Learning Features for Tracking [*]

Michael Grabner Helmut Grabner Horst Bischof
Institute for Computer Graphics and Vision
Graz University of Technology
{mgrabner, hgrabner, bischof}@icg.tugraz.at

Abstract

We treat tracking as a matching problem of detected keypoints between successive frames. The novelty of this paper is to learn classifier-based keypoint descriptions allowing to incorporate background information. Contrary to existing approaches, we are able to start tracking of the object from scratch requiring no off-line training phase before tracking. The tracker is initialized by a region of interest in the first frame. Afterwards an on-line boosting technique is used for learning descriptions of detected keypoints lying within the region of interest. New frames provide new samples for updating the classifiers which increases their stability. A simple mechanism incorporates temporal information for selecting stable features. In order to ensure correct updates a verification step based on estimating homographies using RANSAC is performed. The approach can be used for real-time applications since on-line updating and evaluating classifiers can be done efficiently.

1. Introduction

Despite the huge amount of work spend on tracking research it is still a challenge to design robust tracking methods that can cope with all variations that occur in natural scenes such as shape and appearance changes, illumination variations, pose or partial occlusion of the target object. Moreover, we require that the tracking algorithms can cope with the speed of the tracked object, therefore efficiency is a topic of significant importance.

For tracking many different methods have been proposed, from global template-based trackers [1, 2, 4, 5, 10, 14, 17, 22, 27, 28] to local feature-based trackers [7, 15, 19, 20, 23, 25]. Global trackers consider the template as a whole (*e.g.* tracking target is localized by a bounding box) and are able to handle complex patterns. However these approaches have typically problems with more

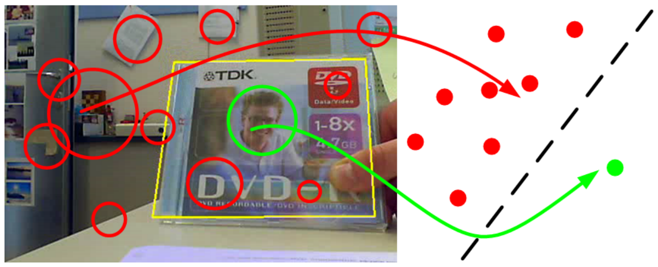


Figure 1. Learning a classifier for feature representation allows to simplify the problem of matching keypoints since the corresponding point has just to be distinguished from to currently detected ones. The usage of an on-line classifier allows to build this representation during tracking by collecting samples over time.

complex transformations of the target object or with appearance changes such as occlusion. In contrast feature-based trackers, can be further divided into the two categories of short-baseline feature tracking, such as the well known KLT approach [25], and approaches proposing wide-baseline feature matching [15, 20, 23] which are also known as tracking-by-detection.

Recently for both, global as well as for feature-based methods, we have seen approaches handling tracking as a classification problem [1, 2, 9, 15]. In case of global methods the idea is to simplify the tracking problem by incorporating background information for selecting a discriminative space to distinguish object and background. As a result tracking is usually simplified by formulating it as a classification task between foreground (target object) and background (current scene). An efficient feature-based tracker based on classification has been recently proposed by Lepetit *et al.* [15]. Their key insight is to use a training phase where a lot of different appearance changes of the target object are presented and a tree-based classifier is trained off-line to discriminate keypoints from each other. A considerable amount of time is spend in the off-line training phase in order to be fast at the on-line tracking phase which simply needs to evaluate the classifier in order to establish the correspondence.

An extension of this approach is presented in [23] where

[*]This work has been sponsored by the Austrian Joint Research Project Cognitive Vision under projects S9103-N04 and S9104-N04, the EC funded NOE MUSCLE IST 507572.

a tree-based classifier is learned from an image sequence. The learned classifier is used to establish matches between frames. In addition, learning appearance and geometry in the training phase makes this approach of on-line training the classifier cumbersome. Another approach for learning local features out of sequences has been proposed by Meltzer *et al.* [20]. They suggest to use Kernel PCA (KPCA) to incorporate information from short baseline matching in order to develop more powerful descriptors for wide-baseline matching. Thereby variations of appearance of local features are learned using KPCA. However for collecting samples to learn descriptors with KPCA a short-baseline feature tracker such as KLT is needed, which assumes small displacement of a detected keypoint.

