Consensus-based Matching and Tracking of Keypoints for Object Tracking

Georg Nebehay Roman Pflugfelder
Safety and Security Department
AIT Austrian Institute of Technology
1220 Vienna, Austria
{georg.nebehay.fl,roman.pflugfelder}@ait.ac.at

Abstract

We propose a novel keypoint-based method for long-term model-free object tracking in a combined matching-and-tracking framework. In order to localise the object in every frame, each keypoint casts votes for the object center. As erroneous keypoints are hard to avoid, we employ a novel consensus-based scheme for outlier detection in the voting behaviour. To make this approach computationally feasible, we propose not to employ an accumulator space for votes, but rather to cluster votes directly in the image space. By transforming votes based on the current keypoint constellation, we account for changes of the object in scale and rotation. In contrast to competing approaches, we refrain from updating the appearance information, thus avoiding the danger of making errors. The use of fast keypoint detectors and binary descriptors allows for our implementation to run in real-time. We demonstrate experimentally on a diverse dataset that is as large as 60 sequences that our method outperforms the state-of-the-art when high accuracy is required and visualise these results by employing a variant of success plots.

1. Introduction

The tracking of a priori unknown objects has drawn considerable interest and has established its place in the tracking community under the name model-free object tracking. Typically, the only information that is presented to the tracker is the initialising region in the first frame of the image sequence and the task of the tracker is to come up with an estimate of the current location of the object of interest. The main advantage of model-free methods is their applicability in various scenarios without requiring any domain-specific knowledge or training. An important property of tracking algorithms is the ability to handle the complete disappearance of the object for an undefined amount of time. Trackers that obviate the need for external re-initialisation are said to address the long-term object tracking problem.

While there has been progress in making model-free object tracking methods more robust, the problem itself is essentially unsolved due to challenges stemming from partial and full occlusions, clutter, and various types of appearance changes caused by changes in object or camera pose and local or global illumination [20]. As the object of interest is unknown beforehand, it is impossible to employ offline machine learning techniques to account for the variability of the object appearance introduced by these challenges. Instead, online learning algorithms have been employed [7, 2, 25] to adapt the object model to changes in the appearance of the object. In practice however, updating a model often introduces errors, as there are no hard class labels available.

In this work we argue that the the choice of the object representation plays an important role in order to be able to overcome these challenges. For instance, it is difficult to express a local change in appearance by a global model, such as a histogram. Models that decompose the object into parts [1, 27, 13, 30] are more robust to the aforementioned nuisances, as local changes only affect individual parts. Even when individual parts are lost or in an erroneous state, other object parts can compensate for them. Many approaches employ rectangular axis-aligned regions for extracting the features of the object and its parts. While computationally convenient, this method has the shortcoming of being non-invariant against changes in scale and rotation. Instead, keypoint detectors such as SIFT [18] and BRISK [16] estimate these properties out of the image data, making keypoints an ideal representation for a parts-pased object model. Recently, the advent of extremely fast keypoint detectors and binary descriptors [24, 16] has dramatically decreased the computational burden of detecting and matching corresponding keypoints, allowing for the use of keypoints in a real-time tracking system. Virtually all state-of-the art approaches update the appearance information during tracking. There is evidence that the combination of static model elements and adaptive elements improve the robustness of tracking algorithms [14, 26]. We propose to decouple the static and the dynamic model elements in a radical fashion by basing the appearance model on the first frame only and by addressing changes in appearance by employing an adaptive tracking technique.

The major contribution of this work is the formulation of the novel tracking approach CMT that employs a keypoint-based object representation. Corresponding keypoints in each frame are found by a combined matching-and-tracking approach. The second contribution is a novel method for identifying keypoints that are in consensus based on a voting mechanism that takes into account the current geometric constellation of the keypoints in order to scale and rotate votes accordingly. Outlier keypoints are identified and removed by clustering the votes. These estimates are also used to identify the pose of the object of interest. The third contribution is a visualisation of the tracking performance of competing trackers in success plots, enabling the quick assessment of tracker performance over a large number of sequences for different requirements on per-sequence performance.

2. Related Work

There has been a considerable amount of work dedicated to parts-based object representations in model-free tracking. In what can be considered a very basic form of a parts-based model, Adam et al. [1] represent the object by multiple image patches (fragments). The patches are arranged in a predefined grid, and each patch votes for the position of the object of interest in a sliding-window framework. The rigid arrangement of the patches makes it impossible to allow for rotations or articulations of the object. In [27] these restrictions are addressed by updating the position of the patches in each frame. Hua et al. [13] propose a part-based approach based on the theory of Markov networks. Instead of blindly fusing all measurements into a single one, they examine the inconsistency of parts to make the approach robust to occlusion, clutter and appearance changes. Zhang et al. [30] employ a connected model to introduce deformation costs into an optimisation problem for multi-target tracking, but suggest that such a model is also applicable to parts-based single-target tracking.

