# Gracker: A Graph-based Planar Object Tracker

Tao Wang and Haibin Ling

**Abstract**—Matching-based algorithms have been commonly used in planar object tracking. They often model a planar object as a set of keypoints, and then find correspondences between keypoint sets via descriptor matching. In previous work, unary constraints on appearances or locations are usually used to guide the matching. However, these approaches rarely utilize structure information of the object, and are thus suffering from various perturbation factors. In this paper, we proposed a graph-based tracker, named *Gracker*, which is able to fully explore the structure information of the object to enhance tracking performance. We model a planar object as a graph, instead of a simple collection of keypoints, to represent its structure. Then, we reformulate tracking as a sequential graph matching process, which establishes keypoint correspondence in a geometric graph matching manner. For evaluation, we compare the proposed Gracker with state-of-the-art planar object trackers on three benchmark datasets: two public ones and a newly collected one. Experimental results show that Gracker achieves robust tracking results against various environmental variations, and outperforms other algorithms in general on the datasets.

**Index Terms**—visual tracking, keypoint, graph matching, pose estimation.

## 1 Introduction

Visual object tracking is among the core problems of computer vision, with wide-ranging applications such as augmented reality and robotics. In this work, we address the problem of tracking a planar object in an accurate and robust manner, with arbitrary motion and no prior knowledge other than its position in the first video frame. Despite tremendous amount of researches on this topic, robust, accurate and efficient pose tracking remains challenging due to challenging environmental variations.

Popular approaches to planar object tracking can be roughly classified as template-based approaches (e.g., [3], [14], [22], [34], [35]) or keypoint-based ones (e.g., [6], [20], [24], [37]). Template-based approaches directly use the appearance of the pixels without extracting features, and optimize a similarity measure between a template and a captured image, based on the Newton method or its variants, to determine the pose of the plane. This type of approaches usually suffer from perturbation factors such as illumination changes, partial occlusions and fast motions.

Keypoint-based approaches have attracted much attention during the last decade, due to their invariance against various perturbation factors including rotation, scaling and viewpoint change [36]. Moreover, they are naturally suited to handle partial occlusions as partial matches between points are sufficient for most tracking scenarios. These approaches use descriptors [4], [8], [31], [33] to store a signature for each keypoint of the object, which are designed to be invariant to various geometric and photometric transformations. These descriptors are then matched, usually in a nearest-neighbour fashion.

Though keypoint-based approaches have been widely used in planar object tracking, they are still facing two main challenges. First, recent studies revealed that focusing only on the object features without considering their structures does not ensure the robustness in real-world applications [48], [50]. The presentation of the object structure plays an important role in order to provide robust and accurate solutions. As the object of interest is unknown beforehand, it is hard to learn offline object appearance and structure. Instead, online learning algorithms have been employed [2], [18], [20] to dynamically build the appearance and structure model of the object. In practice, however, online model updating often introduces errors, as there are no faithful class labels available. Second, most existing keypoint-based approaches rely heavily on keypoint detection, and are thus fail to provide reasonable results in the cases of difficult illumination conditions and motion blur, where reliable keypoints are hard to be detected.

In this paper, we propose a novel *graph-based tracker*, named Gracker, to address the issues discussed above for improving tracking performance. Specifically, we adopt graphs to model planar objects, with graph vertices generated via a reliable automatic selection rather than conventional DoG-based detectors [33]. This selection mechanism makes graph structures stable and hence the approach robust against some dramatic environmental variations such as extreme illumination conditions and motion blur. Furthermore, we incorporate feature correspondence and pose estimation into a unified geometric graph matching framework. Pairwise constraints in graphs allow us to encode comprehensive information which results in robust and accurate solutions against various geometric and photometric transformations.

For a thorough evaluation, we test the proposed approach on three datasets: the *University of California, Santa Barbara* (UCSB) benchmark [17], the *tracking manipulation tasks* (TMT) benchmark [41], and a compiled dataset for fast motion. Experimental results show that the proposed algorithm significantly improves the tracking accuracy and outperforms state-of-the-art tracking algorithms in comparison on almost all video categories. In particular, the proposed approach achieves much higher robustness in the case of fast motion where traditional keypoint-based approaches fail to produce reliable tracking results.

In summary, our main contribution lies in the new graph-based tracker for planar object tracking in three aspects: (1) we introduce graph model and graph matching manner into planar object tracking; (2) we design a novel strategy for predicting both object pose and point matching, and integrate the strategy for

- *T. Wang is with Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing Jiaotong University, Beijing 100044, China. E-mail: twang@bjtu.edu.cn*
- *H. Ling is Computer and Information Sciences Department, Temple University, Philadelphia PA 19122, USA. (Corresponding author: Haibin Ling). E-mail: hbling@temple.edu*

searching optimal solution; and (3) we construct a new real world dataset with annotation for evaluating visual tracking algorithms in the context of fast motion. The source code of the proposed algorithm is shared at http://www.dabi.temple.edu/~hbling/code/Gracker/gracker.htm.

In the rest of the paper, we summarize related work in Sec. 2 and review matching-based pose tracking in Sec. 3. Then, we introduce the proposed Gracker algorithm in Sec. 4 and discuss related optimization in Sec. 5. After that, we present experimental validation in Sec. 6 and draw conclusions in Sec. 7.

## 2 Related Work

Visual object tracking has been investigated for several decades, classical early work dates back to the LK algorithm [34] for 2D template tracking and the ICP algorithm [5] for 3D object tracking. Thoroughly reviewing all visual object tracking papers is beyond the scope of this paper, in the following we sample some related ones that inspire our study on *planar object tracking*.

Keypoint-based tracking is a popular strategy for planar object tracking [19], [20], [25], [37], [42], [46]. Such algorithms generally have three stages: (1) detecting keypoints and storing their descriptors, (2) establishing keypoint correspondences between an object model and an input image, and (3) estimating the transformation of the object in the image using a robust geometric verification method based on hypotheses generated from the correspondences (e.g. RANSAC and its variants [12], [45]).

It is nontrivial to capture local appearance change by a global model, and models that decompose the object into parts are more robust to nuisances. Many approaches [1], [23], [38], [52] employ axis-aligned rectangular regions for extracting features of the object and its parts. Being computationally convenient, these methods may be sensitive to changes in scale and rotation. By contrast, keypoints and descriptors (e.g. SIFT [33]) are designed to be robust for such factors, and are therefore ideal for a parts-based model. Recently, the advent of extremely fast keypoint detectors and binary descriptors [31], [40] has dramatically reduced the computational burden of detecting and matching corresponding keypoints, allowing for their use in real-time systems. There are two main ways to establish the keypoint correspondence in visual tracking: matching and classification. Matching-based approaches [6], [37] relate keypoints by a suitable distance metric to keypoints in a nearest-neighbour fashion. Classification-based approaches [30], [39] treat matching as multi-class classification, in which each keypoint is classified as either background or a particular keypoint from the model.