In this paper we propose a local corner-based tracking approach based on classifying local features (see Figure 1). The novelty of our approach compared to others, is that we do not require an initial off-line training phase. Given some initial region of the target object in the first frame, we are able to on-line learn classifiers for describing object interest points which are used to establish correct matches of keypoints between frames. For verification of the matches, which is important for obtaining correct labels for updating, the homograhy using RANSAC is estimated between successive frames. As new frames arrive, these classifiers are updated by taking matches as positive examples and all other keypoints as negative examples. For on-line training we use a recently proposed on-line boosting algorithm [8]. Since the problem of discriminating one keypoint from the others in the current frame is simple, the complexity of the classifiers can be low. In addition, we are using efficient data structures for feature computation, therefore the tracker is applicable to real-time tasks.

The reminder of the paper is organized as follows. In Section 2 we formulate the keypoint matching problem through discriminative on-line learning of local features. This is followed by Section 3 where keypoint matching is applied to the object tracking problem. In Section 4 we illustrate experimental results of the approach.

2. On-line Classifiers for Keypoint Matching

In [16] the problem of matching keypoints via classification is formulated as follows. Given a set $K = \{k_1, k_2, ..., k_N\}$ of N keypoints for training a classifier Q the goal is to assign to a new input patch $\mathbf{p}$ a class label $Q(\mathbf{p}) \in Y = \{-1, 1, 2, ..., N\}$, where -1 denotes a keypoint not belonging to the training set. The aim is to construct a classifier $\hat{Q}$ such that $P(Q(\mathbf{p}) \neq \hat{Q}(\mathbf{p}))$ is small (P denotes the probability).

Different to Lepetit *et al.*, the idea is to formulate tracking as a classification problem between successive frames by establishing matches over successive frames. This allows us to assume that patches around keypoints have only slight appearance changes. Therefore we propose to learn distance functions in the space of keypoints $d : K \times K \rightarrow [0, 1]$. Similar to [13], boosting can be used to learn a classifier which is then equivalent to learning a distance function. In other words we implicitly define a multiclass classification problem in the original vector space K by generating a binary partition of the data in the product space $K \times K$. We choose the simple one vs. all partition, *i.e.* each keypoint can be distinguished from all the other points. Thus, we end up with a set of classifiers $C = \{C_1, C_2, ..., C_N\}$ each corresponding to a keypoint $\{k_1, k_2, ..., k_N\}$. For making this partitioning more efficient, one can reduce the number of classifiers using a more compact representation like error correcting output codes[1] [6].

As a result, our goal is to find a set of classifiers C such that $P(C_i(\mathbf{p}) = 1)$ is large for a correct match and in case of a mismatch $P(C_i(\mathbf{p}) = 1)$ remains small.

2.1. On-line Classifier

For learning such classifiers as described in the previous section we need a maximum margin classifier. Boosting satisfies this requirement [24]. Since we are dealing with tracking of the object we switch to the on-line variant.

In the following we briefly review the on-line boosting algorithm (for more details see [8, 9]) which allows to generate classifiers that can be efficiently updated by incrementally applying samples. On-line boosting introduces so called selectors h^{sel}, each of them holding a weak classifier pool $\mathcal{H} = \{h_1, ..., h_M\}$. At a certain time instance t the on-line classifier C_i is formed by a linear combination of J selectors h_j^{sel} which reference to one of the weak classifiers h_i from its feature pool,

$$C_i(\mathbf{p}) := H(\mathbf{p}) = \text{sign}(conf(\mathbf{p})) \quad (1)$$

$$conf(\mathbf{p}) = \sum_{j=1}^{J} \alpha_j \cdot h_j^{sel}(\mathbf{p}) \quad (2)$$

where the value $conf(\cdot)$ (which is related to the margin) can be interpreted as a confidence measure of the strong classifier $C_i(\mathbf{p})$. As a new sample arrives for update, boosting is performed on the selectors and not directly on the weak classifiers. Each weak classifier h_i is updated with the corresponding label of the sample. The best weak classifier (having the lowest error) is selected by the selector $h_j^{sel} = h_m,\ h_m \in \mathcal{H},\ m = \arg\min_i e_i$ where the error e_i of the weak classifier h_i is estimated from samples seen so far. Handling over the updated importance weight to the next selector, the selection process of weak classifiers for

[1]We use the simple one vs. all partitioning because this leads to a simple strategy for the dynamic feature exchange (see Section 3.1) since we can directly remove a learned feature since the classifiers are independent.

all selectors can be done. The complexity of the classifiers can be adjusted by the number of selectors.