While there is an abundance of work on keypoint-based pose estimation, virtually all of these approaches rely on a pre-learned database of descriptors to be available. There also are some methods that create databases on-the-fly [23, 22], True model-free approaches include the work of Grabner et al. [9], who propose an approach where keypoint matching is performed using a boosting classifier. When a plausible set of matches is found, the newly discovered appearances of the keypoints are interpreted as training examples and used to update the classifier. In a similar fashion, Hare et al. [11] attach a weight to each keypoint and update these weights in a unifying structured output learning framework that spans over both the matching stage and the robust computation of a transformation matrix. Both of these approaches compute homographies between the original set of keypoints and the keypoints in the current by employing robust statistical methods such as RANSAC. However, the estimation of homographies gives poor results for non-planar objects. In Figure 1 we show an example where the estimation of the homography gives distorted results, even though the keypoint association was performed correctly. Maresca et al. [21] employ multiple techniques for keypoint detection and description in a fallback framework and employ the Generalised Hough Transform (GHT [3]) for detecting outliers.

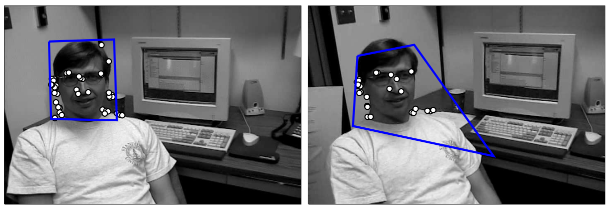
Figure 1: The estimation of homographies is error-prone when planarity assumptions do not hold.

In the GHT, votes are collected in an accumulator space and maxima in this space constitute detections. Voting schemes have been used extensively in the context of shape recognition. A similar mechanism has been presented by Leibe et al. [15] for object categorisation and segmentation, where voting is performed by keypoints according to a learned codebook. Gall et al. [5] train class-specific Hough forests that directly map image patch appearance into a probabilistic vote. Godec et al. [6] adapt this technique to the online learning scenario and compute an object segmentation in order to update the Hough forests and use this technique for the tracking of non-rigid objects. In [4] this work is extended to employ pixel-based descriptors, mainly decreasing the computational cost.

3. Approach

Given a sequence of images $I_1, \dots, I_n$, and an initialising region b_1 in I_1, our aim in each frame of the sequence is to recover the pose of the object of interest or to indicate that the object is not visible. We estimate the object pose up to its center μ, its scale s and the degree of its in-plane rotation α, where s and α are estimated with respect to the initial appearance of the object. For simplicity, we restrict ourselves to axis-aligned rectangular initialising regions. The remainder of this section describes our approach in detail, while an algorithmic formulation is given in Algorithm 1. In the following, we assume that a method for keypoint detection and description is available.

Algorithm 1 CMT

```
Input: $I_1, \dots, I_n, b_1$
Output: $b_2, \dots, b_n$
 1: $O \leftarrow$ detect$(I_1, b_1)$
 2: $K_1 \leftarrow O$
 3: for $t \leftarrow 2, \dots, n$ do
 4:     $P \leftarrow$ detect$(I_t)$
 5:     $M \leftarrow$ match$(P, O)$
 6:     $T \leftarrow$ track$(K_{t-1}, I_{t-1}, I_t)$
 7:     $K' \leftarrow T \cup M$
 8:     $s \leftarrow$ estimate_scale$(K', O)$
 9:     $\alpha \leftarrow$ estimate_rotation$(K', O)$
10:     $V \leftarrow$ vote$(K', O, s, \alpha)$
11:     $V^c \leftarrow$ consensus$(V)$
12:     $K_t \leftarrow$ vote$^{-1}(V^c)$
13:     if $|V^c| \geq \theta \cdot N^O$ then
14:         $\mu \leftarrow \frac{1}{n} \sum_{i=1}^n V_i^c$
15:         $b_t \leftarrow$ bounding_box$(b_1, \mu, s, \alpha)$
16:     else
17:         $b_t \leftarrow \emptyset$.
18:     end if
19: end for
```

3.1. Matching and Tracking of Keypoints

We base our object model on a set of keypoints

$$O = \{(r_i, f_i)\}_{i=1}^{N^O}, \tag{1}$$

where each keypoint denotes a location $r \in \mathbb{R}^2$ in template coordinates and a descriptor f. We employ binary descriptors $f \in \{0,1\}^d$ for computational reasons, but the presented ideas are applicable to real-valued descriptors as well. We initialise O by detecting and describing keypoints in I_1 that are inside the initialising region b_1, followed by a mean-normalisation of the keypoint locations. In order to recover the object pose, in each I_t with $t \geq 2$ we are interested in finding a set of corresponding keypoints

$$K_t = \{(a_i, m_i)\}_{i=1}^{N^{K_t}}, \tag{2}$$

where a refers to the keypoint position in absolute image coordinates and m is the index of the corresponding keypoint in O. By both matching and tracking keypoints, we follow two complementary strategies for finding K_t.

