Initialization-Insensitive Visual Tracking Through Voting with Salient Local Features

Kwang Moo Yi*, Hawook Jeong*, Byeongho Heo*, Hyung Jin Chang**, and Jin Young Choi*

*Department of EECS, ASRI
Seoul National University, Seoul, Korea
{kmyi, hwjeong, ho6946, jychoi}@snu.ac.kr

**Department of EEE
Imperial College, London, UK
hj.chang@imperial.ac.uk

Abstract

In this paper we propose an object tracking method in case of inaccurate initializations. To track objects accurately in such situation, the proposed method uses "motion saliency" and "descriptor saliency" of local features and performs tracking based on generalized Hough transform (GHT). The proposed motion saliency of a local feature emphasizes features having distinctive motions, compared to the motions which are not from the target object. The descriptor saliency emphasizes features which are likely to be of the object in terms of its feature descriptors. Through these saliencies, the proposed method tries to "learn and find" the target object rather than looking for what was given at initialization, giving robust results even with inaccurate initializations. Also, our tracking result is obtained by combining the results of each local feature of the target and the surroundings with GHT voting, thus is robust against severe occlusions as well. The proposed method is compared against nine other methods, with nine image sequences, and hundred random initializations. The experimental results show that our method outperforms all other compared methods.

1. Introduction

Various tracking methods have been developed over the past decade and have proven to be successful for many applications [22]. To track objects accurately, problems such as non-rigid deformations [7, 14, 9], partial occlusions [1, 17], and drifting [20, 4] have been tackled giving promising results. However, the applicability of conventional methods are still limited in actual environments due to their sensitivity to initializations (illustrated in Figure 1). One of the major assumptions many conventional trackers have is that the target object is given rather accurately. Therefore, the trackers are sensitive to how they were initialized at the first frame. Practically, this is not a trivial task and leads to loss in tracking performances. Also, in actual environments, severe occlusions exist where almost all of the target object is occluded, which not many trackers are capable of dealing with.

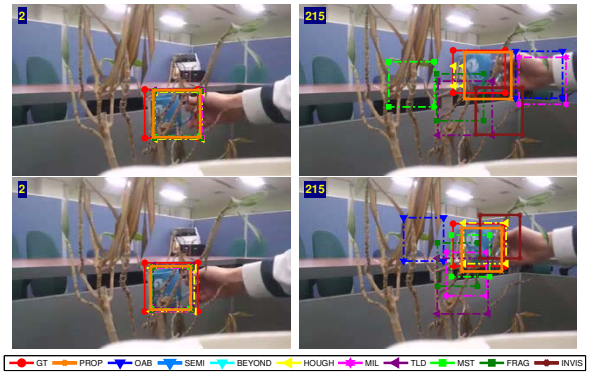


Figure 1: Example of tracking in case of occlusions and with clumsy initializations. Slight difference in initializations for the **occCup** sequence (left column), leading to different results for the same frame (right column). *Best viewed in color.*

The performance degradation of conventional trackers under inaccurate initializations is a problem which has not been well addressed yet. There are methods with automatic initialization for trackers such as the method by Mahadevan and Vasconcelos [16], but still, they do not account for cases when these initializations fail to give an accurate initialization. Performance degradation from inaccurate initializations is closely related to drifting problems. In many tracking methods, such as [7, 1, 10, 11, 20], the model for the target object is constructed using the initial target information given at the first frame. Then, the methods *adapt* the target model with the tracking results for each frames. During the adaptation, rather than learning about the target object and enhancing the model, noise gets involved in the learning process and the performance of the tracker degrades. This drifting phenomenon is evident even in online

boosting based methods [10] or methods which create classifiers [3, 16]. This drifting would cause trackers to be more sensitive to initializations, since having more noise from the beginning would cause faster drifting. In [11], [20], and [4], the authors tackle the drifting problem with sophisticated learning strategies. Both methods show robust results against drifting, but still are vulnerable against inaccurate initializations. A recent method by Kalal *et al*. [13] uses P-N learning scheme, which incorporates results from both detector and tracker to avoid such problems, but still, their work is mainly focused on the problem of trackers drifting. Even without considering drifting problems, a small inaccuracy in initialization can cause much problem.

A major cause for conventional trackers being over sensitive to initializations is much based on the fact that they treat the given initialization as a fixed prior. When we only have a single frame to use, constructing the target model solely based on the initialization is reasonable. However, as more frames or more data is given, it is obvious that we would need to *figure out* what the target object is like, rather than just trying to *adapt* the model we learned from the initialization. This means that we need to question the given prior and not believe it thoroughly. Godec *et al*. [9] recognize this problem as a limitation of bounding-box based approaches. Using the initialization by a bounding-box, they build a classifier to roughly classify pixels into foreground and background. With the rough classification result, they again perform Grab-cut [19] to segment the target object. Though their aim was to well-describe non-rigid deformations and reduce drifting effects, their method is closely related to the initialization problem of trackers. Unfortunately, their method suffers from randomness of the performance due to the purely random nature of their classifier.

