The likelihood of $\hat{\mathbf{y}}_i$ is given by

$$\ell(\hat{\mathbf{y}}_i) = \exp\left(-\frac{\lambda||\hat{\mathbf{y}}_i - \hat{\mathbf{T}}\hat{\mathbf{a}}_i||^2}{k}\right), \tag{4}$$

where $\hat{\mathbf{T}} \in \mathbb{R}^{k \times n}$ is the target templates with only unoccluded pixels and $\hat{\mathbf{a}}_i$ is the coefficient vector for $\hat{\mathbf{T}}$ obtained from the solution of Eq. (2) based only on unoccluded pixels. If there is no occlusion, $\hat{\mathbf{y}}_i = \mathbf{y}_i$ and $\hat{\mathbf{T}} = \mathbf{T}$. By using Eq. (4) instead of (3), the effect of the occluded pixels in the target is removed, and the likelihood for each sample is more reliable. The occlusion mask at frame t is constructed by applying the trained classifier to the likelihood vectors generated from the estimated target window; we generate a feature vector by computing the likelihoods of the patches in each cell (Figure 1) and perform the classification for each cell independently. To compute the patch likelihoods, we compare the best observation ($\mathbf{y}^* = \mathbf{y}_{i^*}$) and its reconstructed image ($\mathbf{T}\hat{\mathbf{a}}_{i^*}$), where $i^* = \arg\max_i \ell(\hat{\mathbf{y}}_i)$. Note that the occlusion reasoning is done in the original d dimensional space using the entire pixels in the target.

When the occlusion is severe—i.e., the occlusion ratio is more than a threshold γ_1, we do not trust the likelihoods of all the samples and tracking is performed by the motion model, which is obtained from the history of the motions of the target for a fixed length time window δ. In practice, it is particularly useful when the posterior estimation based on the observation is not reasonable due to heavy occlusions.

2.3. Template update and feature selection

The template update strategy is same with the original L_1 minimization tracking except that the target template is replaced only if the occlusion ratio of the new template is less than a predefined threshold γ_2.

Some of 4×4 cells within the target may not have sufficient amount of features for motion estimation; it is called *degeneracy*. Tracking performance can be deteriorated by the cells lacking good features to track, and this situation is aggravated when most of unoccluded cells are degenerate. We determine the trackability of a cell by identifying the number of the degenerate pixels in the cell, and ignore non-informative cells for tracking. This idea is almost identical to [22] and improves the localization performance.

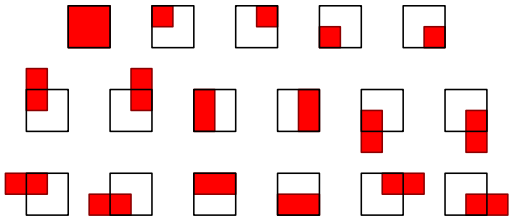


Figure 1. The layout of 17 patches (shaded areas) associated with a cell (rectangle). The likelihoods of the patches constitute feature vectors for training and testing.

3. Learning occlusion with likelihoods

We now describe how to design the classifier for occlusion detection and discuss a couple of important characteristics related to the classifier. Our main objective is to find a general and robust algorithm to detect occlusions during tracking, and we train a binary classifier to determine the occlusion status. We describe the features and the training methodology used for our classifier.

3.1. Features for the classifier

Suppose that a p-dimensional feature vector $\mathbf{f}_i = (f_i^1 \ldots f_i^p)^\top$ ($i = 1, \ldots, q$), where q is the number of cells, is given to the classifier for occlusion detection. Note that, if the feature vector $\mathbf{f}_i$ is extracted directly from the target image, we may need to train the classifier for each target and each sequence separately, which is not desirable for the generality of the algorithm. So, we train the classifier over the likelihood vectors, $\mathbf{l}_i = (l_i^1 \ldots l_i^p)^\top$, where each element in the vector represents the observation likelihood for the p-th small patch associated with the i-th cell.