Studies have shown that utilizing only local features can not ensure tracking robustness in real world applications [50]. Some recent efforts focus on exploring the object structure to improve the tracking performance. In [48], it is proposed to track multiple auxiliary objects defined as the spatial context. These auxiliary objects have consistent motion correlation with the tracked target and thus help to avoid the drifting problem. However, finding the motion correlation between the target and surrounding objects is a costly task that often requires analyzing the whole image in every frame. The Structure-Aware Tracker (SAT) [6] incorporates the internal structural information of the target, but not the structural layout of different scene elements. In this work, the author shows that the structural information of the target, encoded by the keypoint spatial layout, allows achieving accurate tracking and handling partial occlusion by inferring the position of the target using the visible keypoints. Aside from the static models, online learning algorithms have been employed [2], [18], [20] to dynamically update the structure model of the object. In practice, however, online updating may introduce errors due to the lack of hard class labels.

Alternative to the above mentioned keypoint-based approaches, learning-based ones appear to be robust to environmental perturbations. Sampled recent studies include [43] that adopts random forests to learn the relation between the motion parameters and the changes on the image intensities; and [27] that formulates the template-based visual tracking problem as a particle filtering problem on the matrix Lie group. In addition, some recent trackers designed for generic object tracking (e.g. graph-based [7], [44], correlation-based [13], [21] and flock-based [9]) have shown state-of-the-art performance on standard tracking benchmarks, but may meet problems for accurate pose (planar object) tracking.

The proposed Gracker shares with the above algorithms the use of context to assist tracking, but differs in context modeling (i.e., with graph matching) and application (i.e., for planar object tracking). In particular, our work falls into the group of structure-aware tracking, with improvement in two-fold: (1) keypoint correspondence with pairwise constraints via graph matching, and (2) a new strategy for predicting both object pose and keypoint correspondence when searching the optimal solution. Our work aims to provide robust and accurate tracking for planar objects, and the excellent experimental results clearly validate its advantage.

## 3 Matching-based Tracking

### 3.1 Problem statement

Given a sequence of images $\mathcal{I}_1, \ldots, \mathcal{I}_m$ , and a planar object of interest $\mathcal{R}$ (in the form of a patch in $\mathcal{I}_1$), pose tracking aims to acquire the pose of the object in the following frames, or to report the object missing when it is invisible. The relative motion between the object and the camera induces changes in the position of the object in the image. We assume that these transformations can be modeled by a geometric transformation $\tau(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ parameterized by $\mathcal{T}$. For 2-D transformation, $\tau(\cdot)$ is usually defined as affine transformation. Considering six degrees of freedom (6DOF) pose tracking of planar objects, perspective transformation is an appropriate choice.

We formulate the object as a set of $m$ local parts[1] $T^{\mathrm{M}} = \{p_i^{\mathrm{M}} | 1 \leq i \leq m\}$. For simplicity, we abuse the notation $p_i^{\mathrm{M}}$ to denote its location. A popular way to track the individual parts is with *Maximum a Posterior* (MAP) estimation. Alternatively, we view the tracking of individual parts as a matching problem after obtaining the set of parts $T^t = \{p_i^t | 1 \leq i \leq n\}$ from current frame $\mathcal{I}_t$ via part (keypoint) extraction. It is common to represent the correspondence by an assignment matrix $X \in \{0,1\}^{m \times n}$, where $X_{i,j} = 1$ if and only if the $i$-th part $p_i^M$ in the object model corresponds to $j$-th part $p_j^t$ in the current frame. Then, the optimal matching result can be obtained by finding optimal $X^*$ and $\tau^*$ that maximize a score function $\mathcal{E}(X, \tau)$:

$$\begin{aligned} (X^*, \tau^*) &= \arg\max_{X,\tau} \mathcal{E}(X, \tau), \\ \text{s.t.} \quad & X\mathbf{1}_n \leq \mathbf{1}_m, X^T\mathbf{1}_m \leq \mathbf{1}_n, \end{aligned} \tag{1}$$

where $\mathbf{1}_n$ denotes a column vector of $n$ ones. The constrains guarantee that each part can be matched at most once.

1. In this paper, we treat "parts" and "keypoints" interchangeably.

### 3.2 Typical solutions based on unary constraints

To achieve robust and accurate matching, $\mathcal{E}(X,\tau)$ usually encodes unary appearance or geometric consistences between local parts

$$(X^*,\tau^*) = \arg\max_{X,\tau} \mathcal{E}(X,\tau) = \sum_{i,j} X_{i,j} s_\tau(i,j), \quad (2)$$

where $s_\tau(\cdot,\cdot)$ is the similarity function between local parts.

A classic example is the *iterative closest point* (ICP) algorithm [5] that specifies the similarity function as localization distance $s_\tau(i,j) = -\|\tau(p^i_{\mathrm{M}}) - p^j_t\|_2^2$. In general, the joint optimization over $X$ and $\tau$ is non-convex with no known closed-form solution. Typically, the optimization is solved separately by first acquiring the optimal part correspondences $X^*$ using local features and then computing the optimal transformation $\tau^*$ using a robust geometric verification method based on generated correspondences (e.g. RANSAC and its variants [12], [45]).

For unary appearance information, the similarity function is specified to indicate the photometric similarity $s_\tau(i,j) = -\|f^i_{\mathrm{M}} - f^j_t\|_2^2$, where $f^{\mathrm{M}}_i$ and $f^t_a$ are photometric descriptors of the corresponding parts.

Although a tracking result can be obtained by using the unary constraints only, such result is often unreliable because unary information, either appearance or location, lacks structure information and is vulnerable to various geometric and photometric transformations. In this paper, we reformulate the matching between local parts in a graph matching framework. Consequently, our solution produces robust and accurate tracking results by incorporating structure information into pairwise constraints.

## 4 The Proposed Graph-Based Tracker

In this section, we propose a graph-based object tracking framework that integrates the pairwise mutual relation between local parts and views part-based object tracking as graph matching problem. In this framework, the appearance and internal structure of the target can be well integrated. Instead of representing the target as the collection of local parts or star model, we represent the target as an undirected graph. Given the model graph $G^{\mathrm{M}}$ and the candidate graph $G^t$ of frame $t$, our goal is to find the optimal correspondence between them, and to determine the optimal target state based on the correspondence results.

### 4.1 Graph construction

An undirected graph of $n$ vertices can be represented by $\mathbb{G} = (\mathbb{V},\mathbb{E})$, where $\mathbb{V} = \{v_1,\ldots,v_n\}$ and $\mathbb{E} \subseteq \mathbb{V}\times\mathbb{V}$ denote the vertex and edge sets, respectively. A graph is often conveniently represented by a symmetric adjacency matrix $A \in \mathbb{R}^{n\times n}$, such that $A_{i,j} > 0$ if and only if there is an edge between $v_i$ and $v_j$.

Given the initial region $\mathcal{R}$ of the object of interest in the first frame $\mathcal{I}_1$, we construct a model graph $G^{\mathrm{M}}$ for the object as follows.

**Vertex generation.** We extract keypoints from the frame to represent local parts and model them as vertices to build the graph. A classical way is to get the keypoints as local minima/maxima of the DoG images across scales [33], i.e., SIFT. However, the number of such keypoints varies depending on the detectors and frame content. In addition, SIFT may be sensitive to some types of environmental variations, such as illumination changes and motion blurs, and thus harm the tracking accuracy.

