PICASO: PIxel Correspondences And SOft Match Selection for Real-time Tracking

Radu Timofte\textsuperscript{a,*}, Junseok Kwon\textsuperscript{a}, Luc Van Gool\textsuperscript{a,b}

\textsuperscript{a}Computer Vision Lab, D-ITET, ETH Zurich, Sternwartstrasse 7, Zurich 8092, Switzerland
\textsuperscript{b}VISICS/iMinds, ESAT, KU Leuven, Kasteelpark Arenberg 10, Heverlee 3001, Belgium

Abstract

Visual tracking is one of the computer vision’s longstanding challenges, with many methods as a result. While most state-of-the-art methods trade-off performance for speed, we propose PICASO, an efficient, yet strongly performing tracking scheme. The target object is modeled as a set of pixel-level templates with weak configuration constraints. The pixels of a search window are matched against those of the surrounding context and of the object model. To increase the robustness, we match also from the object to the search window, and the pairs matching in both directions are the correspondences used to localize. This localization process is robust, also against occlusions which are explicitly modeled. Another source of robustness is that the model - as in several other modern trackers - gets constantly updated over time with newly incoming information about the target appearance. Each pixel is described by its local neighborhood. The match of a pixel is taken to be the one with the largest contribution in its sparse decomposition over a set of pixels. For this soft match selection, we analyze both $l_1$ and $l_2$-regularized least squares formulations and the recently proposed $l_1$-constrained ‘Iterative Nearest Neighbors’ approach. We evaluate our tracker on standard videos for rigid and non-rigid object tracking. We obtain excellent performance at 42fps with Matlab on a CPU.

1. Introduction

Visual tracking is a core computer vision task. Surveillance, robotics, human computer interaction, and driver assistance are only a few of the domains using it. Such commercial interest, combined with hardware developments, have fueled a relentless progress in tracker development. A robust tracker should handle noise, occlusions, rigid and non-rigid deformations, background clutter, and illumination changes (Yilmaz et al., 2006).

We propose a fast and robust tracker that finds matches between source and goal region pixels. The idea is that these regions enclose a tracked target object model in subsequent video frames (e.g. in Fig. the green region initializes the object model and plays the role of source region, while the yellow region is the goal region within which the target object is localized more precisely; going to the next frame a similar tracking process happens). For each pair of frames, our tracker matches bi-directionally between such regions, i.e. after matching pixels in the source and goal regions, these swap roles and matching is also tried in the opposite direction. A pair of pixels is in correspondence if the matching procedure works both ways.

In order to match pixels, a pixel is characterized through a template, that consists of the intensities of the pixels in a rectangular neighborhood. These template intensities from the search window are expressed as a sparse linear decomposition in the template intensities of the object pixels and the context pixels. The template that dominates this decomposition is the match and it will cor-
responds to either a pixel from the object model or the context. The object pixel templates are also decomposed over the search window pixels, the pixels corresponding to the dominant templates in the decomposition being selected as matching pixels. This soft match selection is performed efficiently via the recently introduced Iterative Nearest Neighboring procedure of Timofte and Van Gool (2015), an alternative to solving the $l_1$-regularized least squares problem. Our model adapts over time and takes possible occlusions into account, as will be explained in detail later. The position of the object in the next frame is estimated robustly by median filtering the displacements between corresponding pixels. Once the new object position has been predicted, we consider all the pixels falling inside the predicted region for the model update.

Our main contributions are threefold:

1. High performance and low time complexity visual tracker based on Pixel Correspondences And Soft match selection (PICASO tracker);
2. Fast soft match selection techniques for correspondences mining;
3. Occlusions handling by efficient masks and median filtering.

The remainder of the paper is organized as follows. We summarize related work in Section 1.1. The proposed method and its components are introduced in Section 2. The influence of the method’s parameters is analyzed in Section 3.2. The experimental results on standard datasets are reported and discussed in Section 3.3 while the conclusions are drawn in Section 4.

1.1. Related Work

Although there is a vast literature on visual tracking, we only discuss related work grouped into 3 categories.

**Tracking by Correspondence** methods most similar to ours are of Pernici and Del Bimbo (2014), Babu and Parate (2013), Nebehay and Pflugfelder (2014), Vojir and Matas (2014), Dekel et al. (2015) and Kwon and Lee (2013).

The Appearance Learning In Evidential Nuisance (ALIEN) tracker of Pernici and Del Bimbo (2014) uses SIFT features, the nearest neighbor ratio for matching, adaptive object and context models, and RANSAC for homography estimation. Our PICASO on the other hand, uses simpler features and models, soft match selection methods, and deals with occlusions in a principled way. Moreover, it can cope with stronger non-rigid object deformations and is faster. Similarly to ALIEN, the Consensus-based Matching and Tracking of Keypoints (CMT) approach of Nebehay and Pflugfelder (2014) uses BRISK features, but a consensus-based scheme and clustering of votes for changes prediction, and no appearance model updates.

The Flock of Trackers (FoT) of Vojir and Matas (2014) runs a collection of local trackers for different object regions and estimates a global motion from their combined, estimated transformations. In the same vein, the tracker of Babu and Parate (2013) is casted as a 2-way assignment problem between local blocks of pixels in the object model and local blocks in a search window, by means of $l_1$ minimization. Compared to both these trackers, PICASO uses a denser set of pixels, also from the context, and considers occlusion masks. Moreover it uses a robust matching method to determine matches. As a result, our PICASO tracker tends to produce more accurate correspondences for visual tracking.

Recently, Dekel et al. (2015) proposed Best-Buddies Similarity (BBS) for template matching. BBS counts pairs of bi-directional nearest neighbor points from source and target sets extracted over an uniform grid. For tracking 100 points and a sliding window is employed. Compared with BBS, PICASO extracts a denser set of pixels (> 600) ranked by corner scores, includes context, employs a more robust matching, occlusion masks, different localization and model update procedures, and no grid or sliding window search. PICASO is much faster.

The tracker of Kwon and Lee (2013) is modeling the target object with a connected set of patches and tracking a highly non-rigid target. This tracker, however, suffers from the high computational complexity as the number of patches increase. In comparison, PICASO is real-time.

**Tracking as Classification** methods consider tracking as a binary classification problem, with the trackers of Lu and Hager (2007), Avidan (2007), Kalal et al. (2012), Hare et al. (2011), Zhang et al. (2014), and Henriques et al. (2015) as examples akin to our approach.

The tracker of Lu and Hager (2007) samples patches to
build foreground and background models, trains a binary classifier and runs mean shift on its confidence map to locate the object. A consistency check ensures that only inliers are used to update the models. A superpixel segmentation determines the boundary between object and background. The similarity to PICASO lies in the use of local patches and the use of an object and a background model. The tracker of Avidan (2007) trains an ensemble of weak classifiers, generates a confidence map for pixels belonging either to the object or the background, and also finds the new object position through mean shift. Kalal et al. (2012) also use classifiers, trained online, to distinguish the object from its surrounding background. Henriques et al. (2015) employ kernelized correlation filters based on circulant matrices and discrete Fourier transform for fast tracking. Zhang et al. (2014) propose a fast compressive tracking algorithm employing sampling, multiscale filter banks and compressed features, followed by a binary classifier. Hare et al. (2011) follow the tracking-by-detection framework and estimate the object transformation between frames instead of learning a binary classifier.