We obtain a 17-dimensional feature vector from each cell in the target based on the 17 associated patches as illustrated in Figure 1. Although the occlusion detection is performed to each cell independently, we consider neighborhood information indirectly by using features overlapped with adjacent cells. By adopting the transformed feature vector (likelihood vector) instead of the feature vector extracted from images directly, the trained classifier can be utilized for general purposes. Note that, in our algorithm, only a *single* classifier is trained for all sequences, objects and cells within the target.

3.2. Training strategy

Typically, the likelihoods between target model and candidates decrease over time due to gradual appearance changes of the target, and the absolute value of the likelihood may be extremely small if significant time has passed since the target appearance model is constructed. Since the length of the input video sequence is unknown in advance, we need to find a reasonable way to limit the lower bound of the likelihood; otherwise, it is more difficult to obtain a good training dataset since training data should cover many different orders of magnitude of likelihoods for reliable classification performance.

The method we chose to figure out this challenge is the integration of a tracking algorithm that updates the target appearance model adaptively; the likelihoods between the stored target appearance model and new observations do not decrease indefinitely in practice since the target model is updated occasionally using the new observations.

On the other hand, if training is performed based on the likelihood vectors with no consideration of testing environment, the performance of the classifier for occlusion reasoning may not be able to identify the characteristics of the occlusion in testing phase appropriately. Collecting the training data whose patterns are similar to the testing data by modeling occlusions realistically with spatio-temporal tracking context analysis would improve classification performance. We simulate tracking procedure in collecting the training data for the classifier.

Because of the two factors, we adopt the L_1 minimization tracking algorithm with template update strategy [18] as our tracking algorithm for both training and testing. The details about training procedure is described next.

3.3. Training the classifier

For training, the groundtruths for the motion and the occlusion mask of the target in all frames are given by manual annotation. Let $\mathbf{g}_t$ be the groundtruth observation of the target at frame t. Also, denote by $\mathbf{B}_t = [\mathbf{T}_t \ \mathbf{I}_t \ -\mathbf{I}_t]$ the set of the stored target and trivial templates at frame t. Then, the groundtruth observation $\mathbf{g}_t$ is represented as

$$\mathbf{g}_t = \mathbf{B}_t \mathbf{c}_t \quad \text{subject to} \quad \mathbf{c}_t \geq 0, \tag{5}$$

where $\mathbf{c}_t^\top = [\mathbf{a}_t^\top \ \mathbf{e}_{+,t}^\top \ \mathbf{e}_{-,t}^\top]$, and the reconstructed target image $\mathbf{r}_t$ corresponding to $\mathbf{g}_t$ is given by

$$\mathbf{r}_t = \mathbf{T}_t \mathbf{a}_t. \tag{6}$$

As mentioned earlier, the target region is divided into 4×4 grid cells and the occlusion status is determined for each cell independently. Each cell is associated with many small patches, which are overlapped with the cell fully or partially as illustrated in Figure 1. Let $\mathbf{g}_t^{(i,j)}$ be the groundtruth appearance of the j-th patch associated with the i-th cell. A likelihood vector for training is the collection of p likelihoods from the p patches, where each likelihood in the i-th cell is given by

$$l_i^j = l(\mathbf{g}_t^{(i,j)}, \mathbf{r}_t^{(i,j)}) = \exp\left(-\frac{||\mathbf{g}_t^{(i,j)} - \mathbf{r}_t^{(i,j)}||^2}{\sigma \cdot \dim(\mathbf{g}_t^{(i,j)})}\right), \tag{7}$$

where σ is a constant, $\dim(\cdot)$ denotes the dimensionality of a vector, and $j = 1, \ldots, p$. If the i-th cell is occluded, its label $C(\mathbf{l}_i) = 1$; otherwise $C(\mathbf{l}_i) = 0$. Note that a single likelihood vector for training is obtained from each cell. The pixels outside target region are not used to compute the likelihood in Eq. (7), and the boundary cells may have redundant dimensions in their feature vectors.

Once the positive and negative examples are collected, we train a linear Support Vector Machine (SVM), which is used to determine the state of occlusion of each cell in an on-line manner during tracking. The pseudocode of overall tracking procedure with our occlusion reasoning is described in Algorithm 1.