In this paper, we adopt a more robust method to extract keypoints. We first compute the SIFT response for each pixel in $\mathcal{R}$. Subsequently, we divide $\mathcal{R}$ evenly into $N$ grids, and then select one keypoint with maximum response from each grid. We model selected keypoints as graph vertices, and compute SIFT descriptors of these keypoints as vertex attributes.

**Edge generation.** There are several popular methods for edge construction, such as the $\varepsilon$-neighborhood graph, the $k$-nearest neighbor graph, and the fully connected graph. The fully connected graph contains substantial structure information, but it costs too much storage space and computational time and is thus not suitable for real-time applications. The $\varepsilon$-neighborhood graph varies depending on the selected parameter $\varepsilon$ and suffers from the scale changes of the object. In this paper, we adopt Delaunay triangulation [28] to build graph edges because it is invariant to translation, scaling and rotation. Although the edges generated by Delaunay triangulation are highly dependent on results of the feature extraction, the constructed graph structures are reasonably stable across frames because we always extract $N$ keypoints from $N$ evenly divided grids.

For each incoming frame $t$, we construct a candidate graph $G^t$ using the same way, and then formulate the matching problem in a graph matching framework.

### 4.2 Geometric graph matching

We first review the problem in a graph matching view to incorporate pairwise consistences between matches.

For graph matching, given the model graph $\mathbb{G}^{\mathrm{M}} = (\mathbb{V}^{\mathrm{M}},\mathbb{E}^{\mathrm{M}})$ and the candidate graph $\mathbb{G}^t = (\mathbb{V}^t,\mathbb{E}^t)$ of size $N$, the problem is to find a vertex correspondence $X \in \{0,1\}^{N\times N}$ between $\mathbb{G}^{\mathrm{M}}$ and $\mathbb{G}^t$ maximize the following global consistency:

$$\mathcal{E}_{\mathrm{G}}(X) = \sum_{i,a} c_{i,a} X_{i,a} + \sum_{i,j,a,b} d_{i,j,a,b} X_{i,a} X_{j,b}, \quad (3)$$

where $c_{i,a}$ measures the consistency between the $i$-th vertex in $\mathbb{G}^{\mathrm{M}}$ and the $a$-th vertex in $\mathbb{G}^t$, and $d_{i,j,a,b}$ the consistency between edge $(i,j)$ in $\mathbb{G}^{\mathrm{M}}$ and edge $(a,b)$ in $\mathbb{G}^t$. The correspondence matrix $X$ denotes matching result, i.e., $X_{i,a} = 1$ if and only if $v_i \in \mathbb{V}^{\mathrm{M}}$ corresponds to $v_a \in \mathbb{V}^t$.

Eq. (3) is often formulated in a pairwise compatibility form

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \mathcal{E}_{\mathrm{G}}(\mathbf{x}) = \mathbf{x}^\top K \mathbf{x}, \quad (4)$$

where $\mathbf{x} \doteq \mathrm{vec}(X) \in \{0,1\}^{N^2}$ is the vectorized version of matrix $X$ and $K \in \mathbb{R}^{N^2\times N^2}$ is the corresponding affinity matrix:

$$K_{\mathrm{ind}(i,a),\mathrm{ind}(j,b)} = \begin{cases} c_{i,a} & \text{if } i=j \text{ and } a=b, \\ d_{i,j,a,b} & \text{if } A^{\mathrm{M}}_{i,j} A^t_{a,b} > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

where $A^{\mathrm{M}}$ and $A^t$ are adjacency matrices of $\mathbb{G}^{\mathrm{M}}$ and $\mathbb{G}^t$ respectively, $(i,a)$ denotes a candidate match from vertex $v^{\mathrm{M}}_i \in \mathbb{V}^{\mathrm{M}}$ to vertex $v^t_a \in \mathbb{V}^t$ for simplicity, and $\mathrm{ind}(\cdot)$ is a bijection that maps a vertex correspondence to an integer index.

For the general graph matching problem, the transformation between two sets of vertices is not considered due to lacks of prior knowledge. As for the object tracking task, we are able to take advantage of transformation information gained from previous frames to guide the matching problem. In this paper, we propose a *geometric graph matching* (GGM) framework to incorporate transformation cues into graph matching manner. Inspired by the *deformable graph matching* (DGM) algorithm presented in [53], the proposed GGM framework combines feature matching and transformation estimation into a single unified framework. The

main differences between DGM and the proposed GGM lie in the different objective functions and subsequent optimizations for both correspondence and transformation.

The score function in Eq. (4) is extended to measure pairwise consistences between matches under transformations as

$$(\mathbf{x}^*, \tau^*) = \arg\max_{\mathbf{x},\tau} \mathcal{E}_{\mathrm{GRA}}(\mathbf{x}, \tau) = \mathbf{x}^\top K(\tau)\mathbf{x}, \tag{6}$$

where $K(\tau) \in \mathbb{R}^{N^2 \times N^2}$ is the pairwise affinity matrix under given transformation $\tau$. To incorporate both photometric and geometric constraints, we define the affinity matrix $K(\tau)$ as

$$K_{\mathrm{ind}(i,a),\mathrm{ind}(j,b)}(\tau) = \begin{cases} \cos(f_i^{\mathrm{M}}, f_a^t) & \text{if } i = j \text{ and } a = b, \\ d_{i,j,a,b}(\tau) & \text{if } A^{\mathrm{M}}_{i,j} A^t_{a,b} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

where $f_i^{\mathrm{M}}$ and $f_a^t$ are photometric descriptors of the vertices, $\cos(\cdot,\cdot)$ is the vector cosine of two vectors, $d_{i,j,a,b}(\tau)$, as defined by (8), is the consistency between edge $(i,j)$ in $\mathbb{G}^{\mathrm{M}}$ and edge $(a,b)$ in $\mathbb{G}^t$ under transformation $\tau$:

$$d_{i,j,a,b}(\tau) = \omega - \|(\tau(p_i^{\mathrm{M}}) - \tau(p_j^{\mathrm{M}})) - (p_a^t - p_b^t)\|_2, \tag{8}$$

in which $\omega$ is chosen to be sufficiently large to ensure that the pairwise affinity is greater than zero. When the two matches $(i,a)$ and $(j,b)$ indicates the same one, the corresponding entry records the photometric similarity between vertices. Otherwise, the entry encodes the geometric consistence between edges.

After optimization for Eq. (6), we obtain the optimal matching $\mathbf{x}^*$ as well as the optimal transformation $\tau^*$. Due to the non-convex nature of the objective function, no closed-form solution of this problem is known. We discuss the optimization of this problem in details in Sec. 5.

### 4.3 Candidate matches filtering

In graph construction, we extract $N$ keypoints as vertices for each graph. There are totally $N^2$ candidate matches between vertices of the model graph $G^{\mathrm{M}}$ and the candidate graph $G^t$. The size of the affinity matrix $K(\tau)$ is, therefore, as large as $N^4$. Although we use Delaunay triangulation to reduce graph edges and make the affinity matrix $K(\tau)$ sparse, the number of total candidate matches and the size of $K(\tau)$ remain huge. Too many entries in the affinity matrix $K(\tau)$ lead to high costs in not only storage space but also computational time.