PICASO tracker differs from these trackers because it does without explicit training and learning, instead matches pixels with a sparse decomposition procedure and focuses on robust techniques to locate the object and to check the inliers. Another difference is that in our case also the outliers can be used to update the model, which makes it possible to much more swiftly include newly visible pixels, just like ALIEN can.

Tracking with \(l_1\) Optimization started with the seminal paper of Mei and Ling (2009). This and related papers model the object with a set of rigid whole object templates over which the \(l_1\) regularized least squares formulation is solved to decide on the presence of the object at a position. Moreover, the time complexity is high. To address this issue, Bao et al. (2012) use an accelerated proximal gradient approach.

Perhaps the best recent method is the one by Zhang et al. (2013). The \(l_1\)-tracker of Mei and Ling (2009) is shown to be a particular case of Zhang et al.’s Structured (or not) Multi-Task Tracking (S-MTT and MTT in shorthand) formulations. S-MTT and MTT have comparable average performances. S-MTT and MTT show good tracking performance, but are computationally complex. Instead, PICASO is real-time.

Tracking with \(l_1\) Optimization started with the seminal paper of Mei and Ling (2009). This and related papers model the object with a set of rigid whole object templates over which the \(l_1\) regularized least squares formulation is solved to decide on the presence of the object at a position. Moreover, the time complexity is high. To address this issue, Bao et al. (2012) use an accelerated proximal gradient approach.

Perhaps the best recent method is the one by Zhang et al. (2013). The \(l_1\)-tracker of Mei and Ling (2009) is shown to be a particular case of Zhang et al.’s Structured (or not) Multi-Task Tracking (S-MTT and MTT in shorthand) formulations. S-MTT and MTT have comparable average performances. S-MTT and MTT show good tracking performance, but are computationally complex. Instead, PICASO is real-time.
Our PICASO tracker is based on finding correspondences between pixels in the model window $\mathcal{D}$ and the search window $\mathcal{G}$. In particular, this pixel matching (see Section 2.3) is carried out in both directions: pixels in $\mathcal{D}$ look for their best match in $\mathcal{G}$ and pixels in $\mathcal{G}$ look for their best match in the combination of $\mathcal{D}$ and $\mathcal{C}$, the context of $\mathcal{G}$. The final correspondences are pairs of pixels that best match in both directions between $\mathcal{G}$ and $\mathcal{D}$. These correspondences are used to locate the object in the new frame (see Section 2.4). This new localization yields the next $\mathcal{D}$ to proceed with the tracking.

2.2. Object Model and its Updates

As said, the target object model mainly consists of pixel neighborhood descriptions (templates). Yet, from the start, not all pixels are considered. In the very first frame, the pixels in $\mathcal{D}$ and $\mathcal{C}$ are rank-ordered according to their Harris corner scores and then a fixed maximum number with super-threshold scores is kept. Also at initialization a first matching happens between the surviving pixels of the model window $\mathcal{D}$ and those of its context $\mathcal{C}$. A fixed portion of the object pixels, namely those that best match context pixels, are removed from the model. In our experiments this portion was set to $1/4$. Thus, we deal with the fact that the rectangular object window may contain background pixels. For subsequent frames a similar selection on Harris scores happens, but now on all pixels in $\mathcal{G}$ and its context $\mathcal{C}$.

During tracking, the model is also updated. The object model $\mathcal{D}$ is a compound of pixels from the initialization window, which is of special importance as it was manually annotated and therefore reliable, and an adaptive part containing pixels observed during subsequent frames. In particular, 50% of the model pixels are selected from the initial pixels and the other 50% are added during tracking. The second frame provides 50% of such new pixels in one go, whereas subsequent frames are allowed to each replace only 10% (as often found in the literature (Yilmaz et al., 2006)), taken among the updatable 50%. The 10% replacing pixels are taken from the strongest matches at each stage of the tracking. The 10% that is replaced are those pixels that have been matched the least successfully. Restricting the number of pixels in the model helps the system to keep up its speed. It is also for reasons of speed that the object window size is restricted to a few hundred pixels, by downscaling the image regions under consideration.

2.3. Soft Match Selection

One can expect that the pixels in a scene share a number of similar textural patterns in their neighborhood. In other words, the pixel templates can be expected to populate a number of template subspaces. Therefore, it stands to reason that a pixel’s template from a source region can be decomposed into a sparse, linear combination of goal region templates. The enforced sparsity ensures that only the most relevant goal pixel templates contribute. The most relevant template will receive the largest weight in the sparse decomposition.

We refer to this process of sparse decomposition as soft match selection. It stands in contrast to usual matching procedures that compare against candidates one at a time. Here the selection of the single, actual ‘match’ is based on summing all the weights in the decomposition that corresponds to the same template (the decomposition is typically built iteratively and the same template can be selected multiple times). Hence, the matching procedure selects a single candidate, in contrast to ‘soft assignment’ procedures that distribute votes over multiple candidates (see Fig. 3).

We compute for each pixel from the object model ($\mathcal{D}$) its match in the search window ($\mathcal{G}$) and for each pixel in the search window ($\mathcal{G}$) its match among the object model and context pixels, $\mathcal{D} \cup \mathcal{C}$. The pairs of pixels, that are in bijection, are the correspondences that we use for estimating the position of the object within the search window: $\mathcal{A} = \{(d, g) \in \mathcal{D} \times \mathcal{G}\}$.

Next, we describe the soft match selection procedure of our PICASO tracker in more detail. A standard approach for obtaining the match of a query feature vector $\mathbf{q}$ (pixel template in our case) is its nearest neighbor (NN) selection, $x_q$ in the pool of $M$ candidates (pixel templates)
\[ X = \{ x_1, x_2, \ldots, x_M \}, \text{ where:} \]
\[ a = \arg \min_i ||q - x_i||_2. \]  
\[ (1) \]

Our soft match selection requires obtaining a sparse decomposition, for which one may use the least squares formulation with a regularization controlled by the parameter \( \lambda \):
\[ \hat{w} = \arg \min_w ||q - Xw||_2 + \lambda ||w||_l, n = 1 \text{ or } 2. \]  
\[ (2) \]

where \( w \) are the coefficients (weights), and \( l_n \) is the norm imposed over the coefficients of the representation.

If we use the \( l_1 \) norm in (2), then the sparsity of the solution \( \hat{w} \) is enforced and we are in the case of a Sparse Representation (SR) (\cite{Wright2009}).

If we use the \( l_2 \) norm in (2), then the sparsity can be enforced by imposing a very large \( \lambda \), thus putting less importance on minimizing the reconstruction error. We are in the Ridge Regression (RR) formulation (\cite{Timothe2013}) and have a closed form solution for (2):
\[ \hat{w} = (X^T X + \lambda I)^{-1} X^T q \]  
\[ (3) \]

where \( P = (X^T X + \lambda I)^{-1} X^T \) can be precomputed.