We detect and describe candidate keypoints

$$P = \{(a_i, f_i)\}_{i=1}^{N^P}, \tag{3}$$

in I_t that are determined by their absolute position a and their descriptor f. For each candidate keypoint, we compute the Hamming distance

$$d(f^1, f^2) = \sum_{i=1}^{d} \text{XOR}(f_i^1, f_i^2) \tag{4}$$

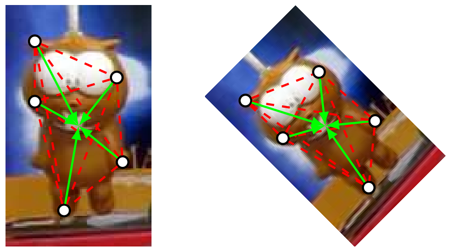

Figure 2: Initialisation of voting vectors occurs in the first frame only. Votes are scaled and rotated according to the current keypoint constellation.

of its descriptor to the descriptors of all keypoints found in I_1, including background keypoints. We match candidate keypoints in P to keypoints in I_1 by requiring that the nearest neighbour must be closer than the second-nearest neighbour by a certain ratio ρ. The set of matched keypoints M then consists of the subset of keypoint locations in P that match to O, augmented (in analogy to Eq. 2) with the corresponding model keypoint index. Candidate keypoints that match to background keypoints are excluded from M.

For tracking, we compute the displacement of each keypoint in K_{t-1} from I_{t-1} to I_t by employing the pyramidal variant of the method of Lucas and Kanade for estimating optical flow [19]. For $t = 2$, K_1 is obtained by transforming O to absolute image coordinates. The set of tracked keypoints T is then obtained by updating the keypoint locations in K_{t-1} while maintaining the keypoint index. Keypoints that fail to be tracked or that end up outside the image boundaries are removed from T.

We fuse T and M into a set K' of size $N^{K'}$, discarding all tracked keypoints when there exists a matched keypoint associated with the same model keypoint. Intuitively, matched keypoints are more robust as they do not rely on a recursive estimation. Typically, K' still contains outliers as there is some intrinsic ambiguity in the process of matching and tracking keypoints.

3.2. Voting

In order to locate the object of interest, each keypoint (a, m) in K' casts a single vote $h(a, m) \rightarrow \mathbb{R}^2$ for the object center, resulting in a set of votes

$$V = \{h(a_i, m_i)\}_{i=1}^{N^{K'}}, \tag{5}$$

as shown in Figure 2. In its simplest form, we consider only translational changes of the object

$$h^T(a, m) = a - r_m, \tag{6}$$

where r_m is the relative position of the corresponding keypoint in O. When the scale of the object changes, votes

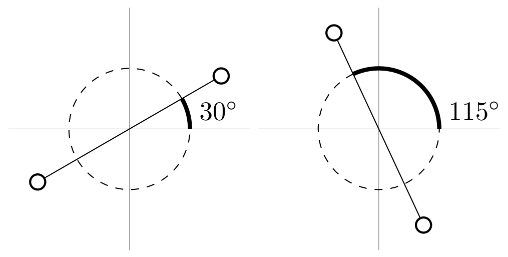

Figure 3: The pairwise angular change between keypoints is used to estimate the object rotation. Here, the angular change between the initial constellation of two keypoints and their constellation at a later stage is 85°.

either overshoot the object center or fall short. We overcome this limitation by scaling the votes by a single scale factor s. Eq. 6 therefore becomes

$$h^S(a, m) = a - s \cdot r_m. \tag{7}$$

In order to compute s, we rely on the pairwise Euclidean distance between a_i and a_j in K' and compare these distances to the distances of the corresponding keypoints r_{m_i} and r_{m_j} in O. Let $a^{i,j} = a_i - a_j$ and $r^{i,j} = r_{m_i} - r_{m_j}$, then the distribution of all individual changes in scale is

$$D_s = \left\{ \frac{\|a^{i,j}\|}{\|r^{i,j}\|}, i \neq j \right\}. \tag{8}$$

The median of this distribution $s = \text{med}\,(D_s)$ is a suitable estimate for the scale as it is robust against outliers.

When the object undergoes an in-plane rotation, votes have to be rotated accordingly in order to still target the object center. We do so by updating Eq. 7 to become

$$h^R(a, m) = a - s \cdot R r_m, \tag{9}$$

where R is the 2D rotation matrix

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}. \tag{10}$$

We estimate the rotation α by analysing the pairwise angular change between keypoints with respect to their initial constellation, as depicted in Figure 3, by employing the function atan2[1]

$$\alpha_{i,j} = \text{atan2}(a_y^{i,j}, a_x^{i,j}) - \text{atan2}(r_y^{i,j}, r_x^{i,j}). \tag{11}$$

We obtain a robust estimate for the rotation of the object from the distribution of all pairwise angular changes

$$D_\alpha = \{\alpha_{i,j}, i \neq j\}. \tag{12}$$

by computing its median $\alpha = \text{med}\,(D_\alpha)$. We refrain from using the information about scale and rotation that is available through most keypoint detectors, as we have found it not to be reliable enough.