Algorithm 1 L_1 tracking with occlusion reasoning

Init. $\mathbf{T}$, $\mathbf{h}_s$ (degeneracy mask), $\mathbf{h}_o$ (occlusion mask)
$\mathbf{h} \leftarrow \mathbf{h}_s \vee \mathbf{h}_o$, $\gamma \leftarrow \sum_{i=1}^{q} \mathbf{h}(i)/q$.
while the target is in the scene **do**
  if $\gamma < \gamma_1$ **then**
    Apply $\mathbf{h}$ to $\{\mathbf{y}_1, \ldots, \mathbf{y}_m\} \rightarrow \{\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_m\}$.
    Apply $\mathbf{h}$ to $\mathbf{T} \rightarrow \hat{\mathbf{T}} = [\hat{\mathbf{t}}_1, \ldots, \hat{\mathbf{t}}_n]$.
    $[\mathbf{y}^*, \hat{\mathbf{c}}] \leftarrow L_1$_Track$(\{\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_m\}, \hat{\mathbf{T}})$.
    Compute $\{\mathbf{l}_1, \ldots, \mathbf{l}_q\}$ based on $\mathbf{y}^*$ and $\mathbf{T}\hat{\mathbf{a}}_{i^*}$.
    $\mathbf{h}_o(i) \leftarrow C(\mathbf{l}_i), \ \forall i = 1, \ldots, q$.
    The occlusion ratio $\gamma_o \leftarrow \sum_{i=1}^{q} \mathbf{h}_o(i)/q$.
    if $||\mathbf{y}^* - \mathbf{t}_{\text{argmax}_i \hat{\mathbf{a}}_i}|| > \xi$ and $\gamma_o < \gamma_2$ **then**
      $\mathbf{T} \leftarrow$ Template_Update$(\mathbf{y}^*, \mathbf{T})$.
      Compute $\mathbf{h}_s$ from $\mathbf{y}^*$.
    end if
  else
    Predict the target state $\mathbf{x}_o$ with motion prior.
    $[\mathbf{y}_o, \mathbf{c}_o] \leftarrow L_1$_Track$(\{\mathbf{y}_o\}, \mathbf{T})$.
    Compute $\{\mathbf{l}_1, \ldots, \mathbf{l}_q\}$ based on $\mathbf{y}_o$ and $\mathbf{T}\mathbf{a}_o$.
    $\mathbf{h}_o(i) \leftarrow C(\mathbf{l}_i), \ \forall i = 1, \ldots, q$.
  end if
  $\mathbf{h} \leftarrow \mathbf{h}_s \vee \mathbf{h}_o$, $\gamma \leftarrow \sum_{i=1}^{q} \mathbf{h}(i)/q$.
end while

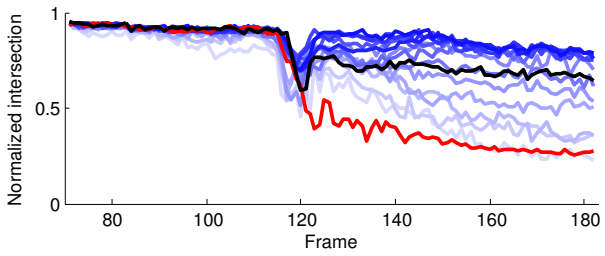
4. Experiment

Our occlusion reasoning algorithm for tracking is evaluated in various scenarios. We first present the potential advantages of our algorithm, and illustrate the performance of our tracking algorithm in several challenging videos.