The Iterative Nearest Neighbors (INN) method of \cite{Timothe2015} combines the power of SR with the computational simplicity of NN by means of a constrained least squares decomposition:
\[ \arg \min_{s_i} ||q - \sum_{i=1}^{K} \frac{\lambda}{(1 + \lambda)^i} s_i ||_2 \]  
\[ (4) \]

where \( \lambda \) is the regulatory scalar parameter, \( s_i \) are samples selected from \( X \), and the imposed weights (\( \frac{1}{(1 + \lambda)^i} \)) sum up to 1 for \( K \to \infty \). The same sample \( x_j \in X \) may be selected multiple times in (4), therefore in the INN representation its overall coefficient will be the sum of the weights corresponding to its different selections (\( s_i \)'s). As shown by \cite{Timothe2015}, with a tolerance \( \beta = 0.05 \) one recovers the coefficients up to a summed value of 0.95 and one needs \( K \) NN iterations of the INN algorithm, where \( K \):
\[ K = \lceil -\log(1 - \beta) \quad \log(1 + \lambda) \rceil. \]  
\[ (5) \]

Let \( \hat{w} \) be the coefficients in the resulting representation.

When using the above decomposition methods – SR, RR, and INN – for matching, we look for the most important sample, \( x_a \), in the representation of \( q \) over \( X \), based on the magnitude of the coefficients:
\[ a = \arg \max_i |\hat{w}_i|. \]  
\[ (6) \]

The selected match of \( q \) in \( X \) is \( x_a \).

2.4. Localization with Occlusion Handling

At this point we have a set of correspondences \( A \) and their position displacements. For estimating the object location we distinguish two cases, with their complications:

1) Non-occluded case: here we can still have non-object correspondences and correct object correspondences can have different displacements due to object deformations;

2) Occluded case: here, in addition, we can have correspondences to the occluder, which, fortunately, tend to disagree on a single displacement.

First we handle the non-occluded case. For obtaining the dominant displacement we take the medians of the displacements along the horizontal and vertical axes individually. To further increase the robustness, we drop one third of the correspondences that are furthest from these medians and from the remaining correspondences we extract refined median displacements. This displacement calculations may seem rather restrictive in terms of the object motions it can handle, but also in cases of object rotations and even deformations we found it to be effective.

The median filtering procedure per se often suffices to deal with occlusions, with up to 50% of the correspondences incorrect. Yet, under serious occlusion, as when a rigid block occludes 50% or more of the object, extra counter-measures are needed. We propose the use of occlusion masks. Simply put, we apply a number of masks to suppress the correspondences in some fixed areas of the search window. We chose masks covering 25% or 50% of the window area, as illustrated in Fig.\[\]E For each of these masks, and for the unoccluded assumption, we calculate the displacement estimates. Finally, we take the median of these displacement estimates as the final displacement. One can derive that this way the odds of ending up with the correct displacement improve.
3. Experiments

3.1. Implementation details

Whenever the initialization size of the object is larger than 2000 pixels we scale down all the images such that the tracker normalized window size is 1024. In this way, for most of the sequences the processing time per frame is comparable. Moreover, apart from keeping up the computation times, working with normalized windows allows us to work with a more consistent set of parameters. This window size still yields sufficient discriminant information. Note that for face or pedestrian detection for instance, the normalized window has often been set to less than 1024 pixels (Viola and Jones, 2004).

We further assume that the object displacement from one frame to the next is no larger than half its size. Thus, the search window is fixed to a 1.5 times the object window size (same aspect ratio). For extracting the context pixels, we fix the working window to a 2.5 times the object window size. The rim between its boundary and the search window contains the context pixels. All the windows are centered to the predicted object position in the previous frame. For a tracked object of 1024 pixels, the total working window contains 1024×2.5² = 6400 pixels.

For pixel selection we use the Harris corner detector of Kovesi (2014), where we set the parameters sigma blur to 1, radius to 0.5, and the threshold changes dynamically to extract the desired number of pixels. A pixel’s template covers the surrounding block of 5 × 5 pixel intensities. This template vector is l2-norm normalized. Harris corner scores ranking can be less reliable in the case of weak or unstable patterns, however using a large number of pixels (> 100) assures statistically useful information.

For solving the SR formulation we use the lasso Matlab wrapper function from SPAMS library Mairal et al. (2014), while for INN we use the Matlab script of Timote and Van Gooll (2012). Apart from the solver for SR, the entire implementation is in Matlab. We will release the codes.

3.2. Analysis of the Proposed Method

As most tracking methods in the literature, PICASO has a number of parameters. Next we evaluate the impact of some of these parameters on the overall performance. For this purpose we pick 9 standard sequences: ‘Car11’, ‘Jumping’, ‘Deer’ as used in Wang et al. (2013); ‘Singer’, ‘Torus’, ‘Woman’ as used in Kristan et al. (2013); ‘Surfer’, ‘Coupon Book’ from Babenko et al. (2009); and ‘Panda’ from Kalal et al. (2012). These sequences, as a separate training set, are different from those we use to benchmark in the next section 3.3. The recall also known as rate of success and rate of correctly tracked frames is taken as quantitative performance measure, i.e. the percentage of frames in which the tracker output overlaps sufficiently with the ground truth annotation. We use the standard overlapping measure \(A_T \cap A_G\) between the tracker window around the object, \(A_T\), and the area, \(A_G\), of the ground truth object window. If the overlap is above a fixed threshold of 0.5 we consider that the object is tracked correctly in that frame. The average recall is the average of the recall values obtained for the individual sequences. Center location errors (CLE) for our methods are found in the supplementary material.

**Soft Match Selection Parameters** Soft match selection is at the core of our system. We can implement it with SR, INN, or RR methods. Each has a regulatory parameter (λ) controlling the performance. Fig. reports how the average recall of PICASO tracker on the training 9 videos
varies with the parameter of each matching method when we extract maximum 600 pixels and their templates. NN is reported for reference. The jittering in the recall results is mainly due to the fact that PICASO loses track at different frames for some sequences. The trend however, is visible in the recall results. As expected, for large $\lambda$-intervals, the soft match selection approaches lead to a better recall than the NN baseline. SR and INN are the top performers for recall. RR improves over NN if $\lambda$ is set quite large such that the sparsity of the solution is enforced even if we have an $l_2$-norm regularization. Both INN and SR reach their best recall for small $\lambda$ values.

Fig. 4 reports how the running time in frames per second (fps) varies with the ($\lambda$) parameter of each matching method. NN is reported for reference, it is the fastest matching method among the competitors. RR has a relatively stable running time, that does not depend on the $\lambda$ parameter, while INN and SR get slower when $\lambda$ decreases, and thus, the sparsity is relaxed. A good tradeoff between performance and speed is achieved for $\lambda \in (0.05,0.5)$ for both INN and SR. Therefore we set $\lambda = 0.25$ for INN and $\lambda = 0.15$ for SR in the following experiments, while for RR we take $\lambda = 10^7$.