[1]The function atan2(y, x) computes the arctangent of its argument while taking into account the appropriate quadrant.

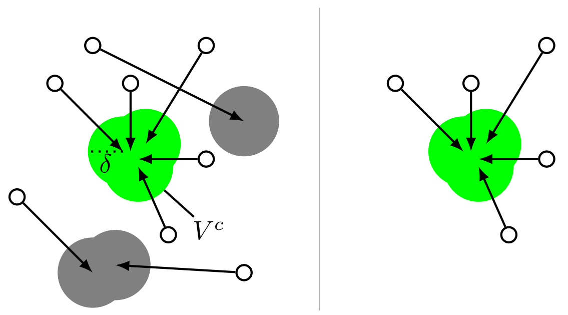

Figure 4: Finding consensus in voting behaviour. Left: Cast votes are clustered based on their Euclidean distance and a cutoff threshold δ. The consensus cluster V^c is identified based on the highest number of votes. Right: Keypoints that voted into the consensus cluster are kept, all other keypoints are removed.

3.3. Consensus

Whenever either the location a or the model index m of an entry in K' is wrong, votes will not target the object center, but rather point to arbitrary image locations. Before calculating the object center μ, we identify and remove outlier keypoints and their votes by looking for consensus in the voting behaviour as depicted in Figure 4. To this end, we apply hierarchical agglomerative clustering [29] on V based on the Euclidean distance as a dissimilarity measure. In this type of clustering, data is organised into hierarchical structures according to a proximity matrix, resulting in a dendrogram that is then cut off at a certain threshold δ. Thus, V is partitioned into disjoint subsets $V^1, \dots, V^m$. We consider the subset containing the largest number of elements to be the consensus cluster V^c and set K_t to the subset of K' that voted into V^c. Hierarchical clustering is a relatively expensive operation, as it is in $O(N^2)$, but as V typically does not exceed 200 elements, in our case it is rather cheap. A central advantage of our method is that it does not make any assumptions about the planarity of objects. Instead, keypoints are allowed to drift slightly from their original positions. The degree of allowed flexibility is steered by the parameter δ.

If V^c contains less than $\theta \cdot |O|$ elements, we assume the object is not visible. Otherwise, we turn the votes in the consensus cluster into an estimate for the object center

$$\mu = \frac{1}{n} \sum_{i=1}^{n} V_i^c, \tag{13}$$

where $n = |V^c|$. The object center μ, together with the scale s and the rotation α defines the pose of the object of interest. As the final output, we compute a non-axis-aligned bounding box by transforming the four corners $c_1, \dots, c_4$ of

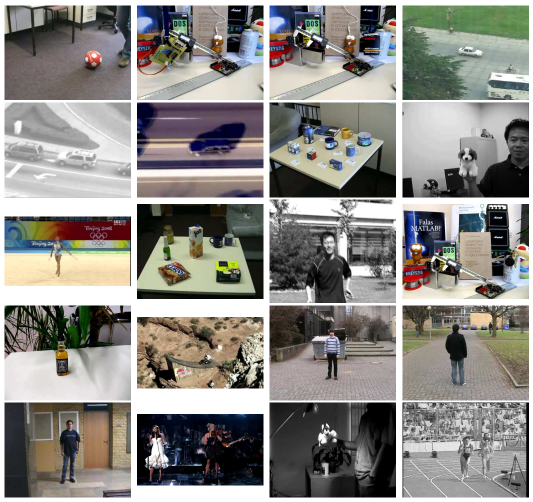

Figure 5: Sequences used for quantitatively assessing tracker performance. From left to right, top to bottom: *ball*, *board*, *box*, *car*, *car 2*, *carchase*, *cup on table*, *dog1*, *gym*, *juice*, *jumping*, *lemming*, *liquor*, *mountain-bike*, *person*, *person crossing*, *person partially occluded*, *singer*, *sylvester*, *track running*.

b_1 by

$$c'_i = \mu + s \cdot Rc_i, \tag{14}$$

where R is the rotation matrix from Eq. 10. The parameter $\theta \in [0, 1]$ influences the receiver operating characteristics, as uncertain results are suppressed for higher values of θ.

4. Experiments

Our main claim is that our proposed method performs accurately under changes in translation, scale and rotation different object classes, while being robust to partial and full occlusions as well as to a variety of changes in appearance. In order to support this claim, we collected 20 sequences that were used previously for evaluating tracking methods. As it can be seen in Figure 5, the sequences encompass a wide range of different objects and scenarios. In our implementation[2], we employ the same parameter values for all sequences during the quantitative evaluation. For the detection and description of keypoints, we employ BRISK [16] with a dimensionality $d = 512$. For matching candidate keypoints to the model, we follow [18] and set the ratio threshold $\rho = 0.8$. We experimented with different values of the cut-off threshold and found $\delta = 20$ to give good results. We require that at least 10% of the initial keypoints must be found ($\theta = 0.1$). In Figure 6 we present qualitative results on selected sequences, where the first image of each row shows the first frame of the sequence. Inlier and outlier keypoints are shown in white and red, respectively.