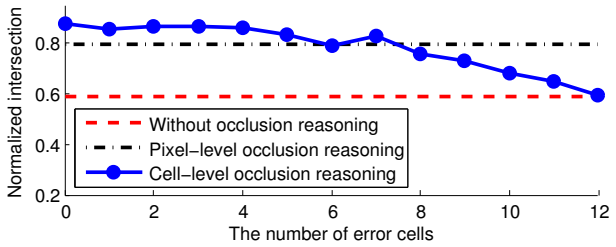
4.1. Simulation

We present the behavior of the L_1 minimization tracking algorithm with occlusion handling by simulation. Because our occlusion reasoning is not perfect, we tested the performance of the tracking algorithm in the presence of occlusion detection errors, which is illustrated in Figure 2. For the experiment, groundtruth bounding box, pixel-level occlusion mask, and cell-level occlusion mask are constructed manually in each frame; we assume that a cell is occluded if more than 30% of the pixels in the cell are occluded. We perform L_1 minimization tracking and compare the normalized intersection ratio—intersection over union of two bounding boxes—between groundtruth and tracking results in each frame, where the results are generated for 12 different levels in occlusion detection accuracy. The errors in occlusion detection are simulated by flipping the groundtruth binary labels of randomly selected cells in the target.

As seen in Figure 2(a), the tracking accuracy with occlusion reasoning does not degrade much with minor or moderate occlusion reasoning errors—it is sometimes even better, which is interesting. The performance with and without occlusion reasoning is illustrated in Figure 2(b). The cell-level occlusion reasoning is better than no occlusion reasoning and as good as perfect pixel-level occlusion reasoning in tracking, even with non-trivial error rates in occlusion detection. The results in Figure 2 suggest that decent occlusion reasoning improve tracking performance significantly.



(a) Normalized-intersection ratios for the different level of occlusion reasoning accuracies $(0/16, 1/16, \ldots, 12/16)$.



(b) Comparisons between with and without occlusion reasoning by varying the number of error cells.

Figure 2. The simulation of the L_1 trackers with occlusion reasoning in the *campus* sequence; there happens a severe occlusion around frame 120 in the sequence (see Figure 4). In (a), the red and black lines illustrate the tracking accuracies with no occlusion reasoning and perfect pixel-level occlusion reasoning, respectively. The blue lines denote the accuracies with cell-wise occlusion reasoning with errors; the darker means smaller error ratio.

4.2. Data

We constructed the dataset for training the linear SVM classifier based on 8 car blackbox videos downloaded from *YouTube*. We manually annotated bounding boxes and the occlusion masks; the tracking procedure was simulated to obtain feature vectors for training as described in Section 3.3. We obtain 16 feature vectors from the target in each frame since a target is composed of 4×4 grid cells; we extracted 16,288 feature vectors altogether from the entire dataset to train the classifier.

Five videos are used to evaluate our algorithm, and no video for training is used in testing; three public datasets—*TUD-Crossing* [2], *CAVIAR* [4], and *Dudek* [15]—are included, and the other two videos are downloaded from *YouTube*—*campus* and *accident* sequence. All sequences involve various types of partial and/or full occlusions, and most of them have multiple occlusions.

4.3. Tracking results

The performance of our algorithm is tested in various scenarios, and our results are compared with other tracking algorithms—incremental subspace learning (IVT) [21] and L_1 minimization tracking (L1T) [18]. To illustrate the state of the target in image, we used red, purple, orange, and green bounding boxes, which denote tracking results of our algorithm, L1T, IVT, and groundtruths in the frame, respectively. The occlusion status is also presented at the lower-right corner in image for our algorithm, and the status of each block is specified with different intensities—black for occluded cell, gray for degenerate cell and white for unoccluded and non-degenerate cell.

We first tested our algorithm in the *TUD-Crossing* sequence and the qualitative comparisons are illustrated in Figure 3. Our tracking algorithm tracked the target successfully and maintained target templates accurately even with multiple occlusions by pedestrians while L1T and IVT were distracted by the occlusions significantly.

Tracking a moving car in the *campus* sequence is challenging because the target is occluded partially and almost completely by people and another vehicle. As illustrated in Figure 4, L1T and IVT managed to handle a weak occlusion by pedestrians between frame 20 and 55, but both algorithms failed after a severe occlusion around frame 120, and could not recover from it. However, our algorithm tracked the target by identifying small unoccluded areas in the car accurately; the leftmost part of the target is visible in frame 120 (see the target window in the lower-right corner) and