To track objects robustly with inaccurate initializations and severe occlusions, we propose a method employing *motion saliency* and *descriptor saliency* of local features to learn and track the target object based on GHT (a voting based method which combines partial solutions effectively to obtain a global solution) [5]. In order to achieve good tracking results even with inaccurate initializations, we define the two saliencies related to motions and descriptors (explained in detail in Section 2). With the two proposed saliencies, our model learns to put more weight on good partial results when obtaining the global solution with GHT voting. In other words, rather than just trying to find what was given at initialization, the two saliencies work together to learn the salient characteristics of the target object, which also results in change of the influences of initial features. Method in [16] also uses the concept of saliency, but their definition of saliency is a criterion for selection of features (features such as colors or or edges, not to be confused with feature points) similar to [3]. Also, the bottom-up saliency they use for initialization is a center-surround saliency (unlike ours which considers target and non-target rather than center and surround), and does not always guarantee that it highlights the target object.

With the two saliencies, we use GHT in order to properly combine the estimates from multiple local feature points. GHT is a powerful method for combining partial estimates into a whole used in many tracking methods [9, 12, 2]. Unlike them, in our case, the mapping table of GHT containing partial solutions (votes) acts as a model learning descriptor saliencies. Since we use GHT voting to combine multiple estimates from local features, our method gives robust results even if some features of the target object becomes occluded. Furthermore, we learn the model using all local features in the scene and keep local features which move along with the target object, thus giving robust results even when all of the target object is occluded (and in case of severe occlusions). This is similar to the method by Grabner *et al*. [12], which creates *supporters* with nearby features, and use them to aid tracking in case of severe occlusions. However, the performance of their method relies much on the primary tracker being used, which is not always accurate and suffers from initialization problems, whereas our method successfully deals with both problems simultaneously. Dinh *et al*. [8] also use supporters to aid tracking, but their method still treats initialization to be accurate.

2. Proposed Method

The overall scheme of the proposed method is depicted in Figure 2. Our method is based on GHT. For each frame, we extract local features. Then, using the learned feature database (DB), we match each feature to obtain partial solutions for the center of the target object. We then combine the partial results with GHT to obtain a global solution. When collecting the partial results using GHT, in order to deal with inaccurate initializations, each solution is weighted according to the two proposed saliencies (the descriptor saliency and the motion saliency).

The feature DB is learned *on-the-fly* during the tracking process. The feature DB keeps track of distinctive features w.r.t. their descriptors (descriptor saliency), and where the center of the target object would be for each item. The proposed motion saliency is obtained using the learned descriptor saliency of features and the optical flow of the local feature. The motion saliency is designed so that the features showing distinctive motion characteristics of the target object have higher values. With the two saliencies, the salient characteristics of the target object are learned in the model (feature DB). Details of the proposed method are explained in the following subsections.

2.1. Tracking based on GHT

The proposed tracking scheme using GHT starts by building a likelihood map for the center position of the tar-

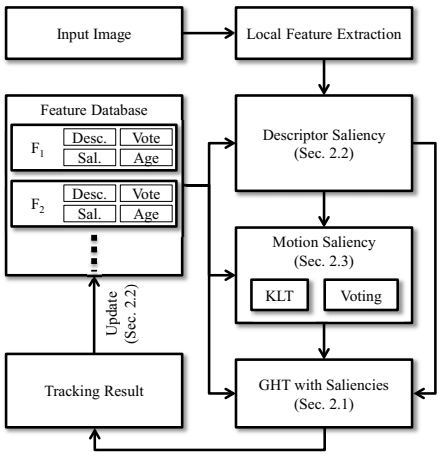


Figure 2: Overall scheme of the proposed method.

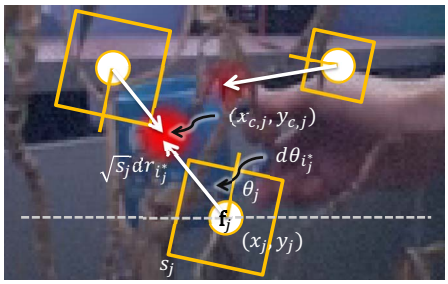


Figure 3: Illustration of GHT voting with SURF features

get object and obtains the result by finding the maximum on this map. The likelihood map is created through GHT by combining the center estimates (votes) from each local feature point. When combining the estimates, we weight them w.r.t. their saliencies so that salient features are more accounted for. Figure 3 is an illustration of this process. Mathematically, if we denote the estimated center for the j^{th} feature in the current observation as $(x_{c,j}, y_{c,j})$ and its weight as w_j, the likelihood map $\mathbf{A}$ is defined as