[2]Available at: http://www.gnebehay.com/cmt

There are a range of measures available throughout the literature for assessing the performance of tracking algorithms quantitatively. Many authors employ the center-error measure that expresses the distance between the centroid of the algorithmic output and the centroid of the ground truth. This measure is only a rough assessment of the localisation as it completely ignores the scale and the aspect ratio of the bounding boxes. Furthermore it is not bounded, making the comparison of results obtained on different sequences difficult. Instead, we employ the widely used overlap measure

$$o(b_T, b_{GT}) = \frac{b_T \cap b_{GT}}{b_T \cup b_{GT}}, \tag{15}$$

where b_T is the tracker output and b_{GT} refers to the manually annotated bounding box. This measure has been shown to penalise translation and scale alterations equally [12] and therefore is a better indicator for per-frame success. Furthermore it is bounded between 0 and 1. As this measure is defined on axis-aligned bounding boxes, we align all non-aligned bounding boxes by fitting the smallest possible bounding box around them.

In order to convert the per-frame measures into an overall score for a sequence, one possibility would be to compute their mean. However, an average overlap of 0.5 does not reveal whether a method was completely accurate for half of the sequence or whether it was not very accurate for the whole sequence. We therefore employ a threshold τ on Eq. 15, thus converting frames into true positives (TP) and false negatives (FN). The threshold τ steers the requirement on per-frame accuracy. We employ 0.25, 0.5 and 0.75 as values for τ, which we interpret as low, medium and high requirements on accuracy, providing a categorisation for potential applications. For even lower values, the localisation becomes increasingly rough and higher thresholds are not reasonable as there is some intrinsic ambiguity in the process of manual annotation that cannot be eliminated [17]. We compute

$$recall = \frac{TP}{TP + FN} \tag{16}$$

as the performance measure for a sequence. This measure (sometimes referred to as percentage of correctly tracked frames) indicates how many frames the tracker output satisfies the requirement on the overlap τ when the object was visible.

We compare our approach quantitatively to the state-of-the-art tracking approaches STRUCK (Structured output Tracking [10]), TLD (Tracking-Learning-Detection[14]), LM (LearnMatch [11]), FT (Fragments-based Tracking [1]), HT (HoughTrack [6]) and SB (Semi-supervised online Boosting [8]). We obtained the source code for these trackers from the respective project websites and left all parameter settings at their default values. The algorithms were initialised using the first bounding box of the annotation. In