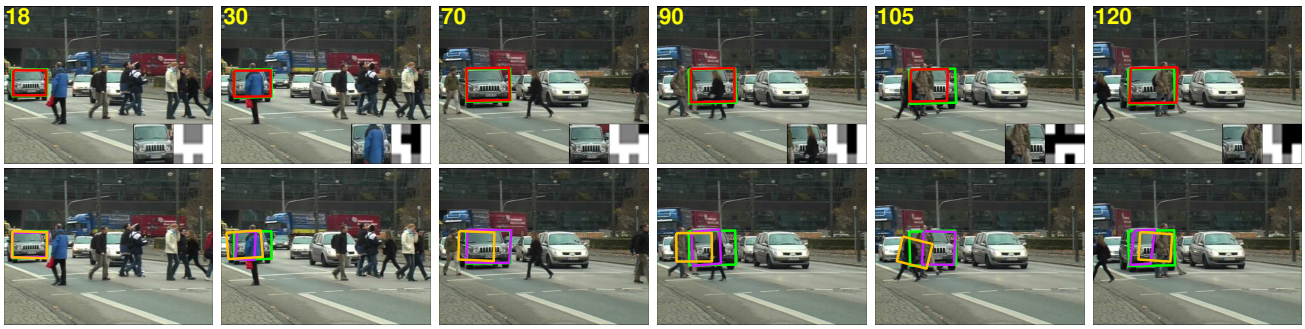


Figure 3. The results of the *TUD-Crossing* sequence. **(top)** ours (red), **(bottom)** L1T (puple) and IVT (orange); groundtruths are denoted by green bounding boxes. Our tracking algorithm is successful, but others fail due to multiple occlusions.

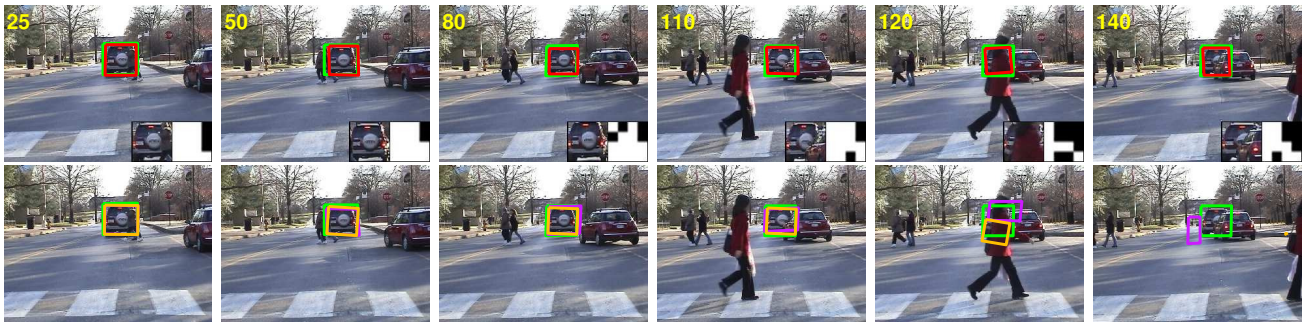


Figure 4. The results of the *campus* sequence. **(top)** ours (red), **(bottom)** L1T (puple) and IVT (orange); groundtruths are denoted by green bounding boxes. The severe occlusion ($t = 120$) and the long-term occlusions ($t = 110, 120, 140$) differentiate the performance of three algorithms.

the occlusion mask coincides with the observation. Also, our tracking algorithm successfully maintained a long term occlusion after the severe occlusion.

The performance of the three algorithms are also compared quantitatively by illustrating the normalized bounding box intersection ratios between groundtruths and tracking results in Figure 5. Other occlusion reasoning methods are also tested and the results are included in the figure—random guessing (ORR) and reasonings based on the likelihood of the entire region in each cell (ORM and ORC), where ORM sets the decision boundary to minimize the training error and ORC makes a more conservative decision to declare occlusions. The performance of our algorithm is obviously better than all others, and tracking accuracies of our algorithm are stable even in the middle of occlusions; note that there are occlusions at $t = 20 \sim 45$ and $t = 78 \sim 123$ in the *TUD-Crossing* sequence and at $t = 14 \sim 54$ and $t = 83 \sim 182$ in the *campus* sequence. Our occlusion detection algorithm is not very accurate as illustrated in Figure 6, but its performance is still better than others. Note that tracking is significantly improved with the decent performance of occlusion reasoning, which is a good property of our algorithm. In our occlusion reasoning, the cells with sudden appearance changes or non-trivial occlusions (but still less than 30%) are sometimes misclassified as occlusions since we trained the classifier based on likelihoods and binary labels. They degrade our occlusion reasoning performance, but frequently help the tracker.