Since updating all the weak classifiers requires most of the time, we use a single global weak classifier pool which is shared by all selectors as proposed in [9]. This modification brings an significant speed-up which allows in combination with efficient computed hypotheses for weak classifiers to use this approach for real-time applications.

2.2. Classifier-based Keypoint Description

In contrast to methods using a fixed metric for keypoint description (*e.g.* shape descriptors [3] or SIFT [18]), discriminative learning of keypoint descriptions allows to incorporate scene information and therefore simplifies the tracking problem to a classification problem among the currently detected scene keypoints. The descriptor is formed by a linear combination of weighted weak classifiers each corresponding to an axis of the subspace, see Figure 2, which has been chosen by the boosting procedure. Note that each classifier might consist of different weak classifiers resulting in different subspaces. As a result, by discriminative learning we determine a subspace spanned by the weak classifiers, in which the corresponding patch can be best discriminated from the others. In comparison, the usage of a fixed metric for describing a patch would always project the patch into the same predefined subspace.

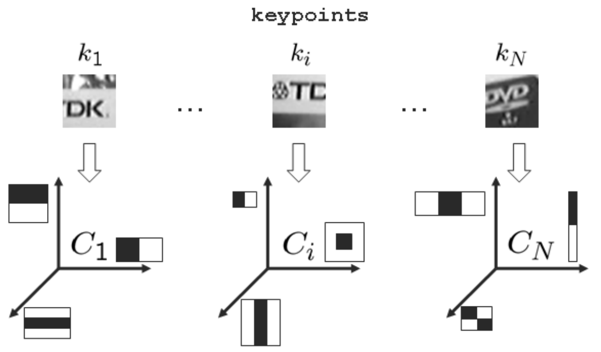


Figure 2. Each classifier C_i corresponds to a keypoint k_i which is trained using the specific keypoint for a positive update and any other keypoints for negative examples. Thus, a distance funtion $d : K \times K \rightarrow [0, 1]$ is learned for each keypoint.

For learning a description of a keypoint k_i, we first initialize a new classifier C^{new} and apply its surrounding region which is covered by patch $\mathbf{p_i}$ for a positive update $C_i^+(\mathbf{p_i})$. Any other keypoint k_j, where $j \neq i$, can be used for a negative update $C_i^-(\mathbf{p_j})$. Note that updating classifiers can cause a change in the selection of weak classifiers of the selectors meaning that the subspace changes. This means the classifiers are capable to adapt to changes in the current classification problem.

3. Object Tracking

The major idea in using the aforementioned classifier-based keypoint description for object tracking is to incorporate background information. This has successfully been done for template-based trackers [2, 5, 9, 22, 27] but is novel for feature-based trackers. Classifiers are used in order to learn descriptions of keypoints lying on the object. However, since we only want to track the specific object within the current scene, we only have to discriminate the current object keypoints from the once detected in the background. The usage of on-line classifiers allows to collect samples over time for improving generalization and in addition to adapt keypoint descriptions even if the scene changes. As a result discriminative classifiers allow to incorporate scene information by considering them as negative samples for the keypoint descriptions.

The only required initialization is the definition of the object region in the first frame $t = 1$. Given the keypoints K_1 by some detector (*e.g.* Harris corners [11] or DoG points [18]), we can now separate them into object keypoints O_1 and points B_1 lying on the background such that $O_1 \cap B_1 = \emptyset$ and $O_1 \cup B_1 = K_1$. Depending on the number of classifiers we randomly choose a subset of O_1 of size N for extracting local features. Each patch $\mathbf{p_{i,1}}$ extracted from keypoint k_i is applied for a positive update of classifier $C_i,\ i = 1, ..., N$. Patches for negative updates are randomly chosen from all other keypoints k_j such that $i \neq j$. Now assuming the target object has been successfully tracked up to time instance $t-1$. In order to detect the object at time t we perform the following steps (see also Algorithm 1). First the set of keypoints K_t are extracted from the current frame. Second we establish possible matches $M = \{m_1, m_2, ..., m_N\}$ by finding for each classifier C_i its best possible match of K_t, computing

$$m_i = \underset{k_j \in K_t}{\operatorname{argmax}}\, C_i(\mathbf{p_{j,t}}). \tag{3}$$