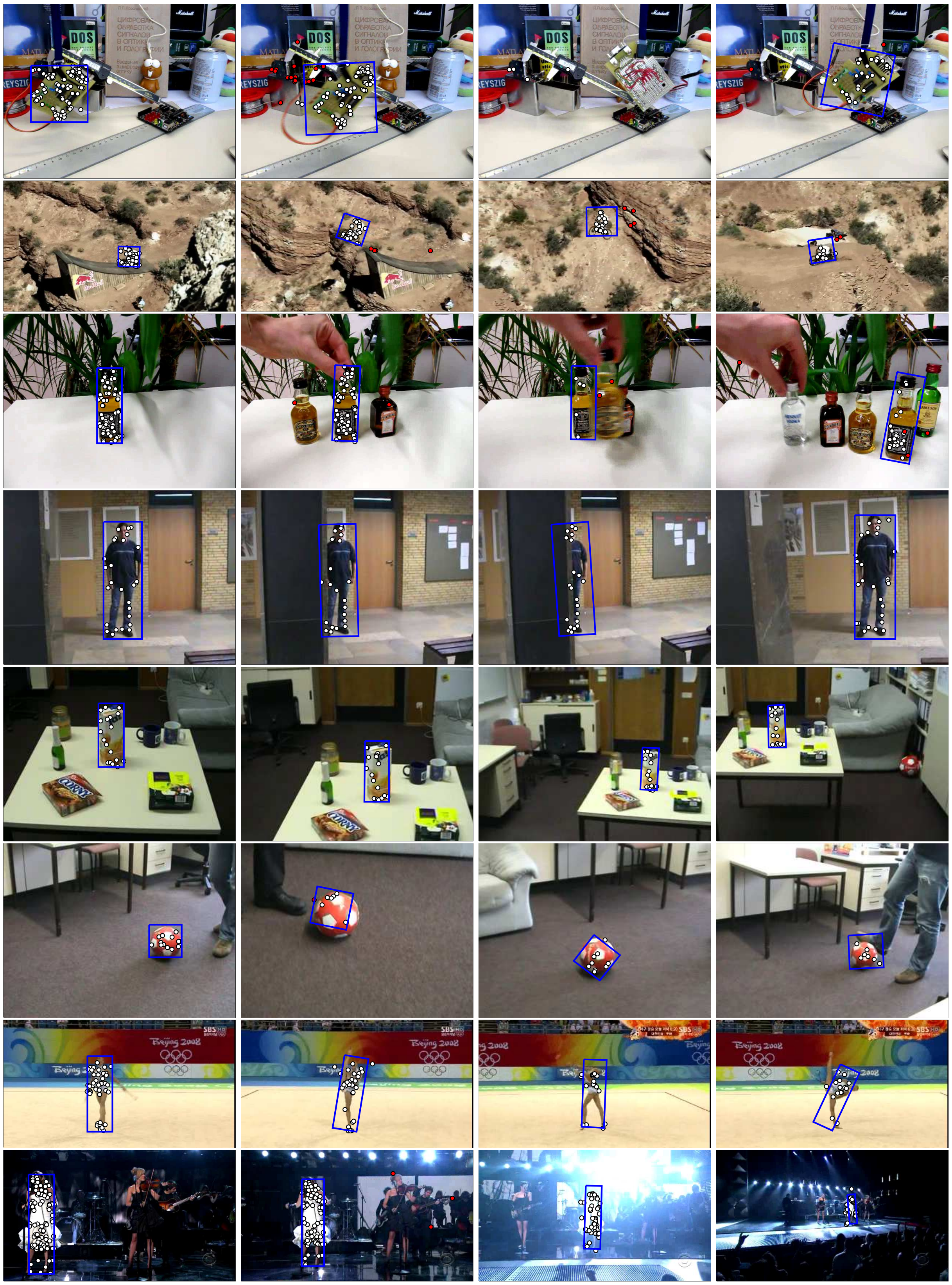

Figure 6: Qualitative results on *board*, *mountain-bike*, *liquor*, *person occ*, *juice*, *ball*, *gym* and *singer*.

				Recall			
Sequence	CMT	STRUCK	TLD	FT	LM	HT	SB
ball	**0.98/0.57/0.19**	0.30/0.15/0.10	0.40/0.28/0.19	0.31/0.19/0.13	0.14/0.12/0.09	0.15/0.11/0.10	0.30/0.28/0.12
board	0.84/**0.83**/0.23	**0.91**/0.81/**0.29**	0.45/0.19/0.02	0.82/0.70/0.19	0.33/0.32/0.06	0.26/0.21/0.04	0.15/0.14/0.11
box	0.94/0.83/**0.66**	**0.99/0.90**/0.33	0.39/0.35/0.15	0.07/0.05/0.00	0.63/0.61/0.44	0.14/0.05/0.02	0.37/0.28/0.08
car	0.59/**0.45**/0.03	**0.98**/0.21/0.05	0.52/0.14/0.05	0.33/0.10/0.05	0.14/0.10/0.03	0.57/0.17/0.03	0.12/0.09/**0.06**
car 2	0.90/0.88/0.64	0.81/0.47/0.11	**1.00/1.00/0.95**	0.04/0.04/0.03	0.46/0.36/0.17	0.59/0.47/0.00	0.72/0.72/0.70
carchase	**0.30/0.20/0.07**	0.08/0.03/0.02	0.16/0.15/0.06	0.04/0.03/0.02	0.00/0.00/0.00	0.04/0.04/0.00	0.08/0.08/0.05
cup on table	0.83/0.81/**0.61**	**1.00**/0.92/0.35	0.89/0.64/0.06	**1.00**/0.88/0.40	0.68/0.54/0.31	**1.00/1.00**/0.48	0.47/0.47/0.34
dog1	**1.00/0.95/0.72**	0.86/0.67/0.22	0.90/0.80/0.23	0.84/0.63/0.15	0.77/0.74/0.12	0.83/0.59/0.23	0.51/0.50/0.30
gym	0.93/0.86/0.22	**1.00/0.93/0.30**	0.76/0.32/0.08	0.24/0.22/0.12	0.10/0.05/0.02	0.30/0.00/0.00	0.61/0.58/0.22
juice	**1.00/1.00**/0.97	**1.00**/0.48/0.41	**1.00**/0.44/0.35	0.09/0.08/0.07	**1.00/1.00/0.97**	**1.00**/0.44/0.00	0.43/0.43/0.41
jumping	0.90/0.81/0.32	**1.00**/0.80/0.17	0.88/**0.88/0.65**	0.39/0.24/0.11	0.14/0.10/0.05	0.99/0.33/0.16	0.07/0.07/0.07
lemming	0.65/0.62/**0.29**	**0.72/0.66**/0.23	0.15/0.07/0.03	0.16/0.14/0.07	0.02/0.01/0.01	0.29/0.23/0.10	0.17/0.16/0.07
liquor	**0.91/0.89/0.85**	0.74/0.66/0.49	0.70/0.68/0.23	0.46/0.44/0.38	0.86/0.84/0.70	0.07/0.00/0.00	0.43/0.43/0.43
mountain-bike	**0.99/0.98/0.48**	**0.99**/0.93/0.23	0.37/0.36/0.16	0.65/0.63/0.18	0.11/0.08/0.04	**0.99**/0.40/0.03	0.20/0.17/0.08
person	0.95/0.82/0.49	**1.00/0.95**/0.50	0.92/0.71/0.25	**1.00**/0.95/**0.54**	0.75/0.67/0.31	0.49/0.00/0.00	0.52/0.52/0.40
person crossing	0.76/0.70/**0.58**	0.51/0.42/0.12	0.86/0.70/0.10	0.88/0.66/0.15	0.80/0.75/0.42	0.18/0.10/0.04	**0.96/0.91**/0.16
person occ	**1.00**/0.94/**0.82**	**1.00**/0.91/0.80	**1.00**/0.87/0.58	**1.00**/0.91/0.80	**1.00/0.95**/0.82	**1.00**/0.93/0.44	0.99/0.91/0.80
singer	**1.00/0.99/0.52**	0.48/0.28/0.14	0.77/0.70/0.28	0.52/0.28/0.11	0.21/0.21/0.20	0.24/0.09/0.00	0.15/0.15/0.13
sylvester	0.99/**0.96**/0.55	0.99/0.96/0.38	0.97/0.94/0.31	0.86/0.73/0.33	0.62/0.49/0.18	**1.00**/0.95/**0.64**	0.41/0.40/0.29
track running	**1.00/1.00/0.84**	**1.00**/0.25/0.11	0.01/0.01/0.01	0.94/0.25/0.08	0.01/0.01/0.01	0.20/0.00/0.00	0.31/0.24/0.08
avg	**0.87/0.81/0.50**	0.82/0.62/0.27	0.65/0.51/0.24	0.53/0.41/0.20	0.44/0.40/0.25	0.52/0.31/0.12	0.40/0.38/0.25