Our classifier for occlusion detection is trained with vehicle data only, but the classifier is applied to pedestrian and face tracking successfully as in Figure 7. This shows our classifier is useful to unknown general objects, too. Occlusions in these sequences are relatively mild, but the occlusion reasoning is reliable throughout the sequences; the accuracy of the occlusion reasoning is 0.80 in the *CAVIAR* sequence and 0.90 in the *Dudek* sequence.

By our explicit occlusion detection algorithm, most of cells in the target may be classified as "occluded". In this case, we rely on prediction—motion history—for tracking instead of observations. We illustrate a comparative result between our algorithm and L1T in the presence of a full occlusion in Figure 8. The degree of occlusion including degeneracy is presented in Figure 9, and the shaded area denotes the frames tracked by prediction.

Finally, we combined our occlusion reasoning algorithm with IVT to demonstrate that our algorithm is applicable to other tracking methods. As illustrated in Figure 10, IVT with our occlusion reasoning algorithm successfully tracked the targets in the *TUD-Crossing* and the *campus* sequence, in which the original IVT failed to track the same targets.

Figure 7. The results by applying our classifier to non-vehicle targets. **(top)** *CAVIAR* sequence, and **(bottom)** *Dudek* sequence.

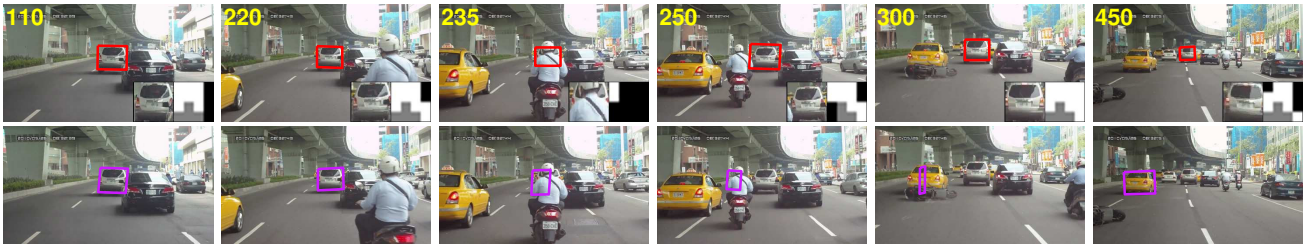
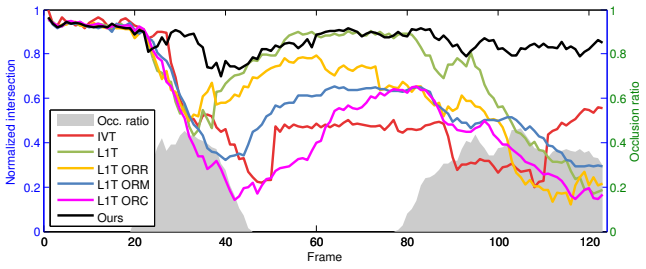
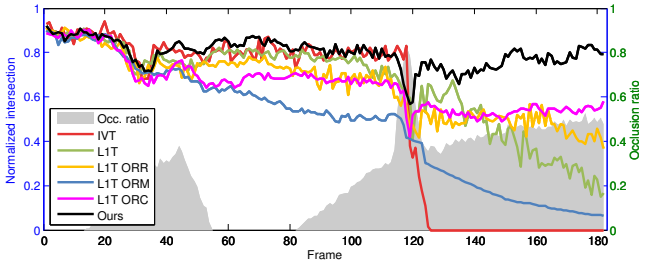


Figure 8. The results of the *accident* sequence. **(top)** ours and **(bottom)** L1T. Ours is successful after the full occlusion around frame 235.