We propose to further reduce the size of $K(\tau)$ by filtering candidate matches under a reasonable continuity assumption that forbids any leap of matches between consecutive frames. For an incoming frame $t$, we construct a candidate matches set for each vertex $v_i^{\mathrm{M}} \in \mathbb{V}^{\mathrm{M}}$ applying geometric and photometric constaints

$$C_i^t = \{(i,a) \mid \|p_a^t - \tau_{t-1}(p_i^{\mathrm{M}})\|_2 \le \epsilon_{\mathrm{g}}, \cos(f_i^{\mathrm{M}}, f_a^t) \ge \epsilon_{\mathrm{a}}\}, \tag{9}$$

where $\epsilon_{\mathrm{g}}$ and $\epsilon_{\mathrm{a}}$ are tolerances of geometric and appearance changes respectively. The geometric tolerance $\epsilon_{\mathrm{g}}$ is set to be the radius of the object region at the previous frame. The appearance tolerance $\epsilon_{\mathrm{a}}$ is initialized to 0.6, and then adapted per frame according to the appearance similarity between matched vertex pairs as

$$\epsilon_{\mathrm{a}} = (1-\rho)\epsilon_{\mathrm{a}} + \rho \frac{1}{N} \sum_{i,a} \cos(f_i^{\mathrm{M}}, f_a^t) X_{i,a}, \tag{10}$$

where $X$ the final acquired assignment matrix, and $\rho = 0.2$ a pre-defined learning rate.

To reduce computational time of the subsequent procedure of graph matching, we remove redundant matches from $C_i^t$ and remain at most $n_{\mathrm{c}}$ matches with maximum appearance similarity. The final set of candidate matches is constructed by combining the sets for all vertices $C^t = \cup_i C_i^t$. After constructing the candidate match set $C^t$, we condense the affinity matrix $K(\tau)$ by removing the corresponding row and column for each match $(i,a) \notin C^t$. The size of the affinity matrix $K(\tau)$ is thus reduced to $n_{\mathrm{c}}^2 N^2$ at most. We set $n_{\mathrm{c}} = 5$ throughout our experiments.

## 5 Optimization

Since optimizing Eq. 6 jointly over $\mathbf{x}$ and $\tau$ is non-convex, we optimize this geometric graph matching problem by alternatively solving the matching $\mathbf{x}$ and the transformation $\tau$. As illustrated in Fig. 1, our approach falls into the *prediction and refinement* framework. After graph construction and match filtering for the incoming frame, we first predict $\mathbf{x}$ and $\tau$ using the solutions from previous frames, and then refine them by alternatively optimizing one of them while fixing the other one. The refinement is iterated until convergence or maximum iterations reached.

### 5.1 Pose prediction

We need to predict the motion parameters $\tau_{t+1}$ at time $t+1$ after $\tau_t$ has been acquired. A simple way of the prediction is to initialize $\tau_{t+1} = \tau_t$, which is effective for slow motion but it fails in presence of fast motions.

In this paper, we adopt a more precise way to linearize the problem by expanding $\tau_{t+1}$ in a Taylor series

$$\tau_{t+1} = \tau_t + \nabla\tau_t + h.o.t., \tag{11}$$

where $\nabla\tau_t$ denotes the magnitude of $\tau$ at time $t$, and $h.o.t.$ are the high order terms of the expansion that can be neglected. We approximate $\nabla\tau_i = \tau_{i+1} - \tau_i$ for each time $i (1 \le i < t)$, and then estimate $\nabla\tau_t = \frac{1}{k}\sum_{i=1}^{k} \nabla\tau_{t-i}$, where $k$ controls the sliding window for prediction. We set $k = 5$ throughout our experiments.

### 5.2 Graph matching with prediction

Given a fixed transformation function $\tau$, the geometric graph matching problem is equivalent to the traditional graph matching problem. Since graph matching is in nature a combinatorial problem and there is no known efficient algorithm for global optimum, a common way is to search for approximate solutions under relaxed conditions or constraints. There are a large number of literatures dedicated to this problem [16], some examples include [11], [29], [32], [47], [51]. Among these algorithms, the *reweighted random walks for graph matching* (RRWM) algorithm [11] achieves an excellent balance between matching accuracy and computational efficiency. While we choose RRWM as a component in our tracker, other graph matching algorithms may also be used.

**Reweighted Random Walks for Graph Matching (RRWM).** RRWM introduces an association graph constructed with nodes as candidate correspondences and edges as pairwise compatibilities between candidate correspondences. The authors cast the search for correspondences between two given graphs as a node ranking and selection problem in the association graph, and introduce an affinity-preserving random walk algorithm to drive the node ranking based on its quasi-stationary distribution. This algorithm is shown to be closely related to [29] for the *integer quadratic*

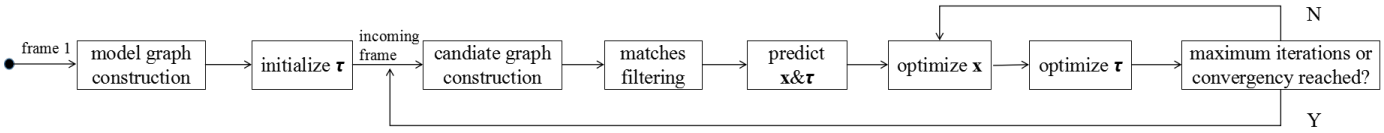



Fig. 1. The framework for the proposed Gracker algorithm.

*programming* (IQP) problem, where one main difference lies that the probability distribution is updated by adopting the jump as

$$\mathbf{x}^{(n+1)} = (1-\alpha)\mathbf{P}\mathbf{x}^{(n)} + \alpha\mathbf{r}, \tag{12}$$

where $\mathbf{x}^{(n)}$ denotes the probability distribution at time $n$ of the association graph, $\mathbf{P}$ its transition matrix, $\mathbf{r}$ the reweighting jump vector toward the matching constraints, and $\alpha$ a pre-defined probability for jumping.

RRWM starts with an initial distribution, uses the jumps at each iteration for generating a biased random walk to the matching constraints, and finally converges at its quasi-stationary distribution. Its fast convergence is observed empirically in practice, which makes it very efficient and suitable for real-time application. More details can be found in [11].

**Matching prediction.** The initial matching is important for RRWM since different initializations may result in total different final solutions. However, the input matching is usually initialized to be trivial in general graph matching tasks due to lacks of prior knowledge. Fortunately, for tracking, we propose to predict matching using the result from the previous frame, and take the prediction as the input of RRWM.

For each incoming frame $t$, denote $\tau_{t-1}$ the final transformation acquired from the previous frame. The candidate match set $C^t$ is constructed as described in Sec. 4.3. For each candidate match $(i,a) \in C^t$, we initialize the corresponding probability as

$$\mathbf{x}_{\text{ind}(i,a)} = 1 - \frac{\|p_a^t - \tau_{t-1}(p_i^{\text{M}})\|_2}{\max_{(j,b)\in C^t} \|p_b^t - \tau_{t-1}(p_j^{\text{M}})\|_2}, \tag{13}$$

where $p_i^{\text{M}}$ and $p_a^t$ denote the location of vertices $v_i^{\text{M}} \in \mathbb{V}^{\text{M}}$ and $v_a^t \in \mathbb{V}^t$ respectively. The intuition of this prediction lies in that model vertices are more likely to match to close locations in consecutive frames. We normalize the prediction $\mathbf{x}$ and take it as the input of RRWM.