Fig. 6 shows that PICASO (INN, $\lambda = 0.25$) performs significantly worse without taking into consideration the context pixels during the matching process.

**Soft Match Selection vs. Soft assignment** Soft assignment is an one-to-many relationship, while our soft match selection is one-to-one. For PICASO we can replace eq. (6) of the soft match selection from Section 2.3 with a soft assignment one:

$$[a]_j = \arg_r (|\hat{w}_i| > 0), \quad (7)$$

where $\hat{w}$ are the sparse decomposition coefficients of INN or SR of the query $q$ over the pool of candidates $X$ and $[a]_j$ are the indeces of the nonzeros picked for assignment. We plug the soft assignment into our PICASO method and the obtained results are shown in Fig. 5. Overall the results with soft match selection are better than those using soft assignment. For example, for INN ($\lambda = 0.25$) and maximum number of pixels (density) of 600, PICASO achieves an average recall of 70.9% on our training videos with soft match selection and 61.2% with soft assignment. With SR ($\lambda = 0.15$) the soft assignment leads to 63.4%.

**Pixel Density** The impact of the maximum number of pixels on the recall performance of PICASO tracker is depicted in Fig. 5 for the training videos. All parameters are fixed, only the maximum number of pixels is changing. Despite the jittering in the recall results, a trend is visible. When the number of pixels is small ($\leq 100$) the differ-
ence between the soft match selection methods is small. This is due to the fact that at this level all extracted pixels have relatively strong Harris corner scores and are quite discriminant. Then NN search suffices to establish correspondences. The larger the number of pixels, usually the better the recall when combined with SR, INN, or RR soft match selection. Since we have a normalized working area of fixed size (6400 pixels), we face a plateau in recall when the number of extracted pixels is above 1500 pixels, as the remaining pixels are usually in homogeneous areas, locally redundant, or less informative. Fig. 5 also shows the impact the number of pixels has on running time. We conclude that we are able to achieve very good speed and recall performance.

Based on these results, the final choices for our PICASO method are: soft match selection based on INN, with $\lambda = 0.25$ and 600 as maximum number of pixels extracted in the working area. PICASO operates in real-time at 42 frames per second with Matlab scripts on a desktop with CPU Intel i7-4770K.

**Pixel Descriptors** PICASO’s pixels are described by their surrounding block of pixel intensities of size $5 \times 5$. If we use templates of size $3 \times 3$, PICASO’s average recall performance on our 9 training videos drops from 70.89% to 53.66%. If we use bigger $7 \times 7$ templates then the performance drops to 67.87% and also the running time increases. Besides pixel intensities we tried commonly used point descriptors such as SURF of Bay et al. (2008), FREAK of Alahi et al. (2012), and BRISK of Leutenegger et al. (2011) by plugging them directly into PICASO. We used their default parameters, all are 64 dimensional descriptors. If PICASO with pixel intensities achieves 70.89% average recall on the training videos, with SURF descriptors it gets 51.7%, with FREAK - 52.8%, and with BRISK - 49.2%, resp. In Fig. 7 we vary the number of maximum pixels. PICASO works the best with templates of pixel intensities of size $5 \times 5$.

**Occlusion Masks** For handling the occlusions and also to robustify the tracker prediction (localisation) in presence of (non-rigid) deformations or other image corruptions (such as motion blur, illumination variation) we employ occlusion masks and median filtering as explained in Section 2.4. To quantify and assess the importance of this component in PICASO we plot the results with and without employing masks in Fig. 6. When the masks are not employed the predicted location is based on the median of displacements after pruning one third of them. Without occlusion masks the average recall drops 8% with INN ($\lambda = 0.25$) and 3% with SR ($\lambda = 0.15$) when maximum 600 pixels are used. Increasing the maximum number of pixels usually increases the number of correspondences. The wrong correspondences (also those on the occluded regions) usually do not agree on a consistent displacement and thus the importance of the masks diminishes with the increase of maximum number of pixels (above 1700). Fig. 8 shows examples from ‘Panda’ and ‘Coke’ videos where the masks help PICASO to handle occlusions and to keep the track.

**Object Model** The PICASO tracker is rather rigid in modeling the positions of pixels relative to the object center. While its adaptive part allows it to gradually add pixels at new positions and thus to cope with some object deformations and appearance changes, it is not an object model designed for large deformations. Alternative choices of updating pixels and how PICASO could better cope with large (rigid/non-rigid) deformations are topics for future investigation.

**Scale and Aspect Ratio** The current PICASO tracker

---

Matlab Computer Vision System Toolbox
does not change the scale or aspect ratio of the target object. Even if it deals with the scale changes in the different example videos pretty well though, one direction of future work could be to explicitly include scale adaptation. For instance, one could use the correspondences to not only propose the displacement, but also a new object scale. A single correspondence proposes a scale change by comparing the distances of the source and the goal pixels to the corresponding window centers. As before, in order to gain robustness, we could then obtain the median scale, prune one third of the proposals, and get the final median scale estimate. In our preliminary implementation of the aforementioned scale estimation we obtained inconclusive results. On average the average recall of PICASO without scale adaptation is better (70.89%) than that with (64.23%) on our training sequences. The main reason for this is that the correspondences are not sufficiently accurate or dense in the critical parts with large deformations/image corruption of the video sequences. Small errors in scale estimation can accumulate and drift the tracker. This is a topic for future investigation.

Track Recovery The PICASO tracker is quite robust and could determine when it encounters a difficult prediction. The number of correspondences and the comparison of displacements with the occlusion masks can help determine when best to employ a specific track recovery procedure. One way to recover is to gradually increase the search window until the object is located with a significant number of correspondences. This and other procedures to recover are directions of future work.

3.3. Comparison with Other Tracking Methods

Benchmark Our main benchmark consists of two datasets of challenging standard videos, as in Duffner and Garcia (2013). It covers a large number of rigid (8 videos, Babenko et al. (2009) and non-rigid (11 videos, Godec et al. (2013)) pose changes and different challenges due to occlusion, motion blur, illumination conditions, or background changes. We compare our PICASO tracker with recent state-of-the-art methods.

Although there are many state-of-the-art tracking methods (Kristan et al., 2013; Wu et al., 2013), we compare most relevant trackers with ours, as follows: HoughTrack (HT) proposed by Godec et al. (2013), Tracking-Learning-Detection (TLD) proposed by Kalal et al. (2012), PixelTrack, a HT-variant of Duffner and Garcia (2013), the structured multi-task sparse learning method (MTT) of Zhang et al. (2013), the ALIEN method based on oversampling invariant features proposed by Pernici and Del Bimbo (2014), the enhanced Flock of Trackers (FoT) method of Vojir and Matas (2014), the improved MeanShift method (MSfb) of Vojir et al. (2011), the fast compressive tracker (FCT) of Zhang et al. (2014), and the Kernelized Correlation Filters (KCF) method of Henriques et al. (2015). Some of these methods are designed especially for non-rigid tracking, such as HT, PixelTrack, or MSfb, while others are bound to work best on cases with only small non-rigid deformations - MTT, ALIEN. This is the main reason for reporting results only for a subset of the methods for each benchmarking dataset. Note that we indirectly compare with all the trackers evaluated on the same datasets and setups.