Table 1: Obtained recall on 20 sequences for low/medium/high requirements on accuracy.

Table 1 we present the obtained recall for low, medium and high requirements on accuracy and additionally provide averaged values. Maximal values are highlighted. Our algorithm achieves highest recall in 9 (low acc.), 11 (medium acc.) and 12 (high acc.) out of 20 sequences and attains highest average recall in all categories For high accuracy, it is ahead of the second-ranking algorithm by a margin of 0.23 average recall points.

Different applications have different requirements with respect to performance metrics computed over the whole sequence. For instance, an application might be interested in a tracking algorithm that achieves a recall of 0.7 in at least 50% of all sequences. We employ success plots similar to [28] in order to provide this information for all possible requirements on the performance. A success plot is a two-dimensional plot where the x axis denotes the value of the performance measure at which the output on a sequence is considered a success, ranging from the theoretical minimum of the performance measure to its maximum. The y axis denotes the percentage of sequences where the performance of the tracker satisfies this requirement (success rate). Similar to precision-recall curves, an optimal tracker reaches the top right corner of the plot. In Figure 7 and Figure 8, success plots are given for medium and high accuracy. For these plots we employed 60 sequences, including the ones used for the tabular comparison. For medium accuracy, our tracker compares favourably to the other trackers, ranking second after STRUCK. When high accuracy is desired, our tracker clearly outperforms its competitors, most notably when the required recall is between 0.35 and 0.6.

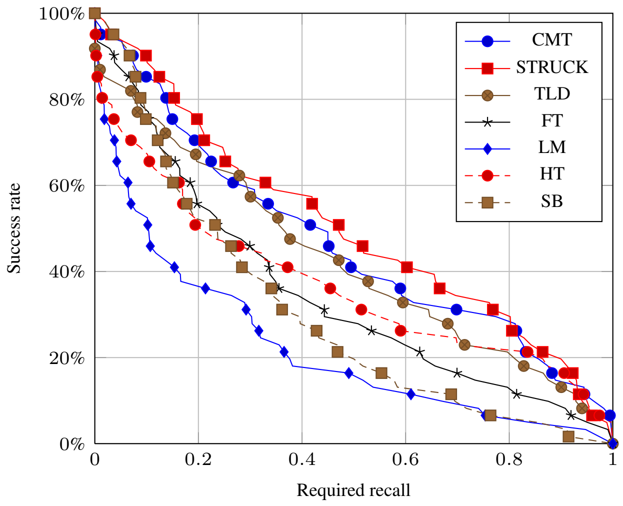

Figure 7: Success plot for medium accuracy ($\tau = 0.5$).

The experimental evaluation demonstrates that our proposed method is able to achieve state-of-the-art results and that it is especially well suited when high accuracy is desired. The extensive evaluation also reveals that trackers relying on strong assumptions about the object structure (such as LM) suffer a dramatic loss in performance when applied to arbitrary objects. This success of our method largely stems from the fact that we pose only slight assumptions about the geometric constellation of the keypoints. We have shown that keypoints are a powerful building block for tracking when embedded into an appropriate framework.