### 5.3 Optimization for the transformation

The optimization over $\tau$ given $\mathbf{x}$ is reformulated to

$$\tau^* = \arg\max_{\tau} \Big\{ \sum_{(i,a)\in D} \cos(f_i^{\text{M}}, f_a^t) + \sum_{(i,a),(j,b)\in D} (\omega - \|(\tau(p_i^{\text{M}}) - \tau(p_j^{\text{M}})) - (p_a^t - p_b^t)\|_2) \Big\},$$

where $D = \{(i,a) | \mathbf{x}_{\text{ind}(i,a)} = 1, (i,a) \in C^t\}$ is the set of true correspondence. It can be further reduced as

$$\begin{aligned} \tau^* &= \arg\min_{\tau} \sum_{(i,a),(j,b)\in D} \|(\tau(p_i^{\text{M}}) - \tau(p_j^{\text{M}})) - (p_a^t - p_b^t)\|_2 \\ &= \arg\min_{\tau} \sum_{(i,a),(j,b)\in D} \|\tau(p_i^{\text{M}} - p_j^{\text{M}}) - (p_a^t - p_b^t)\|_2 \end{aligned} \tag{14}$$

For planar object tracking, we always constraint transformation $\tau$ as a linear function, this optimization therefore becomes a linear least squares problem, and can be easily solved by linear fitting.

## 6 Experiments

The experimental results provided in this section consist of two parts. The first one illustrates how the tracking accuracy and computational time of the proposed Gracker algorithm are influenced by the size of graphs. The second one compares Gracker with eight popular tracking algorithms on two public benchmarks and a collected dataset.

### 6.1 Baselines and Benchmarks

In this section, we report experimental results of the proposed Gracker algorithm in comparison with eight state-of-the-art baselines, including Struck [20], IC [3], ESM [35], GOESM [10], GPF [27], TLD [24], KCF [21] and SRDCF [13]. Among these algorithms, the first five ones are planar object trackers, and the last three ones are recently proposed for generic object tracking.

For a thorough evaluation, we report experimental results on the following three datasets:

**UCSB [17]:** The dataset comprises 96 video streams displaying six differently textured planar targets with a total of 6,889 frames, featuring geometric distortions (panning, zoom, tilting, rotation), nine levels of motion blur, as well as different lighting conditions, with all frames affected by natural amounts of noise.

**TMT [41]:** The dataset contains 109 image sequences of manipulation task recorded by a human user and a robot arm. The sequences were grouped under two broad categories: *Oriented Motion Tasks* and *Composite Motion Tasks*. An oriented motion task refers to one or more highly structured geometric transformations, including zoom, tilting, rotation, translation and occlusion. All tasks contain two different light conditions, among them oriented motion tasks are recorded at five different speeds.

**Motion Blur:** This dataset is collected to evaluate tracking performance under heavy blur due to fast motion. It consists of 14 video streams and 3,181 frames showing fast motions of several different planar targets, such as banknote, book and picture.

All videos come with (semi-)manually annotated ground-truth across all frames. The standard overlap criteria of PASCAL VOC [15] is applied for evaluation in the following experiments.

It is worth mentioning that there are standard benchmarks for generic object tracking, e.g. VOT [26] and OTB [49]. However, the targets in these benchmarks are in general non-planar and sometimes even non-rigid. Moreover, the annotations in these benchmarks are axis-aligned bounding boxes. These issues make them inappropriate for evaluating planar object trackers.

### 6.2 Analysis of key parameters and components

The parameter $N$ described in Sec. 4.1 decides the number of extracted keypoints and hence the size of the graphs. It is the most crucial parameter in the proposed Gracker algorithm and directly affects tracking accuracy and computational time. In this section, we report the average tracking accuracy and computational time of Gracker with respect to $N$. As shown in Fig. 2, the tracking

TABLE 1
Average tracking accuracy (±standard deviation) on the UCSB dataset. Bold font indicates the best tracking accuracy (the same style is also used for Tables 2 , 3 and 4).

| Motion task | Struck [20] | IC [3] | ESM [35] | GPF [27] | GOESM [10] | KCF [21] | SRDCF [13] | TLD [24] | Gracker |
|---|---|---|---|---|---|---|---|---|---|
| panning (6) | 0.84±0.25 | 0.29±0.25 | 0.68±0.31 | 0.90±0.02 | 0.35±0.41 | 0.90±0.03 | 0.92±0.03 | 0.79±0.13 | **0.96±0.02** |
| tilting (6) | 0.73±0.41 | 0.82±0.30 | **0.90±0.19** | 0.73±0.28 | **0.90±0.18** | 0.67±0.33 | 0.68±0.33 | 0.62±0.38 | 0.85±0.24 |
| rotation (6) | 0.65±0.33 | 0.74±0.21 | **0.80±0.14** | 0.79±0.12 | 0.79±0.14 | 0.66±0.18 | 0.59±0.24 | 0.65±0.18 | **0.80±0.13** |
| zoom (6) | 0.73±0.34 | 0.73±0.29 | **0.92±0.07** | 0.87±0.07 | 0.88±0.11 | 0.55±0.28 | 0.88±0.06 | 0.77±0.21 | 0.89±0.07 |
| lighting (12) | 0.80±0.33 | 0.68±0.39 | 0.83±0.21 | 0.90±0.02 | **0.99±0.01** | 0.92±0.01 | 0.91±0.02 | 0.58±0.42 | **0.99±0.01** |
| blur (54) | 0.40±0.41 | 0.29±0.37 | 0.44±0.40 | 0.81±0.07 | 0.36±0.43 | 0.82±0.07 | 0.87±0.03 | 0.65±0.33 | **0.88±0.11** |
| unconstrained (6) | 0.36±0.34 | 0.07±0.22 | 0.16±0.24 | 0.42±0.39 | 0.12±0.22 | 0.28±0.18 | **0.66±0.15** | 0.33±0.34 | 0.58±0.37 |
| Total (96) | 0.53±0.41 | 0.41±0.34 | 0.56±0.32 | 0.80±0.10 | 0.52±0.31 | 0.77±0.10 | 0.83±0.07 | 0.64±0.32 | **0.88±0.11** |

TABLE 2
Average tracking accuracy on the TMT dataset.