Afterwards a verification step for the proposed matches is needed to obtain labels for correctly updating the current set of classifiers. This is done by robust estimation of the homography H using RANSAC [12] over the set of suggested matches[2]. Thus we achieve a subset of correct matches $M^c \subseteq M$ verified by the homography. In case the number of inliers exceeds a threshold, we assume to have correctly determined the homography H and successfully tracked the object. Therefore we can compute for each classifier its corresponding patch in the actual frame which is then used for

[2]Establishing a homography assumes that the tracked object is planar or the depth discontinuities are small compared to the distance to the camera, if this assumptions are violated we need either to assume multiple homographies for piece-wise planar objects or we can use the fundamental matrix for verification, since this does not change the argumentation of the paper we stick to the planarity assumption.

making a positive update of the classifier C_i. For negative updates we again choose patches extracted from any other keypoint $k \in K_t$. If we can not establish the homography H of the object between two consecutive frames we apply no updates to the classifiers and no detection is achieved.

Algorithm 1 Object Tracking via Keypoint Matching

```
Require: classifier set $C = \{C_1, C_2, ..., C_N\}$ trained for $N$ object keypoints up to time $t$
Require: functions for creating $C^{new}$ and updating $C_i^+(\mathbf{p})$ and $C_i^-(\mathbf{p})$ the $i$-th classifier with an image patch $\mathbf{p}$
Require: function $detectKeypoints$
Require: function $estimateHomography$

  $K_t = detectKeypoints()$
  for $i = 1, 2, ..., N$ do
      $m_i = \operatorname*{argmax}_{k_j \in K_t} C_i(\mathbf{p_{j,t}})$
  end for

  $H = estimateHomography(M)$
  $M^c \subseteq M; \quad O_t \subseteq K_t$

  for $i = 1, 2, ...N$ do
    if $m_i \in M^c$ then
      $C_i^+(\mathbf{p_{m_i,t}});\ C_i^-(\mathbf{p_{j,t}}),\ j \neq m_i$
    end if

    $P_{i,t+1} = \beta \cdot P_{i,t} + (1-\beta) \cdot \delta_i,\ \delta_i = \begin{cases} 1 & m_i \in M^c \\ 0 & else \end{cases}$
    if $P_{i,t+1} < \theta$ then
      $C_i = C^{new}$
      $C_i^+(\mathbf{p_{i,t}});\ C_i^-(\mathbf{p_{j,t}}),\ j \neq i$
      $P_{i,t+1} = 0.5$
    end if

  end for
```

3.1. Finding Stable Local Features

During tracking we want to use the most reliable features of the object. Therefore we introduce a mechanism which on-line determines a ranking of the currently used features by estimating for each classifier the probability for a match in the next frame. Since we have independent classifiers for local features, this measure can be used for discarding bad features and replacing them with new ones which probably might be better suited for matching. In addition the selection of stable features over time allows us to learn only a subset of keypoints lying on the object.

For determining the quality of local features we take the idea of measuring the quality of interest point detectors and local feature descriptors (see [21]). First, there is the repeatability measure which is an indicator of how probable an interest point operator detects points repeatably and second usually the matching score is an indicator for the performance of the used descriptor. Exactly these two properties are used for deriving a probability measure over time. We make a fading memory over the matches. Having a correct match in the near past is more important than one in the far past. Therefore the probability for a feature to be matched in the next frame is estimated by

$$P_{i,t+1} = \beta \cdot P_{i,t} + (1 - \beta) \cdot \delta_i \qquad (4)$$

where $P_{i,t+1}$ is the probability that the classifier C_i will match in the next frame and $\beta \in [0, 1]$ determines the weighting of the past estimations and $\delta_i \in \{0, 1\}$ determines if C_i has matched in the current frame. As a result for each classifier we receive the probability that it matches in the next frame. If the probability $P_{i,t+1}$ is below a threshold θ we exchange the keypoint and initialize a new classifier for a random selected keypoint from the object keypoints O_t. The benefit of this exchange is twofold. On the one hand, in order to track quick appearance changes it is better to initialize a new classifier, since the adaption of a new classifier is much faster than adapting the existing one. On the other hand, if a classifier is not performing well (because of a bad local feature), it is detected and replaced by a new one.