$$\mathbf{A}(x,y) = \sum_j w_j \exp\left\{-\frac{(x_{c,j}-x)^2 + (y_{c,j}-y)^2}{2\sigma_{\mathbf{A}}^2}\right\}, \tag{1}$$

where $\sigma_{\mathbf{A}}$ is the standard deviation of the Gaussian kernel used for combining estimates, which is a parameter controlling the smoothness of the likelihood map. This combining process is refered to as voting in GHT, each partial estimates as votes, w_j as voting weights, and the resultant likelihood map as the vote map. Since the target object position changes little between frames, we apply temporal low-pass filtering (temporal weighted averaging) to the vote map to take advantage of this fact. Then, with this vote map, we can find the target object position $\hat{\mathbf{x}}$ as

$$\hat{\mathbf{x}} = (\hat{x}, \hat{y}) = \arg\max_{x,y} \mathbf{A}(x,y). \tag{2}$$

Local Estimates. The estimates from each local feature point are obtained by matching with the feature DB. The feature DB consists of multiple items, of which each item contains descriptor information ($\mathbf{d}$), distance to the estimated object center normalized with the square-root of the size of the feature (dr), angle difference between the major orientation and the vector to the estimated center ($d\theta$), saliency of the item (ζ), and the age (α) of the item. If we denote i^{th} item of the feature DB $\mathbf{F}$ as $\mathbf{F}_i$ then,

$$\mathbf{F}_i = (\mathbf{d}_i, dr_i, d\theta_i, \zeta_i, \alpha_i). \tag{3}$$

For the j^{th} local feature of the current scene, we denote it as $\mathbf{f}_j = (\mathbf{x}_j, \mathbf{d}_j, s_j, \theta_j)$, where $\mathbf{x}_j = (x_j, y_j)$, $\mathbf{d}_j$, s_j, and θ_j is the pixel position, the descriptor, the size (area), and the major orientation of the feature, respectively. Then, we find the best matching item using the learned feature DB $\mathbf{F}^{(t)}$ at time t (learning strategy detailed in Section 2.2) in terms of $\mathbf{d}_j$ and $\mathbf{d}_i^{(t)}$, and use this match for voting. In other words, if there exists i_j^* such that

$$i_j^* = \arg\min_i \left\{ \left\| \mathbf{d}_i^{(t)} - \mathbf{d}_j \right\| \mid \left\| \mathbf{d}_i^{(t)} - \mathbf{d}_j \right\| < \epsilon_d \right\}, \tag{4}$$

where ϵ_d is a threshold, we consider feature j to be matched with $\mathbf{F}_{i_j^*}^{(t)}$ and vote to $(x_{c,j}, y_{c,j})$ with weight w_j as in Figure 3, where

$$x_{c,j} = \sqrt{s_j} dr_{i_j^*} \cos\left(d\theta_{i_j^*} + \theta_j\right) + x_j \tag{5}$$

$$y_{c,j} = \sqrt{s_j} dr_{i_j^*} \sin\left(d\theta_{i_j^*} + \theta_j\right) + y_j. \tag{6}$$

Any type of affine invariant feature can be used such as SIFT [15] or SURF [6] but we used SURF for ease in implementation. Note that in (5) and (6), we take advantage of the affine invariant properties of the local features and compensate the voting vector to fit the current observation with $\sqrt{s_j}$ and θ_j. This lets the proposed method to be able to vote regardless of the scale and rotation change of the target object.

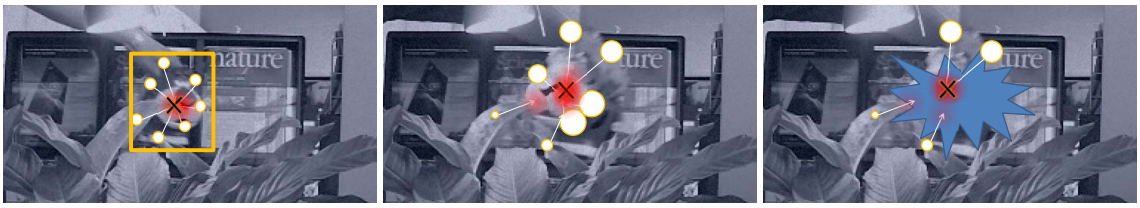
Voting Weights. The weight w_j is designed to account both the motion saliency η_j and the descriptor saliency $\zeta_{i_j^*}^{(t)}$. We use the multiplication of the two saliencies to emphasize features which have both saliency values high. Therefore for the j^{th} detected local feature, the weight is defined as

$$w_j = \eta_j \zeta_{i_j^*}^{(t)}. \tag{7}$$

Details about these saliencies and their effects are presented in Section 2.2 and Section 2.3.