We report the performance in terms of average recall, as in the previous experiments. For PICASO we use the parameters fixed in the previous section 3.2. We give results for both an overlap threshold of 0.1 or 0.5. 0.5 is more strict and requires coping well with scale and aspect ratio changes. 0.1 is used by Duffner and Garcia (2013) since their method tracks a subregion of the object and does not refine the scale and aspect ratio of the tracked output. The threshold is still sufficiently high to assure that the object is loosely tracked.

The results for HT, TLD and PixelTrack are the ones reported by Duffner and Garcia (2013). For MTT, FCT, and KCF methods we used the original codes provided by their respective authors. The ALIEN results are the ones reported by Pernici and Del Bimbo (2014). The Struck and MSfb results are the ones reported by the authors of MSfb, Vojir et al. (2013). The FoT results are extracted from the authors’ original work (Vojir and Matas, 2014).

The PICASO tracker does not use color information, but grayscale. For the non-rigid object tracking dataset we initialize the PICASO tracker to the centered half down-
scaled annotation from the first frame. Thus, PICASO tracks only a quarter of the rectangle area surrounding the object. In so doing, we avoid to include large amounts of background pixels into the object model. Note that HT and PixelTrack were manually initialized to a favorable bounding box for the reported results in their original works (Godec et al., 2013; Duffner and Garcia, 2013). We, on the other hand, apply just the same scaling rule everywhere on this dataset. In order to compute the reported performance of PICASO at high overlap thresholds, we up-scaled the prediction to compensate for the initial down-scaling.

We want to emphasize that we used the same fixed parameters throughout all the experiments. The only exception is the automatic half down-scaling for Godec videos.

**Tracking Accuracy** We give the results for the Babenko videos in Table 3.3. Our PICASO method achieves the best results by a large margin for both overlap thresholds. The most difficult videos are ‘Tiger 1’ and ‘Tiger 2’ where no method obtains a perfect score, but PICASO comes closest. These videos contain several severe occlusions and large displacements, while the searching window of PICASO is set quite small. Moreover, we have no explicit recovery procedure in the case of complete occlusions. This is another direction for future improvements. From Table 3.3, we see that, as expected, the methods designed especially for non-rigid tracking such as HT and PixelTrack fall behind the MTT or ALIEN methods, envisaged for tracking objects exhibiting little non-rigid deformation.

Fig. 9 shows tracking results. On ‘Tiger 2’ MTT loses track after frame 105 and never fully recovers, and on ‘Sylvester’ it experiences difficulties after frame 1100 to end up losing track after frame 1155. PICASO does not lose track on ‘Sylvester’. On ‘Tiger 2’ PICASO loses track at frame 243 but recovers by frame 250. Moreover, it is more accurate in tracking the object through most of the frames, despite the occlusions and appearance changes.

**Tracking Accuracy** We give the results for the Babenko videos in Table 3.3. Our PICASO method achieves the best results by a large margin for both overlap thresholds. The most difficult videos are ‘Tiger 1’ and ‘Tiger 2’ where no method obtains a perfect score, but PICASO comes closest. These videos contain several severe occlusions and large displacements, while the searching window of PICASO is set quite small. Moreover, we have no explicit recovery procedure in the case of complete occlusions. This is another direction for future improvements. From Table 3.3, we see that, as expected, the methods designed especially for non-rigid tracking such as HT and PixelTrack fall behind the MTT or ALIEN methods, envisaged for tracking objects exhibiting little non-rigid deformation.

Fig. 9 shows tracking results. On ‘Tiger 2’ MTT loses track after frame 105 and never fully recovers, and on ‘Sylvester’ it experiences difficulties after frame 1100 to end up losing track after frame 1155. PICASO does not lose track on ‘Sylvester’. On ‘Tiger 2’ PICASO loses track at frame 243 but recovers by frame 250. Moreover, it is more accurate in tracking the object through most of the frames, despite the occlusions and appearance changes.

For Godec videos we summarize the results in Table 2. Our PICASO method has the best average performance. For 9 out of 11 video sequences and the permissive 0.1 overlapping criteria PICASO gets 100% recall. It is followed by PixelTrack, 3.6% behind. At the 0.5 overlap threshold, we can unfortunately not compare directly with PixelTrack. For PICASO we have a fixed-sized rectangle centered on the track prediction. Even under these conditions, the performance of our method is 2.3% better on average than the second best method, MSfb. Note that PICASO works only with grayscale values and does not adapt or change the scale or aspect ratio of its windows during tracking. Yet, it loses track in only 2 sequences out of 11: ‘Skiing’ and ‘Diving’. PixelTrack looses track in 6 sequences.

Fig. 10 shows results of PICASO for 4 most difficult Godec videos. PICASO is losing track early on in the ‘Skiing’ video and never recovers. This is due to the dramatic changes in aspect ratio and the background clutter and the absence of a recovery procedure for our method. The dramatic changes in aspect ratio and the background clutter

Figure 9: PICASO and MTT on difficult Babenko videos: ‘Tiger 2’ (top) and ‘Sylvester’ (bottom). Pixel correspondences are marked by blue ×-es. PICASO output is with green rectangles, MTT output with yellow rectangles, while ground truth with red.

Figure 10: PICASO on 4 most difficult Godec videos: ‘Diving’, ‘Skiing’, ‘Motocross 1’, and ‘Cliff-dive 2’. Pixel correspondences are marked by blue ×-es. PICASO tracked area is with blue rectangles, while with green rectangles is the up-scaled tracker prediction and with red the ground truth.
Table 1: Babenko videos (rigid deformations): % of correctly tracked frames (recall). The best results are with **bold**.