4. Experiments

All experiments run (in a non optimized version) in about 12 fps on a standard 2.0 GHz PC with 1 GB RAM on an image size of 640×480. However the image size is not that critical, since the image information to process is essentially reduced by the number of keypoints. For the detection of local features the Harris corner detector [11] has been used. However note that any other interest point operator can be applied for detecting local features. For the description of objects we trained 60 classifiers capturing a 30×30 pixel neighbourhood of the corresponding keypoint. As weak classifiers for the on-line classifier we used simple Haar-like features [26]. For further details on hypothesis generation of the weak classifiers see [8]. The computation of these features can be done very efficiently using integral images. This allows us to do exhaustive matching while tracking is still possible in real-time. For each keypoint a classifier consisting of 20 selectors, each of them is allowed to choose from the global weak classifier pool containing 250 weak hypotheses.

The rest of the experiment section is organized as follows. First, we show the benefit of on-line updating and second the discriminative strength of our classifier-based keypoint descriptors. Finally we have depicted tracking sequences for illustrating further results.

4.1. Advantage of On-line Learning

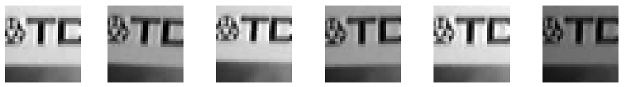
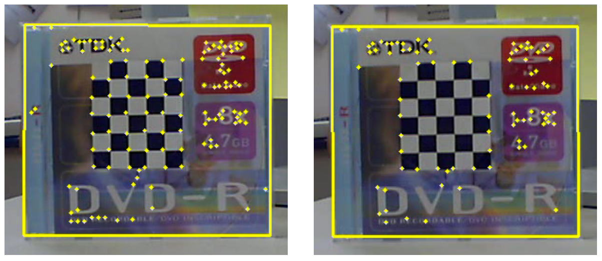


Figure 3. Patches which are collected during tracking. This samples are used for positive updating the corresponding classifier while negative updates are made by randomly selecting patches from any other keypoint.

On-line learning of local features allows us to simplify the classification problem in an elegant way. On the one hand an on-line classifier allows us to exploit information obtained over time by applying correct matches as positive updates to the corresponding classifier. A sample trajectory can be seen in Figure 3. In addition a huge amount of negative samples are available at each time instance since one classifier describes one patch. As a result we obtain classifiers which can exploit variations during tracking for having more reliable descriptions of features.

The benefit of exchanging features is illustrated by two experiments. In the first one the idea is to show the selection of stable features. We put a grid on the target object having a regular structure. Initially, as can be seen in Figure 4, features are detected on this pattern (left). After a few



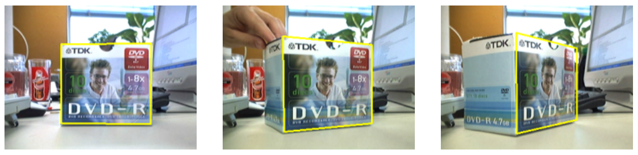
(a) Frame 1 (b) Frame 19

Figure 4. Keypoints which can not be reliably matched are detected over time and discarded.

frames, keypoints at corners of the grid structure have been discarded (right) and are not used for matching anymore. In other words, the mechanism for selecting suitable features over time has decided to discard these keypoints since they are too similar for reliably matching them.

For the second experiment we consider the number of correct matches over time for a certain target object (see Figure 5). After changing the pose of the object, the number of matches decreases. Classifiers are exchanged as well as updated and the tracker stabilizes again.

Table 1 depicts the contribution on the overall tracking performance of on-line updating on the one hand and the effect of on-line exchanging features on the other hand. Sequence of about 1000 frames has been recorded to evaluate the contribution of each mechanism by counting the average number of matches. As depicted in the table we can see that preventing both, the average number of matches is equal to zero, meaning tracking fails totally. When either on-line updating or switching on feature exchange we achieve a similar average matching performance for both. Allowing both we see that they complement one another and the matching accuracy can be increased. To summarize, the dominant mechanism depends on the movement of the object. The on-line updating has the ability to make use of the collected matches in order to make its classifier more stable. As a result this mechanism is especially helpful for handling appearance changes of the patches to be classified. However larger changes which are not handled by invariances of the keypoint detector (*e.g.* for DoG points: is invariant to scale changes, rotation - for Harris: does not determine scale or rotation) are handled by exchanging features.