2.2. Descriptor Saliency and Feature DB Update

To learn the salient features of the target object w.r.t. the shape of the target object, we define the descriptor saliency



(a) (b) (c)

Figure 4: Illustration of the descriptor saliency in action. Detected local features are depicted with circles, having their sizes as their descriptor saliency values (larger means high). Red means having higer vote map value $\mathbf{A}$. (a) Initial voting for the target object position, (b) voting after t frames, and (c) voting in case of occlusion. Note that these are not actual experimental results and are only illustrations.

as how much the descriptor coincided with past consensus. For each item in $\mathbf{F}^{(t)}$, the descriptor saliency $\zeta^{(t)}$ is learned to hold how good the partial (voting) results were when using the item, *i.e*. the value of the vote map $\mathbf{A}$. Since our framework is based on GHT, we can simply achieve this by looking at each votes and DB matches (back projection). For example, the feature items matched with the features pointing to the center of the cup in Figure 3 would be updated with high saliency values, whereas items matched with the feature pointing at the wrong direction (*e.g*. feature point on the hand) would be updated with low saliency value.

Initially, $\mathbf{F}^{(0)}$ is an empty set without any elements. As the target object information is provided with a bounding box in the first frame, feature points inside the bounding box are added to $\mathbf{F}^{(1)}$ with descriptor saliency $\zeta = 1$, and feature points outside the bounding box are added to $\mathbf{F}^{(1)}$ with descriptor saliency $\zeta = 0$ (Figure 4a). Then, we continuously update the descriptor saliencies using the back projection result of each vote on $\mathbf{A}^{(t)}$. This means that at time t, if item $\mathbf{F}_i^{(t)}$ has been matched with M_i features (*i.e*. there exists M_i number of j such that $\left\|\mathbf{d}_i^{(t)} - \mathbf{d}_j\right\| < \epsilon_d$), the descriptor saliency of this item $\zeta_i^{(t)}$ is updated with the average of the vote map value for all matches.

$$\zeta_i^{(t+1)} = \beta_i \zeta_i^{(t)} + (1-\beta_i)\frac{1}{M_i}\sum_{m=1}^{M_i}\mathbf{A}\left(x_{c,m}, y_{c,m}\right) \quad (8)$$

where m is the index of the matched local feature in the current frame, β_i is the variable learning rate $\beta_i = \alpha_i^{(t)}/(\alpha_i^{(t)}+1)$. Similarly, we update the voting information of the item with the average normalized distance and angle from each feature to the obtained tracking result, and also increment the age of the item. At time t, with the tracking result $\hat{\mathbf{x}}^{(t)}$ and position of the matched features $\mathbf{x}_m$

$$dr_i^{(t+1)} = \beta_i dr_i^{(t)} + (1-\beta_i)\frac{1}{M_i}\sum_{m=1}^{M_i}\frac{\left\|\hat{\mathbf{x}}^{(t)} - \mathbf{x}_m\right\|_2}{\sqrt{s_m}} \quad (9)$$

$$d\theta_i^{(t+1)} = \beta_i d\theta_i^{(t)} + (1-\beta_i)\frac{1}{M_i}\sum_{m=1}^{M_i}\left[\angle\left(\hat{\mathbf{x}}^{(t)} - \mathbf{x}_m\right) - \theta_m\right] \quad (10)$$

$$\alpha_i^{(t+1)} = \alpha_i^{(t)} + 1, \quad (11)$$

where $\|.\|_2$ and $\angle(.)$ is the Euclidean norm and the angle of a vector, respectively. If an element in $\mathbf{F}^{(t)}$ has no match, *i.e*. $M_i = 0$, we do not update that element. Also, the elements of $\mathbf{F}^{(t)}$ which were added in the first frame are treated as an exception and we never update $dr_i^{(t)}$ and $d\theta_i^{(t)}$ for them to prevent the tracker from drifting. Still, since we update $\zeta_i^{(t)}$, the effects of initial elements can change. Figure 4a and Figure 4b is an example of some local features (features on leaves) having high descriptor saliency at initialization, but becoming low after learning them correctly as tracking is performed.

Practical Implementation. Ideally, it would be best if we add all new unmatched features into the database, but this would not be practical due to computational and memory requirements. Thus, to keep $\mathbf{F}^{(t)}$ in a reasonable size without harming the overall performance, we apply an update strategy inspired by the work of Avidan [3]. Except for the elements of $\mathbf{F}^{(t)}$ which were added in the first frame, we keep the K best elements in terms of $\zeta_i^{(t)}$ and get rid of others from $\mathbf{F}^{(t)}$. After removing bad elements from $\mathbf{F}^{(t)}$, we add L most motion-salient unmatched feature points into the DB. The L best considering the motion saliency η_j are added to $\mathbf{F}^{(t+1)}$ with its motions saliency as initial descriptor saliency, *i.e*. the following 5-tuple

$$(\mathbf{d}_j, dr_j, d\theta_j, \eta_j, 1) \quad (12)$$

is added, where

$$dr_j = \frac{\left\|\hat{\mathbf{x}}^{(t)} - \mathbf{x}_j\right\|_2}{\sqrt{s_j}}, \quad (13)$$

$$d\theta_j = \angle\left(\hat{\mathbf{x}}^{(t)} - \mathbf{x}_j\right) - \theta_j. \quad (14)$$