<table>
<thead>
<tr>
<th></th>
<th>HT</th>
<th>TLD</th>
<th>PixelTrack</th>
<th>MTT</th>
<th>KCF</th>
<th>FCT</th>
<th>PICASO</th>
<th>FoT</th>
<th>MTT</th>
<th>ALIEN</th>
<th>KCF</th>
<th>FCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>David</td>
<td>89.25</td>
<td>77.42</td>
<td>45.16</td>
<td>51.61</td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td>100.00</td>
<td>N/A</td>
<td><strong>100.00</strong></td>
<td>100.00</td>
<td>100.00</td>
<td>100.00</td>
</tr>
<tr>
<td>Sylvester</td>
<td>55.02</td>
<td>88.10</td>
<td>36.80</td>
<td>86.99</td>
<td>88.10</td>
<td>91.82</td>
<td><strong>100.00</strong></td>
<td>45</td>
<td>79.55</td>
<td><strong>89</strong></td>
<td>80.67</td>
<td>55.02</td>
</tr>
<tr>
<td>Girl</td>
<td>92.22</td>
<td>97.01</td>
<td>93.21</td>
<td><strong>100.00</strong></td>
<td>91.09</td>
<td>99.01</td>
<td><strong>100.00</strong></td>
<td>23</td>
<td><strong>100.00</strong></td>
<td>66</td>
<td>85.15</td>
<td>30.69</td>
</tr>
<tr>
<td>Face Occ. 1</td>
<td>99.44</td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td>100.00</td>
<td>85.39</td>
<td><strong>100.00</strong></td>
<td>75.02</td>
<td>96.74</td>
<td>99</td>
<td>65.73</td>
<td>75.28</td>
<td><strong>100.00</strong></td>
</tr>
<tr>
<td>Face Occ. 2</td>
<td>88.34</td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td>100.00</td>
<td>91.09</td>
<td><strong>100.00</strong></td>
<td>79.55</td>
<td>78.53</td>
<td>87.65</td>
<td>79.14</td>
<td>80.67</td>
<td><strong>91.67</strong></td>
</tr>
<tr>
<td>Coke</td>
<td>72.88</td>
<td>88.14</td>
<td>91.53</td>
<td>72.88</td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td><strong>100.00</strong></td>
<td>40.28</td>
<td><em>65.21</em></td>
<td><em>80.33</em></td>
</tr>
<tr>
<td>Tiger 1</td>
<td>26.76</td>
<td>12.68</td>
<td>46.48</td>
<td>60.56</td>
<td>78.87</td>
<td><strong>94.29</strong></td>
<td>10</td>
<td>32.39</td>
<td>30</td>
<td>69.01</td>
<td>56.34</td>
<td><strong>72.86</strong></td>
</tr>
<tr>
<td>Tiger 2</td>
<td>41.10</td>
<td>19.18</td>
<td>98.63</td>
<td>58.90</td>
<td>90.41</td>
<td><strong>91.67</strong></td>
<td>11</td>
<td>20.55</td>
<td>N/A</td>
<td>41.10</td>
<td><strong>69.86</strong></td>
<td><strong>56.94</strong></td>
</tr>
<tr>
<td>average</td>
<td>72.08</td>
<td>66.07</td>
<td>75.02</td>
<td>89.85</td>
<td>91.56</td>
<td><strong>98.25</strong></td>
<td><strong>40.28</strong></td>
<td><strong>65.21</strong></td>
<td><strong>80.33</strong></td>
<td><strong>74.42</strong></td>
<td><strong>63.30</strong></td>
<td><strong>86.16</strong></td>
</tr>
</tbody>
</table>

are also reasons why PICASO looses track in the ‘Diving’ video over some short intervals, for instance, at frame 141 (see Fig. 10). Fortunately, the object is picked back up by frame 161, after which tracking continues. One observation is due here. PICASO uses an object model rigid by design. The power and robustness of the PICASO model comes from the high number of pixels and the fact that continuously new ones are added to the object model. As a result, PICASO has been performing well in cases with substantial non-rigid motion.

Without using context pixels PICASO’s recall drops to 94.9% on Babenko and 78.8% on Godec datasets but gets faster. Without automatic half downscaling on Godec datasets, PICASO gets 71.3% and 52.1% recalls for 0.1 and 0.5 thresholds, resp., below HT and PixelTrack manually initialized.

**Attribute-based performance** While in our main benchmark we had a clear distinction between rigid and non-rigid tracking, in the benchmark of Wu et al. (2013) each test sequence can be described with as much as 11 attributes such as ‘Scale variation’ or ‘Occlusion’. On Wu et al. (2013) benchmark we evaluated 31 recent tracking methods and our PICASO with default settings (as used for the Babenko videos) on 50 test sequences. As shown in Fig. 5 the more used pixels by PICASO the better the performance but also the slower it gets. Therefore we report, besides PICASO with maximum 600 pixels (42fps), results for other two setups with maximum 900 pixels (PI-
Table 3: Average processing speed in frames per second (fps). All values are as reported by other authors except values marked † computed on the same setup as our PICASO.

<table>
<thead>
<tr>
<th>Method</th>
<th>FPS</th>
<th>Codes</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>PICASO (ours)</td>
<td>41.67</td>
<td>Matlab</td>
<td>Intel i7-4770K</td>
</tr>
<tr>
<td>FoT (Vojir and Matas, 2014)</td>
<td>41.67</td>
<td>Matlab</td>
<td>Intel i7-3770</td>
</tr>
<tr>
<td>Struck (Hare et al., 2011)</td>
<td>125.00</td>
<td>C++</td>
<td>N/A</td>
</tr>
<tr>
<td>KCF (Henriques et al., 2015)</td>
<td>113.80</td>
<td>C++</td>
<td>Intel Xeon @ 3.4 GHz</td>
</tr>
<tr>
<td>PixelTrack (Duffner and Garcia, 2013)</td>
<td>113.80</td>
<td>C++</td>
<td>Intel Xeon @ 3.4 GHz</td>
</tr>
<tr>
<td>TLD (Godec et al., 2013)</td>
<td>74.54</td>
<td>Matlab/C++</td>
<td>Intel i7-4770K</td>
</tr>
<tr>
<td>ALIEN (Pernici and Del Bimbo, 2014)</td>
<td>110.00</td>
<td>Matlab/C++</td>
<td>Intel i7-4770K</td>
</tr>
<tr>
<td>SCM (Zhong et al., 2012)</td>
<td>10.61</td>
<td>Matlab/C++</td>
<td>Intel Xeon W3503</td>
</tr>
<tr>
<td>HT (Koestler et al., 2013)</td>
<td>10.61</td>
<td>Matlab/C++</td>
<td>Intel Xeon W3503</td>
</tr>
<tr>
<td>MTT (Babenko et al., 2013)</td>
<td>4.03</td>
<td>C++</td>
<td>Intel i7-970</td>
</tr>
<tr>
<td>Struck (Hare et al., 2011)</td>
<td>3.46</td>
<td>C++</td>
<td>Intel Pentium 4</td>
</tr>
<tr>
<td>ALIEN (Pernici and Del Bimbo, 2014)</td>
<td>1.32</td>
<td>Matlab</td>
<td>Intel i7-4770K</td>
</tr>
<tr>
<td>SCM (Zhong et al., 2012)</td>
<td>0.50</td>
<td>Matlab/C++</td>
<td>Intel i7-3770@ 3.4 GHz</td>
</tr>
</tbody>
</table>

CASOf900, 35fps) and with maximum 1200 pixels (PICASOf1200, 30fps). PICASO with 2000 maximum used pixels does not further improve the performance.

In Table 4 and Table 5 we only report the results for the top 12 tracking methods and ours on relation with 11 benchmark attributes. We employed the standard metrics of the benchmark. The average precision is computed as the area under the curve (AUC) for a precision plot as function of location error threshold (in pixels); while the average success rate is the AUC for a success rate plot as function of overlap threshold. PICASO proved to be a top tracking method. PICASO provides the best precision on sequences with attributes such as ‘In-plane’ and ‘Out-of-plane rotation’, ‘Scale variation’, ‘Occlusion’, ‘Motion blur’, and ‘Fast motion’ (see Fig. 11 and Table 4) and the best average success rate on 7 out of 11 attributes: ‘Illumination variation’, ‘In-plane’ and ‘Out-of-plane rotation’, ‘Occlusion’, ‘Motion blur’, ‘Fast motion’, and ‘Background clutter’. Noteworthy is that PICASO (with 1200 used pixels) ranks first overall ‘whole’ sequences for both average precision and average success rate. We conclude that PICASO is a robust and promising tracker.