(a) Frame 1 (b) Frame 50 (c) Frame 100

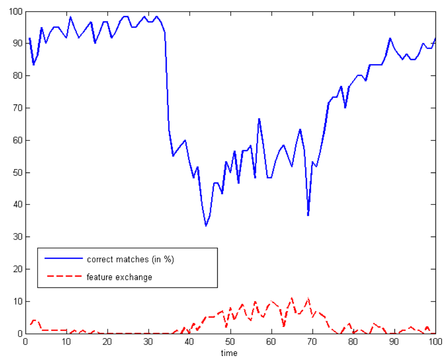
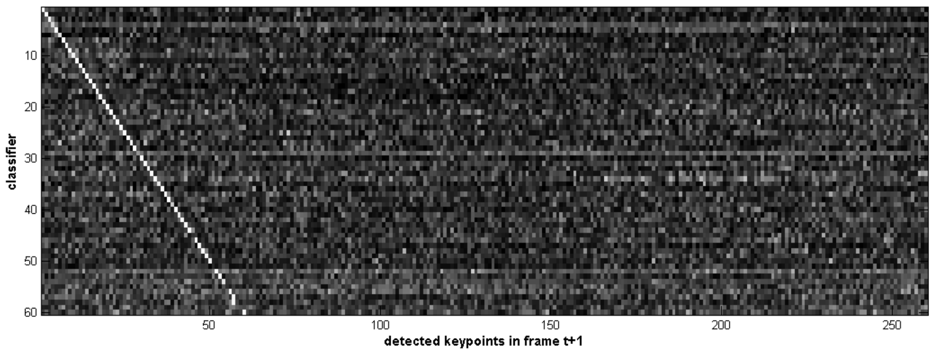


Figure 5. A pose change of the target object causes a decrease of the number of matches. If the appearance change of the object is not too large such that the object does not get lost, the mechanism for exchanging local features allows to find reliable features for matching again.

classifier update	off	on	off	on
feature exchange	off	off	on	on
matching	$\sim 0\%$	51%	62%	77%

Table 1. The average number of matches on an ordinary tracking sequence is analyzed. Allowing on-line updating of classifiers and the contribution of exchanging features shows that both complement one another.



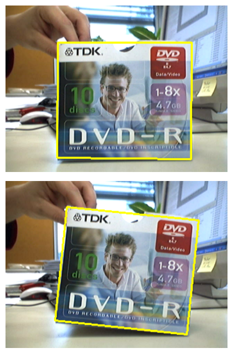


Figure 6. Confusion matrix for two successive frames (right). The dominant diagonal shows that 60 classes can be accurately distinguished from all other detected patches. Since each learned keypoint is matched to all newly detected points the approach is invariant to in-plane movements.

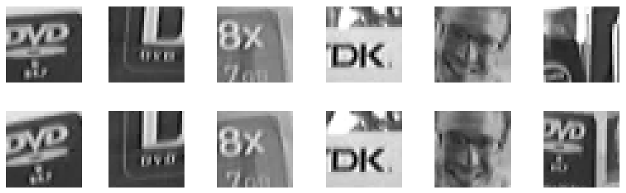


Figure 7. Examples of correct (column 1-5) and incorrect matches (column 6).

4.2. Discriminative Power

To show the strength of discriminative learning of local features we analyze the matching of two subsequent frames, see Figure 6. The observed image pair contains a movement of the target object which can be handled without any problems since the appearance does not vary much. We obtained 54 correct matches from 60 possible matches. In addition we generated a confusion matrix, containing the confidences for all classifiers over all newly detected features. Note that we have sorted the columns by searching for the projected keypoints (which are described by a classifier) their nearest neighbour and to put evaluation results in the corresponding column. As a result we obtain a matrix where the diagonal is dominant. It demonstrates that nearly each of the classifiers has a strong peak for the correct match while the confidence values of non-matching keypoints are low. Samples of correct and incorrect matches during tracking can be seen in Figure 7.