With this update strategy, if we denote the number of elements initially added to $\mathbf{F}^{(1)}$ as K_{init}, the size of $\mathbf{F}^{(t)}$ will

always be smaller than $K_{init} + K + L$. Note that during the feature DB update, all current local features are considered. This means that if there are background local features which help in estimating the target object position, they will also be learned. These un-occluded salient features can act as supporters similar to [12], aiding in case of severe occlusions as in Figure 4c.

2.3. Motion Saliency

To capture the characteristics of the target object in terms of motion, we define the motion saliency of a feature point with its descriptor saliency and optical flow (Figure 5b), and emphasize motions which are distinctive (Figure 5d). By distinctive, we expect the motion of the target object to stand-out from background motion (including motion from other non-target objects). For obtaining motions for each feature points, we use backward optical flow from time t to time $t-1$. The backward optical flow $\mathbf{b} = (b_x, b_y)$, where b_x and b_y are backward optical flows in horizontal and vertical direction, can be easily obtained though KLT [21].

The way we measure the distinctiveness is by constructing a likelihood map of background motions through consensus (Figure 5c). This likelihood map is constructed using voting strategy similar to the GHT voting used for tracking. When constructing the likelihood map, we weight each motion of the detected local feature points so that the resultant likelihood map would have higher values if the motion is likely to be from background. Therefore, with the likelihood map we are able to know which motions are similar to background motions and which are distinct. We will refer to this likelihood map as the *motion vote map*. We obtain the motion saliency of feature point by simply using the inverse of the motion vote map value. By doing so, the motion saliency value will be high when the vote map value is low, *i.e*. not similar to background motions. If we denote the motion vote map as $\mathbf{B}$, then for a feature point $\mathbf{f}_j$ in the current observation with backward optical flow $\mathbf{b}_j$, the motion saliency for this feature η_j can be defined as

$$\eta_j = 1 - \frac{\mathbf{B}(\mathbf{b}_j)}{\max \mathbf{B}(.)}, \tag{15}$$

where the motion vote map $\mathbf{B}$ is constructed by all features in the current frame having a matched item i_j^* from (4) as

$$\mathbf{B}(\mathbf{b}) = \sum_{\forall j | \exists i_j^*} \left(1 - \zeta_{i_j^*}^{(t)}\right) \exp\left\{-\frac{\|\mathbf{b} - \mathbf{b}_j\|^2}{2\sigma_{\mathbf{B}}^2}\right\}, \tag{16}$$

where $\sigma_{\mathbf{B}}$ is the standard deviation of the Gaussian kernel used for voting. Note that for the motion vote map, we weight the votes with $\left(1 - \zeta_{i_j^*}^{(t)}\right)$, which is the inverse of descriptor saliency so that the vote map is about background motions. η_j from (15) has a value in range $[0, 1]$, representing how distinctive the backward optical flow of a feature point is from background motions.

The advantage of our method using motion vote map is that we are able to capture distinctive motions regarding the target object robustly without any sophisticated motion grouping. As in Figure 5d, even though there are other motions due to the camera movement, only the distinctive motions similar to the target object is emphasized.

3. Experiments

We implemented our method (PROP) in C++ with the OpenCV[1] library for SURF and KLT. For all experiments, the threshold for determining feature point matches $\epsilon_d = 0.1$, the variances of the Gaussian kernels for voting $\sigma_{\mathbf{A}}$ and $\sigma_{\mathbf{B}}$ are both set to 10, the number of elements to keep $K = 500$, and the number of features added $L = 100$. We also considered tracking results having $\mathbf{A}^{(t)}\left(\hat{\mathbf{x}}^{(t)}\right) < 0.05$ as tracking failures.

We tested our method against nine other representative trackers. MST [7] and FRAG [1] are kernel-based trackers, where FRAG is mainly focused on solving occlusion problems. OAB [10], SEMI [11], BEYOND [20], and MIL [4] are boosting based methods. HOUGH [9] is a method based on Hough forests and TLD [13] is a method with P-N learning strategy combining the result of both tracker and detector. SEMI, BEYOND, MIL, and TLD are mostly targeted for solving drifting issues and HOUGH is a method targeted to overcome inaccuracies arising from a bounding box representation of the target. Finally INVIS [12] is a method using GHT similar to ours. For the experiments, we used the implementation provided by the authors of each paper except for MST and INVIS. For MST and INVIS, we implemented them in C++ according to the papers. We performed our experiments with nine image sequences. Sequences **coke**, **tiger1**, and **tiger2** are from [4], **sylvester** is from [18], **occFace** and **woman** are from the [1], **motocross1** and **mtn. bike** are from [9], and **occCup** is our own. Implementation of the proposed method and the datasets used are available at the first author's website[2]. . Results for critical frames are shown in Figure 7.