| object | variation | Struck [20] | IC [3] | ESM [35] | GPF [27] | GOESM [10] | KCF [21] | SRDCF [13] | TLD [24] | Gracker |
|---|---|---|---|---|---|---|---|---|---|---|
| bookI | tilting(12) | 0.77±0.37 | 0.92±0.13 | **0.98±0.02** | 0.86±0.06 | **0.98±0.02** | 0.65±0.28 | 0.74±0.21 | 0.67±0.24 | **0.98±0.02** |
| bookII | zoom (13) | 0.87±0.24 | **0.99±0.01** | **0.99±0.01** | 0.89±0.02 | **0.99±0.01** | 0.70±0.20 | 0.88±0.04 | 0.68±0.18 | **0.99±0.01** |
| bookIII | occlusion (11) | 0.84±0.23 | 0.42±0.46 | 0.56±0.43 | 0.53±0.37 | 0.86±0.19 | 0.80±0.12 | 0.86±0.06 | 0.69±0.21 | **0.93±0.06** |
| cereal | rotation (13) | 0.70±0.44 | 0.66±0.41 | 0.83±0.28 | 0.83±0.03 | 0.44±0.43 | 0.68±0.20 | 0.58±0.29 | 0.66±0.22 | **0.96±0.06** |
| juice | rotation (13) | 0.61±0.43 | 0.62±0.42 | 0.79±0.31 | 0.81±0.06 | 0.48±0.43 | 0.60±0.22 | 0.60±0.22 | 0.59±0.25 | **0.91±0.13** |
| mugI | translation (13) | 0.88±0.13 | 0.85±0.19 | 0.92±0.10 | 0.84±0.05 | **0.93±0.08** | 0.86±0.06 | 0.87±0.07 | 0.72± 0.14 | **0.93±0.08** |
| mugII | tilting (13) | 0.70±0.34 | 0.45±0.47 | 0.61±0.39 | 0.69±0.25 | 0.73±0.29 | 0.72±0.25 | **0.77±0.20** | 0.67±0.20 | 0.74±0.34 |
| mugIII | rotation (13) | 0.75±0.30 | 0.58±0.37 | 0.76±0.29 | 0.79±0.13 | 0.83±0.18 | 0.71±0.23 | 0.76±0.20 | 0.67±0.21 | **0.82±0.22** |
| Composite | unconstrained (8) | 0.48±0.46 | 0.63±0.35 | 0.79±0.23 | 0.63±0.17 | 0.86±0.16 | 0.65±0.14 | 0.62±0.16 | 0.57±0.22 | **0.88±0.16** |
| Total | (109) | 0.74±0.32 | 0.69±0.31 | 0.81±0.23 | 0.77±0.12 | 0.78±0.20 | 0.71±0.19 | 0.74±0.16 | 0.66±0.20 | **0.91±0.12** |

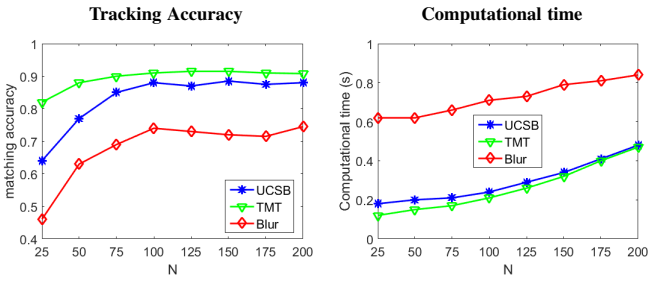



Fig. 2. Average tracking accuracy and computational time of the proposed Gracker algorithm with respect to the size of graphs.

TABLE 3
Comparison on accuracy between Gracker and Gracker$^-$ ($N = 100$).

| Dataset | UCSB | TMT | Motion Blur |
|---|---|---|---|
| Gracker$^-$ | 0.59 | 0.80 | 0.54 |
| Gracker | **0.88** | **0.91** | **0.74** |

accuracy is improved significantly with increasing $N$ when $N$ is smaller than 100, and saturate afterwards. On all the datasets, the computational time is roughly linear in $N$. The algorithm spends more time on the Motion Blur dataset because the objects in this dataset often occupy large image regions, which require much time for keypoint extraction and graph construction.

To validate the influence of the pairwise constraints to the tracking performance, we designed a downgraded version of Gracker, named Gracker$^-$, by removing all pairwise constraints. Gracker$^-$ employs only unary appearance information to generate keypoint correspondences and then utilizes RANSAC to compute the transformation parameters, it is otherwise the same as Gracker. The comparison between Gracker and Gracker$^-$ is summarized in Table 3, which shows clearly the significant improvement brought by the pairwise constraints and the graph-based approach.

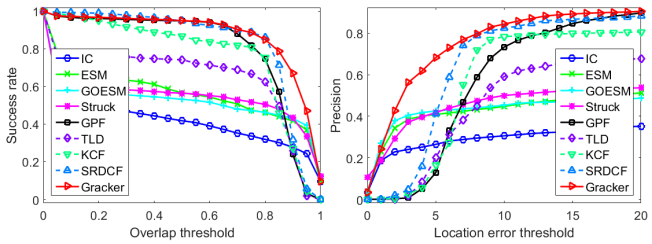
### 6.3 Comparison with existing algorithms

In this section, we report detailed comparison of the proposed Gracker algorithm with eight existing tracking algorithms. We set the parameter $N = 100$ for Gracker.

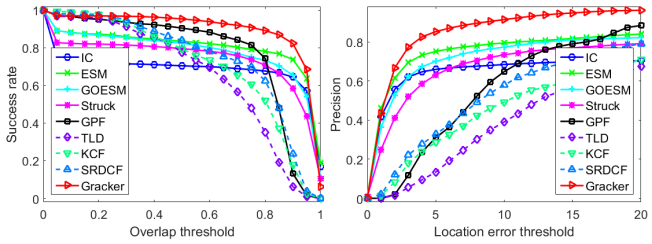
The average tracking accuracy for each video category is reported in Tables 1 (UCSB), 2 (TMT), and 4 (Motion Blur). It is observed that, Gracker achieves the best or nearly best tracking performance in all video categories of the datasets, and exhibits high robustness against not only extreme pose changes but also heavy environmental perturbations. Specifically, Gracker outperforms Struck on all video categories thanks to the help of the graph matching. Some approaches generate comparable results to Gracker on specific categories of videos:

- The template-based planar object trackers, IC and ESM, gain high tracking performance in the case of pose changes including scaling, rotation and tilting, but are sensitive to appearance changes caused by illumination change, occlusion and motion blur;
- GOESM achieves high robustness against illumination change by introducing the gradient orientation feature, but still suffers from motion blur;
- As a probabilistic approach, GPF illustrates high robustness against both illumination change and motion blur, at the cost of relatively low tracking accuracy in presence of occlusion or drastic pose change;
- Designed for generic object tracking, TLD, KCF and SRDCF are able to roughly capture the object in most scenarios, but fail to obtain accurate pose estimation.

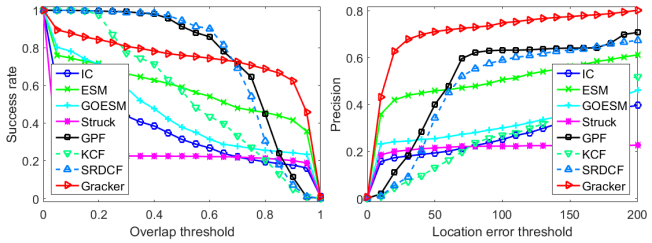
To distinguish between tracking robustness and the accuracy, we provide the success rate curves and precision curves of the compared algorithms in Fig. 3. The curves and associated terms (e.g., success rate) are the same as used in [49]. The figure shows that Gracker achieves both high robustness and excellent accuracy on the three datasets.




(a) the UCSB dataset.




(b) the TMT dataset.




(c) the Motion Blur dataset.

Fig. 3. The success curves (left) and precision curves (right) on (a) the UCSB dataset, (b) the TMT dataset, and (c) the Motion Blur dataset.