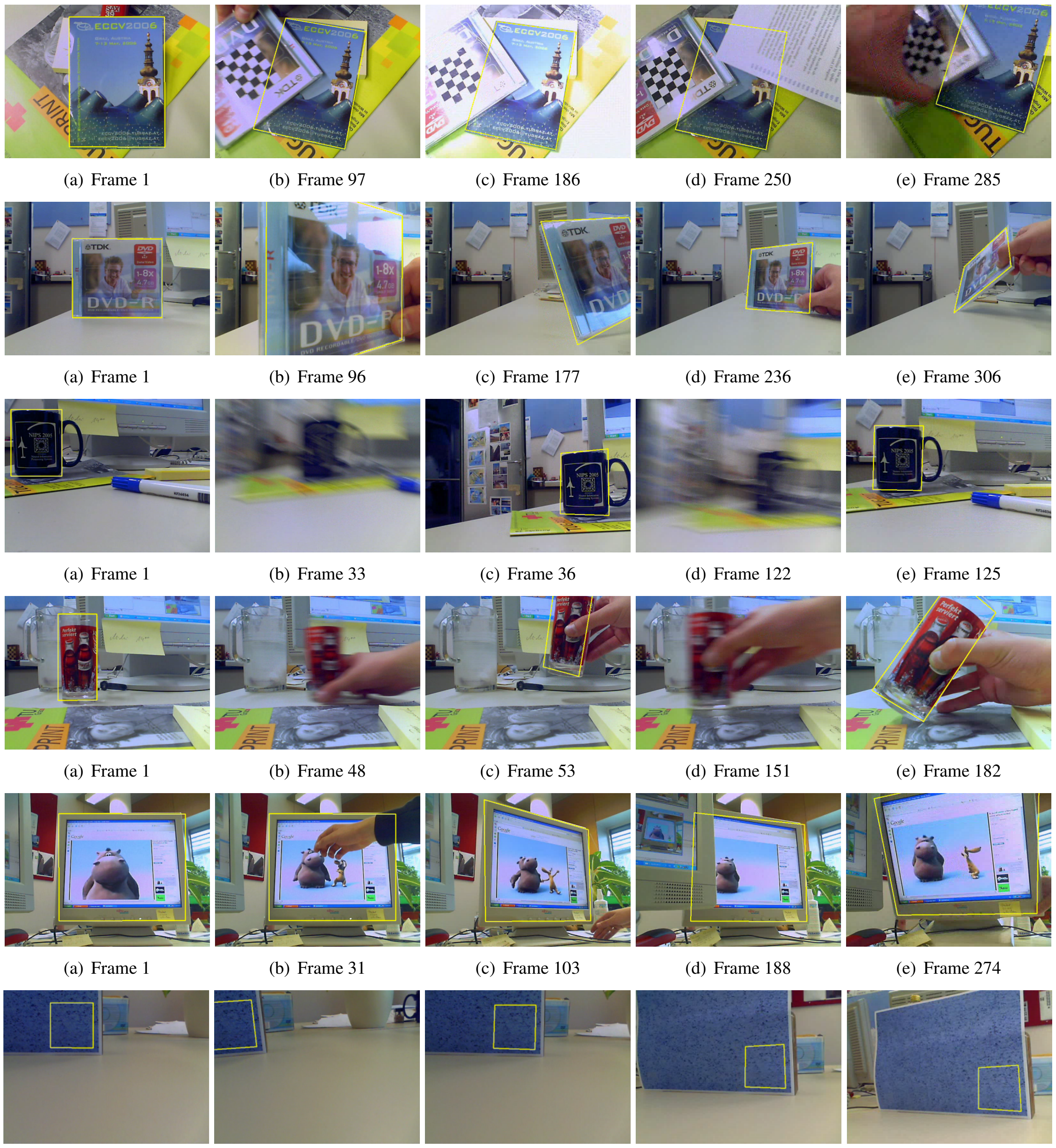
4.3. Sequences

Different objects have been chosen in order to illustrate the tracking performance, shown in Figure 8 (videos are available on the web [3]). The first row shows the property of local approaches of being invariant to occlusions. In addition, illumination changes and affine transformations of the target object do not confuse the tracker. It can even handle large scale and pose changes of the target object as illustrated in row 2 due to the feature exchange property. Fast movement of the target object can cause a loss of the target object, as shown in the next sequences. However, once the appearance is similar to the one before it has been lost, the tracker can re-detect the object and continue tracking. Row 5 illustrates that even targets with a certain amount of changing content can be successfully tracked. In this case we have simulated a changing texture by tracking a monitor that is playing a video. The last sequence shows the benefit of discriminative feature learning. A textured target is tracked even though the same texture is present in the background.