To compare results quantitatively, we used manually annotated bounding box representation of the target object as the ground truth. Two measures were used to evaluate the algorithms; mean error between the ground truth center point and the tracking result, and the percentage of correctly tracked frames. By correctly tracked frames, we counted the tracking result as correct if the center of the tracking result was inside the ground truth bounding box. The reason we applied this measure instead of the commonly used over-

[1] http://opencv.com/downloads.html

[2] https://sites.google.com/site/homekmyi

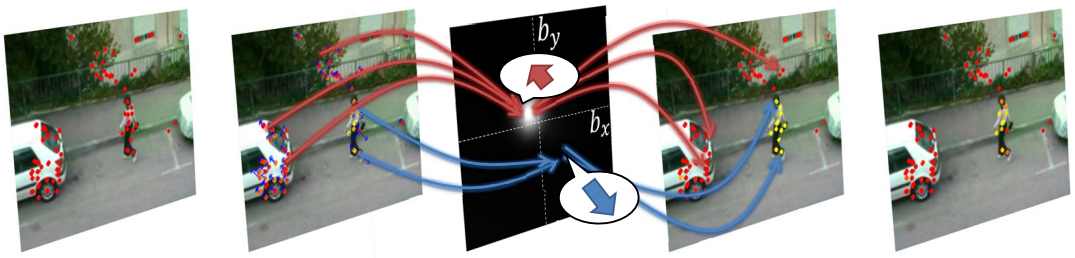


Figure 5: Example of motion saliency obtained for the **woman** sequence. (a) Detected local feature points, (b) descriptor saliency of matched local features and their optical flows (denoted yellow if high saliency and red if low, optical flows displayed 3 times their original magnitude), (c) motion vote map $\mathbf{B}$, (d) motion saliency for each detected local feature, and (e) final voting weight from both saliencies. Arrows from (b) to (c) and (c) to (d) illustrate where motions are mapped. Note that in (b), upper left motion on the cars result in low motion saliency values in (d) whereas down right motion on the person result in high values. *Best viewed in color.*

lap criterion is because we are using random initializations which may have little overlap even from the beginning. This measure can be understood as a weak condition of tracking success. When the overlap measure is used, results for all trackers become degraded since some initializations would be counted as failures even at the first frame. Still, the relative performance of trackers remain similar since this happens equally for all trackers. For trackers being able to detect tracking failures, we did not use these frames for computing the mean error. However, they are considered as tracking failures in the percentage of correctly tracked frames for fair comparison.

3.1. Tracking with Inaccurate Initializations

To validate the robustness of our method against clumsy initializations, we have tested trackers with 100 random initializations. Of the 100, the first initialization is identical to the ground truth and 20 contain initializations having the center point of the bounding box fixed at the ground truth but having different width and height (sampled uniformly having maximum difference to be 20% of the original width or height). Another 20 contain initializations having the same width and height as the ground truth, but with the center point differing (sampled uniformly having maximum difference to be 50% of the width or height of the target object.) Finally, the remaining 59 have both the width and height, and the center point differing from the ground truth in the same sense above.

Results for all sequences with all initializations are shown in Table. 1 (PROPm and PROPd are the results of our method using only motion saliency and descriptor saliency, respectively). Initialization overlap in the table is defined as the percentage ratio between the intersection and the union of the initial bounding box and the ground truth. In Table. 1, it can be observed that as initialization overlap decreases, the average performance of trackers generally degrade (lower percentage of correctly tracked frames and larger mean errors). For percentage of correctly tracked frames, when considering best results, our method shows 94.2% whereas the second best except the results of our method is TLD with 60%∼40% overlap showing 89.3% (denoted by bold blue text). However, when considering the average performance, with the same condition, our method shows 84.5% whereas TLD shows 51.3%. Note that the gap between the best performance and the average performance our method is relatively small compared to other methods. This shows that our method is less sensitive to initializations. Also, in terms of average performance, our method significantly outperforms other methods including INVIS, which uses BEYOND as a primary tracker and uses GHT similar to ours. In case of mean error in the correctly tracked frames, our method is not best but shows comparable results against other methods. Occasionally, BEYOND shows best results in terms of mean error, but the percentage of correctly tracked frames shows that only a limited number of frames were tracked. Note that using only one of the two saliencies degrade performance (PROPm and PROPd).