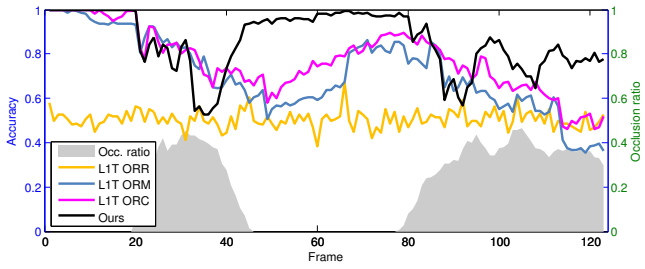
(a) *TUD-Crossing* sequence.



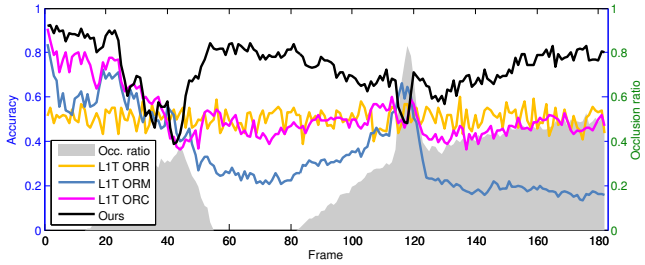
(b) *campus* sequence.

Figure 5. The quantitative comparisons of 6 tracking algorithms—IVT, L1T, ORR, ORM, ORC and ours. The pixel-level groundtruth occlusion ratio over time is represented by shaded area. The average normalized intersection ratios are 0.54, 0.61, 0.63, 0.60, 0.51 and 0.86, respectively, in the *TUD-Crossing* sequence, and 0.54, 0.66, 0.65, 0.48, 0.66 and 0.84, respectively, in the *campus* sequence.

Note that IVT regularly updates the target templates by incremental subspace learning.



(a) *TUD-Crossing* sequence.



(b) *campus* sequence.

Figure 6. The quantitative comparisons of 4 occlusion reasoning algorithms over time—ORR, ORM, ORC and ours. The pixel-level groundtruth occlusion ratio over time is represented by shaded area. The accuracies for the entire sequence are 0.51, 0.71, 0.77 and 0.86, respectively, in the *TUD-Crossing* sequence, and 0.50, 0.32, 0.52 and 0.73, respectively, in the *campus* sequence.

5. Conclusion

We proposed an explicit algorithm to detect occlusions for visual tracking. Our algorithm provides a simple but effective way to handle occlusions in tracking by learning

Figure 10. The results by applying our occlusion reasoning algorithm to IVT. **(top)** the *TUD-Crossing* sequence and **(bottom)** the *campus* sequence. See Figure 3 and Figure 4 to compare with the corresponding results of IVT without occlusion reasoning.

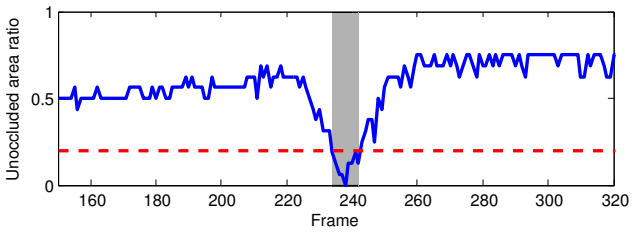


Figure 9. The degree of occlusion including degeneracy in tracking the *accidient* sequence. The target is almost fully occluded in $t = 234 \sim 242$ (shaded area). We ignore observations and track by prediction if the unoccluded area ratio is below 0.2 (dashed line).

a classifier for occlusion detection; the single classifier is trained based on observation likelihoods and can be universally employed for various objects, sequences and tracking algorithms. Our occlusion reasoning algorithm is tested intensively in various tracking scenarios with occlusions and showed significantly improved results in our experiment.

Acknowledgement

This research was supported by Basic Science Research Program through the National Research Foundation of Korea funded by the Ministry of Education, Science and Technology (2010-0003496).