Discussion PICASO has a series of advantages and also limitations, like any other tracker. First we enumerate some advantages. PICASO models the tracked object as a set of pixels (templates) and relative positions to the center, this together with the continuous update of the model and robust matching allows to cope with deformations, corruptions, occlusions in tracking. PICASO rarely loses track as shown on Godec or Babenko benchmarks, and surprisingly, copes the best with scale variation in the 50 videos benchmark. PICASO is very robust, it achieves the best performance for rigid tracking (Babenko videos) but also for non-rigid tracking (Godec videos) while most other trackers (such as HT, TLD, Struck, FCT, KCF) do not cope equally well with both tasks. PICASO is easy to implement, it has a low time complexity, and, as we showed in our experiments, the running time can be easily controlled by setting its parameters, such as the number of used pixels. PICASO uses a normalized working area and runs at comparable speed for most of the provided videos.

The limitations of PICASO are also points for future research and for potential further improvements. PICASO does not adapt to reflect scaling and aspect ratio changes. Our preliminary scale adaptation results were inconclusive. Even so, PICASO handles well scale variation (see Fig. 11). Despite being a top tracking method, PICASO has a rather rigid object model - pixels in a normalized centered coordinate system - and does not employ a track

Supplementary material includes videos and other quantitative results (such as CLE, fps): http://www.vision.ee.ethz.ch/~timotter/PICASO

Tracking Speed The processing speed is important in practice. In Table 3 we directly computed the speed for MTT, FCT, KCF, and our PICASO, while for the other methods we report the values from other independent studies. The FoT speed is from (Vojir and Matas, 2014), that for MSff from (Vojir et al., 2013), for ALIEN from (Pernici and Del Bimbo, 2014), for PixelTrack from (Duffner and Garcia, 2013), and the speeds for Struck, HT and TLD are taken from (Kristan et al., 2013). PICASO, FoT, MSff, PixelTrack, FCT, KCF can reach more than 30 fps on desktop CPUs. However, our PICASO is Matlab implemented, while the other methods use C++. We expect that a C++ implementation of PICASO can be several times faster than the current Matlab implementation. Reading the images, extracting the pixel templates, and INN matching correspond, on average, roughly, to 30%, 20% and 40%, resp., from the processing time. For example, for ‘Panda’, with a model of 230 pixels, an average of 36fps, and CLE=3.5: reading the images takes 36% of the time, 600 pixels extraction -20%, while matching -37%.
recovery procedure. PICASO would benefit from a more elaborate model and such a procedure. Also, it is unclear how well PICASO (and the other top trackers) can cope with occlusions from similarly textured objects.

4. Conclusions

We have introduced the PICASO tracker, a robust visual tracker based on pixel correspondences and soft match selection. We model the object and its context by collections of pixels with their templates and center-related positions. For the matching we proposed techniques based on solving linear decomposition problems with enforced sparsity. We handle occlusions explicitly by means of efficient masks and use median filtering to estimate the object location. PICASO compares favorably against several state-of-the-art methods in terms of tracking accuracy and speed, and this for both rigid and non-rigid object tracking. We also suggested some possible directions for improvements.

Acknowledgements. We thank the anonymous reviewers for their valuable comments. This work was supported by the ERC Advanced Grant VarCity (#273940).