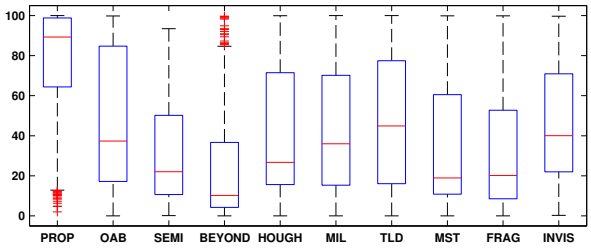


Figure 6: Box plots for % correctly tracked with all initializations.

Figure 6 is a box plot demonstrating the performance of trackers against different initializations. The whiskers (denoted with red pluses) are data points having values more

than 1.5 inter quartile range away from the median (red line). In Figure 6, it can be seen that the best performances (dotted lines) do not differ much. This implies that with a particular initialization designed for each algorithm, the performances may not differ much. However, this is not a trivial task, and when considering average performance, our method outperforms all other compared methods by significant amounts.

3.2. Tracking Under Occlusions

To evaluate the performance of our method against occlusions, we have tested our method on sequences **coke**, **tiger1**, **tiger2**, **occFace**, **woman**, and **occCup**. In these sequences, occlusion of the target object exist, expecially with the **coke** sequence having the object fully occluded at occasions, and **occCup** having the object occluded even at initialization. Critical frames for sequences with occlusions are shown in Figures 7a to 7i. Figure 7b is an example of severe occlusion where the target object gets fully occluded. Our method successfully tracks the target object even in such case, by learning the features of the hand which moves together with the target object. In Figure 7e, the target object is occluded even at initialization. As in Figure 7f, many compared methods fail to recognize the cup as the target object and fail. However, our method successfully tracks the target object.

4. Conclusions

A new visual tracking method for tracking objects in case of inaccurate initialization has been proposed. The proposed method uses *motion saliency* and *descriptor saliency* of local features and obtains the target position through GHT. The motion saliency of a local feature emphasizes features having distinctive motions, compared to background motions. The descriptor saliency emphasizes features which are likely to be of the object in terms of its feature descriptors. Through these saliencies, the proposed method learns the distinctive characteristics of the target object in the image sequence. The saliencies and GHT combined allowed the tracker to have robust performances under clumsy initializations and occlusions.

The proposed method was extensively tested against nine other methods, using nine image sequences, and with hundred random initializations. The experimental results demonstrated the robustness of our method against initializations and (severe) occlusions, outperforming the other compared methods significantly.

Acknowledgement

The research was sponsored by the SNU Brain Korea 21 Information Technology program, and the IT R&D program of MKE & KEIT [10041610, The development of the recognition technology for user identity, behavior and location that has a performance approaching recognition rates of 99% on 30 people by using perception sensor network in the real environment]

References

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

[2] M. Asadi, A. Dore, A. Beoldo, and C. Regazzoni. Tracking by using dynamic shape model learning in the presence of occlusion. In *AVSS*, 2007.

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

[4] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *PAMI*, 33(8):1619 –1632, 2011.

[5] D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[6] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Speeded-up robust features (surf). *CVIU*, 110(3):346–359, 2008.

[7] D. Comaniciu, V. Ramesh, and P. Meer. Kernel-based object tracking. *PAMI*, 25(5):564 – 577, 2003.

[8] T. B. Dinh, N. Vo, and G. Medioni. Context tracker: Exploring supporters and distracters in unconstrained environments. In *CVPR*, 2011.

[9] M. Godec, P. Roth, and H. Bischof. Hough-based tracking of non-rigid objects. In *CVPR*, 2011.

[10] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, 2006.

[11] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *ECCV*, 2008.

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

[13] Z. Kalal, J. Matas, and K. Mikolajczyk. P-n learning: Bootstrapping binary classifiers by structural constraints. In *CVPR*, 2010.

[14] J. Kwon and K. M. Lee. Tracking of a non-rigid object via patch-based dynamic appearance modeling and adaptive basin hopping monte carlo sampling. In *CVPR*, 2009.

[15] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[16] V. Mahadevan and N. Vasconcelos. Saliency-based discriminant tracking. In *CVPR*, 2009.

[17] X. Mei and H. Ling. Robust visual tracking using l_1 minimization. In *ICCV*, 2009.

[18] D. A. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *IJCV*, 77:125–141, 2008.

[19] C. Rother, V. Kolmogorov, and A. Blake. "grabcut": interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23:309–314, 2004.

[20] S. Stalder, H. Grabner, and L. van Gool. Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In *ICCVW*, 2009.

[21] C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, CMU, 1991.

[22] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Comput. Surv.*, 38(4), 2006.