[3]`www.icg.tugraz.at/Members/mgrabner/iptracking`

5. Conclusion

In this paper we have proposed a technique treating the object tracking problem as a matching problem of keypoints. In contrast to existing approaches the novelty is that local features are learned on-line and therefore no offline training of classifiers prior to tracking is needed. The tracking problem is simplified by the fact that for classification only the currently detected interest points have to be considered. Instead of using a predefined metric for obtaining a description the idea is to apply a discriminative learning technique for distinguishing keypoints and thereby benefiting from the simplification of the tracking problem. A mechanism has been presented which allows to incorporate temporal information for selecting stable keypoints over time. In addition the usage of on-line classifiers allows to collect samples as new frames arrive and therefore become more and more stable over time. Updating and evaluating these classifiers is not time consuming because of the usage of efficient features in combination with integral data structures. This allows to use this tracking approach within real-time applications.



(a) Frame 1 (b) Frame 97 (c) Frame 186 (d) Frame 250 (e) Frame 285

(a) Frame 1 (b) Frame 96 (c) Frame 177 (d) Frame 236 (e) Frame 306

(a) Frame 1 (b) Frame 33 (c) Frame 36 (d) Frame 122 (e) Frame 125

(a) Frame 1 (b) Frame 48 (c) Frame 53 (d) Frame 151 (e) Frame 182

(a) Frame 1 (b) Frame 31 (c) Frame 103 (d) Frame 188 (e) Frame 274

(a) Frame 1 (b) Frame 31 (c) Frame 44 (d) Frame 171 (e) Frame 202

Figure 8. Sample sequences.

References

[1] S. Avidan. Support vector tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26:1064–1072, 2004.

[2] S. Avidan. Ensemble tracking. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 494–501, 2005.

[3] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transaction on Pattern Analysis Machine Intelligence*, 24(4):509–522, 2002.

[4] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a

view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[5] R. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1631–1643, 2005.

[6] T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

[7] P. Gabriel, J.-B. Hayet, J. Piater, and J. Verly. Object tracking using color interest points. In *IEEE Conference on Advanced Video and Signal Based Surveillance*, pages 159–164, 2005.

[8] H. Grabner and H. Bischof. On-line boosting and vision. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 260–267, 2006.

[9] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *Proceedings 17th British Machine Vision Conference*, volume 1, pages 47–56, 2006.

[10] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, 1998.

[11] C. Harris and M. Stephens. A combined corner and edge detection. In *Proceedings of The Fourth Alvey Vision Conference*, pages 147–151, 1988.

[12] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[13] T. Hertz, A. Bar-Hillel, and D. Weinshall. Boosting margin based distance functions for clustering. In *International Conference on Machine Learning*, page 50, 2004.

[14] A. D. Jepson, D. J. Fleet, and T. El-Maraghi. Robust online appearance models for visual tracking. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 415 – 422, 2001.

[15] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 775 – 781, 2005.

[16] V. Lepetit, J. Pilet, and P. Fua. Point matching as a classification problem for fast and robust object pose estimation. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 244–250, 2004.

[17] J. Lim, D. Ross, R. Lin, and M. Yang. Incremental learning for visual tracking. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Neural Information Processing Systems*, number 17, pages 793–800. 2005.

[18] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[19] L. Masson, M. Dhome, and F. Jurie. Robust real time tracking of 3d objects. In *Proceedings of 17th International Conference on Pattern Recognition*, pages 252–255, 2004.

[20] J. Meltzer, M.-H. Yang, R. Gupta, and S. Soatto. Multiple view feature descriptors from image sequences via kernel principal component analysis. In *Proceedings 8th European Conference on Computer Vision*, pages 215–227, 2004.

[21] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[22] H. Nguyen and A. Smeulders. Tracking aspects of the foreground against the background. In *Proceedings 8th European Conference on Computer Vision*, volume 2, pages 446 – 456, 2004.

[23] M. Özuysal, V. Lepetit, F. Fleuret, and P. Fua. Feature harvesting for tracking-by-detection. In *Proceedings 10th European Conference on Computer Vision*, volume 3, pages 592–605, 2006.

[24] R. Schapire, Y. Freund, P. Bartlett, and W. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. In *Proceedings International Conference on Machine Learning*, pages 322–330, 1997.

[25] J. Shi and C. Tomasi. Good features to track. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, pages 593 – 600, 1994.

[26] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.

[27] J. Wang, X. Chen, and W. Gao. Online selecting discriminative tracking features using particle filter. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 1037 – 1042, 2005.

[28] O. Williams, A. Blake, and R. Cipolla. Sparse bayesian learning for efficient visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1292–1304, 2005.