References


<table>
<thead>
<tr>
<th>Attribute</th>
<th>FCT</th>
<th>MT T</th>
<th>LSK</th>
<th>CSR</th>
<th>VTS</th>
<th>VTD</th>
<th>CXT</th>
<th>TLD</th>
<th>ASLA</th>
<th>STRUCK</th>
<th>SCM</th>
<th>KCF</th>
<th>PICASO</th>
<th>PICASO 900</th>
<th>PICASO 1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illumination variation (28)</td>
<td>0.376</td>
<td>0.351</td>
<td>0.449</td>
<td>0.481</td>
<td>0.573</td>
<td>0.573</td>
<td>0.503</td>
<td>0.537</td>
<td>0.517</td>
<td>0.558</td>
<td>0.594</td>
<td>0.600</td>
<td>0.538</td>
<td>0.625</td>
<td>0.662</td>
</tr>
<tr>
<td>Out-of-plane rotation (39)</td>
<td>0.424</td>
<td>0.473</td>
<td>0.525</td>
<td>0.540</td>
<td>0.604</td>
<td>0.620</td>
<td>0.574</td>
<td>0.596</td>
<td>0.518</td>
<td>0.597</td>
<td>0.618</td>
<td>0.710</td>
<td>0.666</td>
<td>0.689</td>
<td>0.733</td>
</tr>
<tr>
<td>Scale variation (28)</td>
<td>0.443</td>
<td>0.461</td>
<td>0.480</td>
<td>0.503</td>
<td>0.582</td>
<td>0.597</td>
<td>0.550</td>
<td>0.605</td>
<td>0.552</td>
<td>0.639</td>
<td>0.672</td>
<td>0.677</td>
<td>0.662</td>
<td>0.681</td>
<td>0.740</td>
</tr>
<tr>
<td>Occlusion (29)</td>
<td>0.368</td>
<td>0.426</td>
<td>0.534</td>
<td>0.502</td>
<td>0.534</td>
<td>0.545</td>
<td>0.491</td>
<td>0.563</td>
<td>0.546</td>
<td>0.564</td>
<td>0.640</td>
<td>0.724</td>
<td>0.658</td>
<td>0.577</td>
<td>0.729</td>
</tr>
<tr>
<td>Deformation (19)</td>
<td>0.451</td>
<td>0.332</td>
<td>0.481</td>
<td>0.476</td>
<td>0.487</td>
<td>0.501</td>
<td>0.422</td>
<td>0.512</td>
<td>0.445</td>
<td>0.521</td>
<td>0.586</td>
<td>0.702</td>
<td>0.623</td>
<td>0.649</td>
<td>0.668</td>
</tr>
<tr>
<td>Motion blur (12)</td>
<td>0.392</td>
<td>0.308</td>
<td>0.324</td>
<td>0.342</td>
<td>0.373</td>
<td>0.375</td>
<td>0.569</td>
<td>0.518</td>
<td>0.272</td>
<td>0.551</td>
<td>0.339</td>
<td>0.569</td>
<td>0.531</td>
<td>0.518</td>
<td>0.619</td>
</tr>
<tr>
<td>Fast motion (17)</td>
<td>0.368</td>
<td>0.401</td>
<td>0.375</td>
<td>0.381</td>
<td>0.353</td>
<td>0.352</td>
<td>0.515</td>
<td>0.551</td>
<td>0.253</td>
<td>0.604</td>
<td>0.333</td>
<td>0.559</td>
<td>0.523</td>
<td>0.523</td>
<td>0.612</td>
</tr>
<tr>
<td>In-plane rotation (31)</td>
<td>0.440</td>
<td>0.522</td>
<td>0.354</td>
<td>0.547</td>
<td>0.579</td>
<td>0.599</td>
<td>0.610</td>
<td>0.584</td>
<td>0.511</td>
<td>0.617</td>
<td>0.597</td>
<td>0.702</td>
<td>0.659</td>
<td>0.674</td>
<td>0.721</td>
</tr>
<tr>
<td>Out of view (6)</td>
<td>0.261</td>
<td>0.374</td>
<td>0.515</td>
<td>0.379</td>
<td>0.455</td>
<td>0.462</td>
<td>0.510</td>
<td>0.576</td>
<td>0.333</td>
<td>0.539</td>
<td>0.429</td>
<td>0.650</td>
<td>0.590</td>
<td>0.560</td>
<td>0.638</td>
</tr>
<tr>
<td>Background clutter (21)</td>
<td>0.441</td>
<td>0.424</td>
<td>0.504</td>
<td>0.505</td>
<td>0.579</td>
<td>0.578</td>
<td>0.571</td>
<td>0.442</td>
<td>0.428</td>
<td>0.496</td>
<td>0.585</td>
<td>0.578</td>
<td>0.753</td>
<td>0.614</td>
<td>0.678</td>
</tr>
<tr>
<td>Low resolution (4)</td>
<td>0.364</td>
<td>0.510</td>
<td>0.304</td>
<td>0.411</td>
<td>0.187</td>
<td>0.168</td>
<td>0.371</td>
<td>0.349</td>
<td>0.156</td>
<td>0.545</td>
<td>0.305</td>
<td>0.381</td>
<td>0.532</td>
<td>0.453</td>
<td>0.515</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Attribute</th>
<th>FCT</th>
<th>MT T</th>
<th>LSK</th>
<th>CSR</th>
<th>VTS</th>
<th>VTD</th>
<th>CXT</th>
<th>TLD</th>
<th>ASLA</th>
<th>STRUCK</th>
<th>SCM</th>
<th>KCF</th>
<th>PICASO</th>
<th>PICASO 900</th>
<th>PICASO 1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Illumination variation (25)</td>
<td>0.359</td>
<td>0.362</td>
<td>0.400</td>
<td>0.386</td>
<td>0.425</td>
<td>0.434</td>
<td>0.418</td>
<td>0.420</td>
<td>0.422</td>
<td>0.432</td>
<td>0.470</td>
<td>0.485</td>
<td>0.455</td>
<td>0.480</td>
<td>0.500</td>
</tr>
<tr>
<td>Out-of-plane rotation (39)</td>
<td>0.314</td>
<td>0.348</td>
<td>0.373</td>
<td>0.350</td>
<td>0.400</td>
<td>0.408</td>
<td>0.389</td>
<td>0.421</td>
<td>0.462</td>
<td>0.518</td>
<td>0.427</td>
<td>0.431</td>
<td>0.451</td>
<td>0.471</td>
<td></td>
</tr>
<tr>
<td>Scale variation (28)</td>
<td>0.308</td>
<td>0.342</td>
<td>0.409</td>
<td>0.385</td>
<td>0.398</td>
<td>0.403</td>
<td>0.372</td>
<td>0.402</td>
<td>0.376</td>
<td>0.413</td>
<td>0.467</td>
<td>0.500</td>
<td>0.458</td>
<td>0.484</td>
<td>0.506</td>
</tr>
<tr>
<td>Occlusion (29)</td>
<td>0.358</td>
<td>0.280</td>
<td>0.373</td>
<td>0.343</td>
<td>0.368</td>
<td>0.377</td>
<td>0.324</td>
<td>0.378</td>
<td>0.372</td>
<td>0.393</td>
<td>0.448</td>
<td>0.513</td>
<td>0.449</td>
<td>0.480</td>
<td>0.492</td>
</tr>
<tr>
<td>Deformation (19)</td>
<td>0.325</td>
<td>0.274</td>
<td>0.302</td>
<td>0.305</td>
<td>0.304</td>
<td>0.309</td>
<td>0.369</td>
<td>0.404</td>
<td>0.258</td>
<td>0.433</td>
<td>0.298</td>
<td>0.463</td>
<td>0.429</td>
<td>0.422</td>
<td>0.489</td>
</tr>
<tr>
<td>Motion blur (12)</td>
<td>0.342</td>
<td>0.333</td>
<td>0.328</td>
<td>0.316</td>
<td>0.300</td>
<td>0.302</td>
<td>0.388</td>
<td>0.417</td>
<td>0.247</td>
<td>0.462</td>
<td>0.296</td>
<td>0.363</td>
<td>0.419</td>
<td>0.432</td>
<td>0.480</td>
</tr>
<tr>
<td>Fast motion (17)</td>
<td>0.345</td>
<td>0.395</td>
<td>0.411</td>
<td>0.399</td>
<td>0.416</td>
<td>0.430</td>
<td>0.452</td>
<td>0.416</td>
<td>0.245</td>
<td>0.444</td>
<td>0.458</td>
<td>0.484</td>
<td>0.467</td>
<td>0.472</td>
<td>0.490</td>
</tr>
<tr>
<td>In-plane rotation (31)</td>
<td>0.325</td>
<td>0.342</td>
<td>0.430</td>
<td>0.349</td>
<td>0.443</td>
<td>0.446</td>
<td>0.427</td>
<td>0.457</td>
<td>0.312</td>
<td>0.459</td>
<td>0.361</td>
<td>0.506</td>
<td>0.492</td>
<td>0.494</td>
<td>0.535</td>
</tr>
<tr>
<td>Out of view (6)</td>
<td>0.363</td>
<td>0.337</td>
<td>0.388</td>
<td>0.421</td>
<td>0.428</td>
<td>0.425</td>
<td>0.338</td>
<td>0.345</td>
<td>0.408</td>
<td>0.458</td>
<td>0.450</td>
<td>0.535</td>
<td>0.462</td>
<td>0.506</td>
<td>0.535</td>
</tr>
<tr>
<td>Background clutter (21)</td>
<td>0.289</td>
<td>0.389</td>
<td>0.235</td>
<td>0.350</td>
<td>0.166</td>
<td>0.177</td>
<td>0.312</td>
<td>0.309</td>
<td>0.157</td>
<td>0.372</td>
<td>0.279</td>
<td>0.312</td>
<td>0.349</td>
<td>0.336</td>
<td>0.347</td>
</tr>
<tr>
<td>Low resolution (4)</td>
<td>0.364</td>
<td>0.510</td>
<td>0.304</td>
<td>0.411</td>
<td>0.187</td>
<td>0.168</td>
<td>0.371</td>
<td>0.349</td>
<td>0.156</td>
<td>0.545</td>
<td>0.305</td>
<td>0.381</td>
<td>0.532</td>
<td>0.453</td>
<td>0.515</td>
</tr>
</tbody>
</table>

**Table 4: Attribute-based performance in terms of average precision.** We indicate the best, second best and third best results. We evaluated 12 recent tracking methods on 50 test sequences and report results of the top methods and ours. In the brackets is the number of sequences for each attribute.

**Table 5: Attribute-based performance in terms of average success rate.** We indicate the best, second best and third best results, as in Table 4.