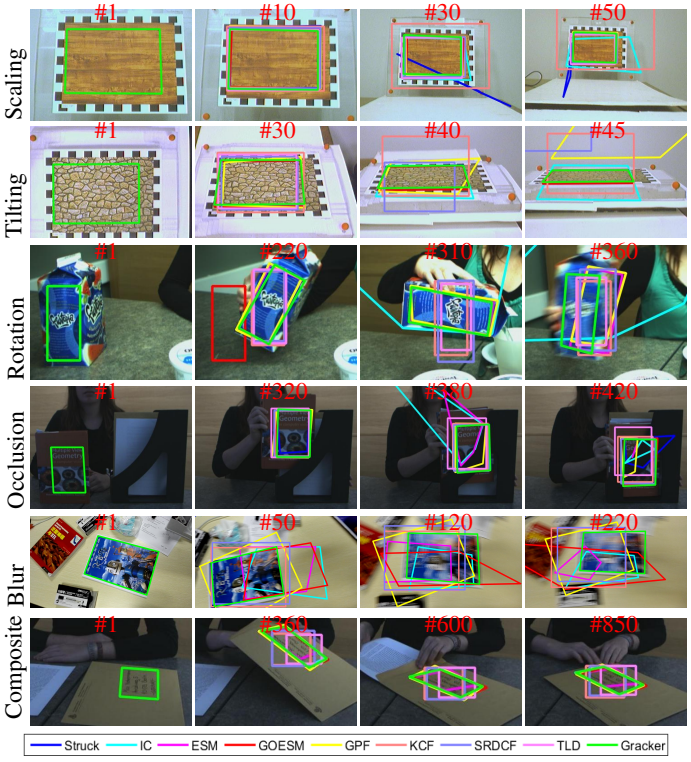



Fig. 4. Examples of tracking results under various transformations. Boxes of different colors indicate tracking results of different algorithms, while absence of boxes of specific colors means the corresponding algorithms lose the object (best viewed in color). Only cropped regions around the object are shown for better illustration.

TABLE 4
Average tracking accuracy (%) on the Motion Blur dataset.

| Algorithm | Struck [20] | IC [3] | ESM [35] | GPF [27] |
|---|---|---|---|---|
| Total | 0.21±0.42 | 0.34±0.37 | 0.57±0.39 | **0.75±0.13** |
| Algorithm | GOESM [10] | KCF [21] | SRDCF [13] | Gracker |
| Total | 0.45±0.35 | 0.55±0.21 | 0.73±0.11 | 0.74±0.33 |

TABLE 5
Average computational time (second) per frame of the algorithms.

| Alg. | Struck [20] | IC [3] | ESM [35] | GPF [27] | GOESM [10] | KCF [21] | SRDCF [13] | TLD [24] | Gracker |
|---|---|---|---|---|---|---|---|---|---|
| UCSB | **0.12** | 0.23 | 0.40 | 0.13 | 2.46 | 0.17 | 0.35 | 0.13 | 0.24 |
| TMT | **0.08** | 0.19 | 0.16 | 0.12 | 2.82 | 0.13 | 0.30 | 0.16 | 0.21 |
| MB | **0.04** | 1.24 | 3.25 | 0.17 | 48.28 | 0.66 | 0.57 | - | 0.70 |

We also report computational time of the algorithms in Table 5. Struck is the most efficient one among these algorithms. Gracker achieves similar computational efficiency with IC and ESM on the UCSB and TMT datasets, but is faster on the Motion Blur dataset. As template-based algorithms, IC, ESM and GOESM suffer from the large size of the object in the Motion Blur dataset. Keypoint-based algorithms and probabilistic algorithms, by contrast, are relatively less affected by the size of the target.

Figure 4 illustrates some representative examples of various types of transformations provided by the proposed Gracker algorithm in comparison with other algorithms.

**Scaling.** The first row shows examples of scale change of a wood picture with very weak texture. Struck, IC, and KCF lose the target where other algorithms provide more accurate results.

**Tilting.** Examples of tilting of a picture with repeated patterns shown in the second row reveal that IC, ESM, GOESM and Gracker are very robust to tilting while other trackers fail to catch the object in presence of extreme tilting transformation.

**Translation & Rotation.** As shown in the third row, planar object trackers are able to handle rotation of the object, but only ESM, GPF and Gracker are robust to rotation of large angles. ESM and GPF become inaccurate after frame 360 because of motion blur, while Gracker provides more stable results across all frames. The generic object trackers, KCF, SRDCF and TLD, however, fail to get an accurate pose estimation in presence of rotation.

**Occlusion.** The fourth row presents tracking results in presence of partial occlusion under dark lighting condition. Based on comparing raw intensities in templates, IC, ESM and GPF suffer from partial occlusion. Struck obtains relative low accuracy due to the lack of reliable keypoints in dark lighting. With the assistance of an online-learned detector, TLD roughly captures the object but with a remarkable drift. The proposed Gracker algorithm, as well as GOESM, KCF and SRDCF, provides more accurate tracking results under partial occlusion.

**Motion blur.** In the fifth row, Struck, IC and GOESM are very sensitive to motion blur, and lose the target from the very beginning to the end. ESM, KCF, SRDCF and GPF roughly catch the object in most frames, but its location is not accurate. In contrast, the proposed Gracker algorithm provides more accurate results across all frames.

**Composite transformation.** The last row shows examples of composite transformations of translation, rotation, tilting and slight non-linear transformation. IC and ESM lose the target from the very beginning to the end. Different to TLD, KCF and SRDCF that can only roughly catch the object, Gracker, GOESM and GPF

provide more accurate tracking results in comparison.

## 7 Conclusion

In this paper, we proposed a novel graph-based tracker, named Gracker, for planar object tracking aiming to improve the tracking performance. Gracker models planar objects as undirected graphs and formulates tracking as a sequential geometric graph matching problem, and it uses a matching prediction strategy to guide graph matching. Experimental results reveal that, Gracker gains accurate and robust tracking performance against various environmental variations, and outperforms recent state-of-the-art algorithms.

## Acknowledgments

This work is supported by China National Key Research and Development Plan (Grant No. 2016YFB1001200), the National Nature Science Foundation of China (nos. 61673048, 61672088, 61671048, 61472028, 61528204 and 61502026), the Fundamental Research Funds for the Central universities (2017JBZ108), and a Research Grant from HiScene Information Technologies.