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, 2006. 1

[2] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *CVPR*, 2008. 5

[3] S. Avidan. Ensemble tracking. *IEEE Trans. PAMI*, 29(2):261–271, 2007. 1

[4] CAVIAR: Context Aware Vision using Image-based Active Recognition. http://homepages.inf.ed.ac.uk/rbf/CAVIAR/. 5

[5] D. Chen and J. Yang. Robust object tracking via online dynamic spatial bias appearance models. *IEEE Trans. PAMI*, 29(12):2157–2169, 2000. 1

[6] P. Chockalingam, N. Pradeep, and S. Birchfield. Adaptive fragments-based tracking of non-rigid objects using level sets. In *ICCV*, 2009. 1

[7] R. Collins. Mean-shift blob tracking through scale space. In *CVPR*, 2003. 1

[8] S. L. Dockstader and A. M. Tekalp. Multiple camera tracking of interacting and occluded human motion. *Proceedings of IEEE*, 89(10):1441–1455, 2001. 1

[9] F. Fleuret, J. Berclaz, R. Lengagne, and P. Fua. Multicamera people tracking with a probabilistic occupancy map. *IEEE Trans. PAMI*, 30(2):267–282, 2008. 1

[10] V. Gay-Bellile, A. Bartoli, and P. Sayd. Direct estimation of non-rigid registrations with image-based self-occlusion reasoning. *IEEE Trans. PAMI*, 32(1):87 –104, 2010. 1

[11] H. Grabner, J. Matas, L. Van Gool, and P. Cattin. Tracking the invisible: Learning where the object might be. In *CVPR*, 2010. 1

[12] B. Han and L. Davis. On-line density-based appearance modeling for object tracking. In *ICCV*, 2005. 1

[13] B. Han and L. S. Davis. Probabilistic fusion-based parameter estimation for visual tracking. *CVIU*, 113(4):435–445, 2009. 1

[14] M. Isard and A. Blake. Condensation - Conditional density propagation for visual tracking. *IJCV*, 29(1), 1998. 2

[15] A. Jepson, D. Fleet, and T. El-Maraghi. Robust online appearance models for visual tracking. In *CVPR*, 2001. 1, 5

[16] J. Kwon and K. Lee. Tracking of abrupt motion using wang-landau monte carlo estimation. In *ECCV*, pages I: 387–400, 2008. 1

[17] H. Lim, O. I. Camps, M. Sznaier, and V. I. Morariu. Dynamic appearance modeling for human tracking. In *CVPR*, 2006. 1

[18] X. Mei and H. Ling. Robust visual tracking using l_1 minimization. In *ICCV*, 2009. 2, 4, 5

[19] A. Mittal and L. S. Davis. M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene. *IJCV*, 51(3):189–203, 2003. 1

[20] H. Nguyen and A. Smeulders. Tracking aspects of the foreground against the background. In *ECCV*, May 2004. 1

[21] D. Ross, J. Lim, and M. Yang. Adaptive probabilistic visual tracking with incremental subspace update. In *ECCV*, 2004. 1, 2, 5

[22] J. Shi and C. Tomasi. Good features to track. In *CVPR*, 1994. 3

[23] E. Sudderth, M. Mandel, W. Freeman, and A. Willsky. Distributed occlusion reasoning for tracking with nonparametric belief propagation. In *NIPS*. MIT Press, 2004. 1

[24] J. Wright, A. Y. Yang, A. Ganesh, S. S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. PAMI*, 31:210–227, February 2009. 2

[25] Y. Wu, T. Yu, and G. Hua. Tracking appearances with occlusions. In *CVPR*, 2003. 1

[26] M. Yang, Y. Wu, and G. Hua. Context-aware visual tracking. *IEEE Trans. PAMI*, 31(7):1195–1209, 2009. 1

[27] M. Yang, J. Yuan, and Y. Wu. Spatial selection for attentional visual tracking. In *CVPR*, 2007. 1

[28] X. Zhou and Y. Lu. Abrupt motion tracking via adaptive stochastic approximation monte carlo sampling. In *CVPR*, 2010. 1