	Algorithm	Initialization Overlap				
		100% ∼ 80%	80% ∼ 60%	60% ∼ 40%	40% ∼ 20%	20% ∼ 0%
Correctly Tracked Frames (%)	PROP	**87.7±15.3 (90.5)**	**88.0±15.7 (92.6)**	**84.6±18.6 (94.2)**	**72.8**±27.0 (**92.9**)	**55.3**±32.9 (**75.8**)
	PROPm	80.7±20.7 (82.2)	77.2±23.8 (85.4)	70.5±27.2 (88.5)	64.1±29.5 (87.7)	46.9±34.3 (70.0)
	PROPd	41.3±39.9 (42.6)	40.8±34.9 (47.9)	38.2±32.5 (48.5)	34.2±30.5 (48.1)	22.8±25.3 (34.0)
	OAB	55.0±35.1 (76.3)	55.4±34.7 (80.6)	50.5±34.6 (88.6)	43.0±32.6 (78.4)	34.8±31.2 (62.9)
	SEMI	28.3±23.0 (46.2)	34.3±24.6 (54.7)	30.8±23.0 (65.9)	31.2±24.6 (67.9)	23.9±23.4 (42.2)
	BEYOND	28.1±29.7 (44.1)	23.9±27.8 (55.7)	22.9±24.8 (57.6)	21.7±**23.8** (56.8)	18.9±21.7 (39.9)
	HOUGH	50.0±34.0 (63.9)	46.9±33.1 (71.3)	44.1±33.1 (79.1)	37.7±33.6 (77.1)	26.4±29.6 (51.7)
	MIL	56.0±32.2 (62.8)	52.5±30.6 (70.3)	46.7±32.3 (75.4)	37.8±31.6 (74.1)	24.4±25.7 (55.7)
	TLD	61.7±32.2 (72.0)	54.3±30.1 (81.8)	51.3±33.7 (**89.3**)	41.3±31.7 (88.5)	26.6±26.4 (58.5)
	MST	43.6±38.1 (50.5)	43.7±37.2 (55.2)	40.1±35.1 (62.4)	29.5±27.3 (56.2)	17.8±**17.5** (31.9)
	FRAG	43.8±33.9 (57.5)	46.1±31.4 (64.2)	35.6±28.8 (59.1)	25.7±28.5 (46.6)	16.0±23.9 (31.4)
	INVIS	50.2±31.4 (71.5)	51.5±29.0 (78.2)	47.9±27.5 (77.5)	43.2±27.2 (71.1)	33.4±26.1 (56.7)
Mean Error (pixels)	PROP	21.1±19.7 (17.8)	20.7±17.8 (13.4)	**25.2±19.2** (15.5)	**34.5±24.4** (19.8)	**47.6±31.6** (**33.7**)
	OAB	38.4±41.6 (14.8)	41.1±35.8 (18.3)	45.4±33.5 (18.6)	50.4±34.5 (23.3)	63.6±40.0 (35.6)
	SEMI	19.6±21.7 (9.6)	23.0±17.7 (8.0)	41.0±37.7 (16.6)	52.5±36.8 (23.1)	72.7±45.7 (45.4)
	BEYOND	**12.0±12.9 (6.1)**	**14.5±12.3 (4.2)**	25.9±19.3 (**7.2**)	38.8±27.1 (**14.0**)	57.0±34.7 (37.8)
	HOUGH	58.3±45.5 (31.5)	56.8±41.9 (26.6)	57.8±42.4 (24.8)	64.2±44.7 (24.6)	89.1±55.7 (44.7)
	MIL	39.3±34.4 (20.5)	43.6±35.6 (14.9)	48.9±38.7 (17.3)	60.5±43.2 (20.7)	73.4±49.4 (39.1)
	TLD	85.7±95.1 (52.3)	87.0±89.2 (40.4)	92.1±93.8 (30.7)	101.5±97.1 (38.9)	124.2±103.4 (75.4)
	MST	75.1±56.6 (51.6)	69.0±51.5 (48.5)	75.6±54.3 (38.4)	83.8±56.4 (49.2)	100.7±58.6 (76.9)
	FRAG	66.8±47.4 (44.7)	65.3±46.2 (40.7)	81.4±55.9 (49.0)	95.3±66.1 (72.9)	113.7±70.7 (86.9)
	INVIS	60.1±52.0 (34.6)	60.8±50.5 (32.1)	63.2±48.0 (36.3)	68.8±49.4 (42.5)	83.5±50.9 (54.3)

Table 1: Results for all sequences. [average±standard deviation (best average)]. For best average, best results for all sequences were averaged. Bold denotes best result among the compared algorithms.

(a) **coke** #2 (b) **coke** #256 (c) **tiger1** #106 (d) **tiger2** #239

(e) **occCup** #2 (f) **occCup** #211 (g) **woman** #144 (h) **woman** #301

(i) **occFace** #562 (j) **sylvester** #1093 (k) **motocross1** #75 (l) **mtn. bike** #132

Figure 7: Critical frames for tracking results. Subcaptions denote sequence names and frame numbers. *Best viewed in color.*