## References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *CVPR*, pages 798–805, 2006. 2

[2] B. Babenko, M.-H. Yang, and S. J. Belongie. Robust object tracking with online multiple instance learning. *PAMI*, 33(8):1619–1632, 2011. 1, 2

[3] S. Baker and I. Matthews. Lucas-kanade 20 years on: A unifying framework. *IJCV*, 56(3):221–255, 2004. 1, 5, 6, 7

[4] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Speeded-up robust features (surf). *CVIU*, 110(3):346–359, 2008. 1

[5] P. J. Besl and N. D. McKay. A method for registration of 3-d shape. *PAMI*, 14(2):239–256, 1992. 2, 3

[6] W. Bouachir and G.-A. Bilodeau. Structure-aware keypoint tracking for partial occlusion handling. In *WACV*, pages 877–884, 2014. 1, 2

[7] Z. Cai, L. Wen, Z. Lei, N. Vasconcelos, and S. Z. Li. Robust deformable and occluded object tracking with dynamic graph. *IEEE Trans. Image Processing*, 23(12):5497–5509, 2014. 2

[8] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *ECCV*, pages 778–792, 2010. 1

[9] L. Cehovin, M. Kristan, and A. Leonardis. Robust visual tracking using an adaptive coupled-layer visual model. *PAMI*, 35(4):941–953, 2013. 2

[10] L. Chen, F. Zhou, Y. Shen, X. Tian, H. Ling, and Y. Chen. Illumination insensitive efficient second-order minimization for planar object tracking. In *ICRA*, pages 751–757, 2017. 5, 6, 7

[11] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. In *ECCV*, pages 492–505, 2010. 4, 5

[12] O. Chum and J. Matas. Matching with PROSAC - progressive sample consensus. In *CVPR*, pages 220–226, 2005. 2, 3

[13] M. Danelljan, G. Häger, F. S. Khan, and M. Felsberg. Learning spatially regularized correlation filters for visual tracking. In *ICCV*, pages 4310–4318, 2015. 2, 5, 6, 7

[14] N. Dowson and R. Bowden. Mutual information for lucas-kanade tracking (milk): An inverse compositional formulation. *PAMI*, 30(1):180–185, 2008. 1

[15] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (VOC) challenge. *IJCV*, 88(2):303–338, 2010. 5

[16] P. Foggia, G. Percannella, and M. Vento. Graph matching and learning in pattern recognition in the last 10 years. *International Journal of Pattern Recognition and Artificial Intelligence*, 28(1):1–40, 2014. 4

[17] S. Gauglitz, T. Höllerer, and M. Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *IJCV*, 94(3):335–360, 2011. 1, 5

[18] H. Grabner and H. Bischof. Online boosting and vision. In *CVPR*, pages 260–267, 2006. 1, 2

[19] M. Grabner, H. Grabner, and H. Bischof. Learning features for tracking. In *CVPR*, pages 1–8, 2007. 2

[20] S. Hare, A. Saffari, and P. H. S. Torr. Efficient online structured output learning for keypoing-based object tracking. In *CVPR*, pages 1894–1901, 2012. 1, 2, 5, 6, 7

[21] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista. High-speed tracking with kernelized correlation filters. *PAMI*, 37(3):583–596, 2015. 2, 5, 6, 7

[22] S. Holzer, M. Pollefeys, S. llic, D. J. Tan, and N. Navab. Online learning of linear predictors for real-time tracking. In *ECCV*, pages 470–483, 2012. 1

[23] G. Hua and Y. Wu. Measurement integration under inconsistency for robust tracking. In *CVPR*, pages 650–657, 2006. 2

[24] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *PAMI*, 34(7):1409–1422, 2012. 1, 5, 6, 7

[25] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proc. of 2nd Int. Workshop on Augmented Reality*, pages 85–94, 1999. 2

[26] M. Kristan, J. Matas, A. Leonardis, M. Felsberg, L. Cehovin, G. Fernández, T. Vojír, G. Häger, G. Nebehay, and R. P. Pflugfelder. The visual object tracking VOT2015 challenge results. In *ICCV Workshops*, pages 564–586, 2015. 5

[27] J. Kwon, H. S. Lee, F. C. Park, and K. M. Lee. A geometric particle filter for template-based visual tracking. *PAMI*, 36(4):625–643, 2014. 2, 5, 6, 7

[28] D.-T. Lee and B. J. Schachter. Two algorithms for constructing a delaunay triangulation. *Int. J. Computer Information Sci*, 9:219–242, 1980. 3

[29] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *ICCV*, pages 1482–1489, 2005. 4

[30] V. Lepetit and P. Fua. Keypoint recognition using randomized trees. *PAMI*, 28(9):1465–1479, 2006. 2

[31] S. Leutenegger, M. Chli, and R. Siegwart. Brisk: Binary robust invariant scalable keypoints. In *ICCV*, pages 2548–2555, 2011. 1, 2

[32] Z. Liu and H. Qiao. GNCCP - graduated nonconvexity and concavity procedure. *PAMI*, 36(6):1258–1267, 2014. 4

[33] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004. 1, 2, 3

[34] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, pages 674–679, 1981. 1, 2

[35] E. Malis. Improving vision-based control using efficient second-order minimization techniques. In *ICRA*, pages 1843–1848, 2004. 1, 5, 6, 7

[36] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *PAMI*, 27(10):1615–1630, 2005. 1

[37] G. Nebehay and R. P. Pflugfelder. Consensus-based matching and tracking of keypoints for object tracking. In *WACV*, pages 862–869, 2014. 1, 2

[38] S. M. Nejhum Shahed, H. Ho, and M.-H. Yang. Visual tracking with histograms and articulating blocks. In *CVPR*, pages 1–8, 2008. 2

[39] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *PAMI*, 32(3):448–461, 2010. 2

[40] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *PAMI*, 32(1):105–119, 2010. 2

[41] A. Roy, X. Zhang, N. Wolleb, C. P. Quintero, and M. Jägersand. Tracking benchmark and evaluation for manipulation tasks. In *ICRA*, pages 2448–2453, 2015. 1, 5

[42] G. Takacs, V. Chandrasekhar, S. Tsai, D. Chen, R. Grzeszczuk, and B. Girod. Unified real-time tracking and recognition with rotation-invariant fast features. In *CVPR*, pages 934–941, 2010. 2

[43] D. J. Tan and S. Ilic. Multi-forest tracker: A chameleon in tracking. In *CVPR*, pages 1202–1209, 2014. 2

[44] F. Tang and H. Tao. Probabilistic object tracking with dynamic attributed relational feature graph. *IEEE Trans. Circuits Syst. Video Techn.*, 18(8):1064–1074, 2008. 2

[45] P. H. S. Torr and A. Zisserman. Mlesac: A new robust estimator with application to estimating image geometry. *CVIU*, 78(1):138–156, 2000. 2, 3

[46] D. Wagner, G. Reitmayr, A. Mulloni, T. Drummond, and D. Schmalstieg. Real-time detection and tracking for augmented reality on mobile phones. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):355–368, 2010. 2

[47] T. Wang and H. Ling. Path following with adaptive path estimation for graph matching. In *AAAI*, pages 3625–3631, 2016. 4

[48] L. Wen, Z. Cai, Z. Lei, D. Yi, and S. Z. Li. Online spatio-temporal structural context learning for visual tracking. In *ECCV*, pages 716–729, 2012. 1, 2

[49] Y. Wu, J. Lim, and M. Yang. Online object tracking: A benchmark. In *CVPR*, pages 2411–2418, 2013. 5, 6

[50] M. Yang, Y. Wu, and G. Hua. Context-aware visual tracking. *PAMI*, 31(7):1195–1209, 2009. 1, 2

[51] M. Zaslavskiy, F. Bach, and J. Vert. A path following algorithm for the graph matching problem. *PAMI*, 31(12):2227–2242, 2009. 4

[52] L. Zhang and L. van der Maaten. Structure preserving object tracking. In *CVPR*, pages 1838–1845, 2013. 2

[53] F. Zhou and F. De la Torre. Deformable graph matching. In *CVPR*, pages 2922–2929, 2013. 3