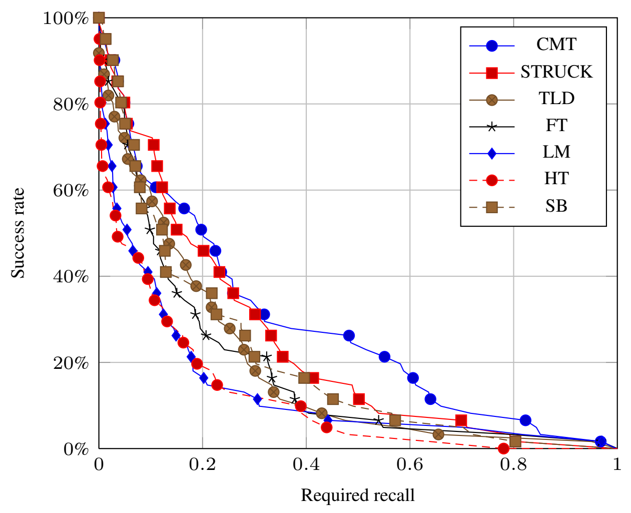

Figure 8: Success plot for high accuracy ($\tau = 0.75$).

5. Conclusion

In this work, we have presented a novel keypoint-based method for long-term model-free object tracking. We have shown in an extensive evaluation that our approach achieves state-of-the-art results on a large number of sequences. Compared to other approaches, our algorithm excels when high accuracy is desired. In future work we will address the question if updating the object model during processing can further improve the robustness of our tracker as well as how much our algorithm is affected by employing different methods for keypoint detection and description. Furthermore it might be interesting applying our method for outlier detection to other areas in computer vision, such as object category recognition.

Acknowledgements

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement no 257906.

References

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

[2] B. Babenko, M.-H. Yang, and S. Belongie. Robust object tracking with online multiple instance learning. *TPAMI*, 33(8), 2011.

[3] D. H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *IJPRAI*, 13(2), 1981.

[4] S. Duffner and C. Garcia. PixelTrack: a fast adaptive algorithm for tracking non-rigid objects . In *ICCV*, 2013.

[5] J. Gall, A. Yao, N. Razavi, L. Van Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *TPAMI*, 33(11), 2011.

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

[7] H. Grabner and H. Bischof. On-line boosting and vision. In *CVPR*, 2006.

[8] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised On-Line boosting for robust tracking. In *ECCV*, 2008.

[9] M. Grabner, H. Grabner, and H. Bischof. Learning features for tracking. In *CVPR*, 2007.

[10] S. Hare, A. Saffari, and P. H. S. Torr. Struck: Structured output tracking with kernels. In *ICCV*, 2011.

[11] S. Hare, A. Saffari, and P. H. S. Torr. Efficient online structured output learning for keypoint-based object tracking. In *CVPR*, 2012.

[12] B. Hemery, H. Laurent, and C. Rosenberger. Comparative study of metrics for evaluation of object localisation by bounding boxes. In *ICIG*, 2007.

[13] G. Hua and Y. Wu. Measurement integration under inconsistency for robust tracking. In *CVPR*, 2006.

[14] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-Learning-detection. *TPAMI*, 34(7), 2012.

[15] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *IJCV*, 77(1-3), 2008.

[16] S. Leutenegger, M. Chli, and R. Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *ICCV*, 2011.

[17] T. List, J. Bins, J. Vazquez, and R. B. Fisher. Performance evaluating the evaluator. In *PETS*, 2005.

[18] D. G. Lowe. Distinctive image features from Scale-Invariant keypoints. *IJCV*, 60(2), 2004.

[19] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981.

[20] E. Maggio and A. Cavallaro. *Video Tracking: Theory and Practice*. 2011.

[21] M. E. Maresca and A. Petrosino. MATRIOSKA: A multi-level approach to fast tracking by learning. In *ICIAP*, 2013.

[22] M. Özuysal, M. Calonder, V. Lepetit, and P. Fua. Fast key-point recognition using random ferns. *TPAMI*, 32(3), 2010.

[23] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature harvesting for Tracking-by-Detection. In *ECCV*. 2006.

[24] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *TPAMI*, 32(1), 2010.

[25] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *ICCV Workshops*, 2009.

[26] J. Santner, C. Leistner, A. Saffari, T. Pock, and H. Bischof. PROST: Parallel robust online simple tracking. In *CVPR*, 2010.

[27] S. M. Shahed Nejhum, J. Ho, and M.-H. Yang. Visual tracking with histograms and articulating blocks. In *CVPR*, 2008.

[28] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *CVPR*, 2013.

[29] R. Xu and D. Wunsch. Survey of clustering algorithms. *TNN*, 16(3), 2005.

[30] L. Zhang and L. van der Maaten. Structure preserving object tracking. In *CVPR*, 2013.