On-line Adaption of Class-specific Codebooks for Instance Tracking

Juergen Gall[1]
gall@vision.ee.ethz.ch
Nima Razavi[1]
nrazavi@vision.ee.ethz.ch
Luc Van Gool[1,2]
vangool@vision.ee.ethz.ch

[1] Computer Vision Laboratory
ETH Zurich, Switzerland
[2] IBBT, ESAT-PSI
K.U. Leuven, Belgium

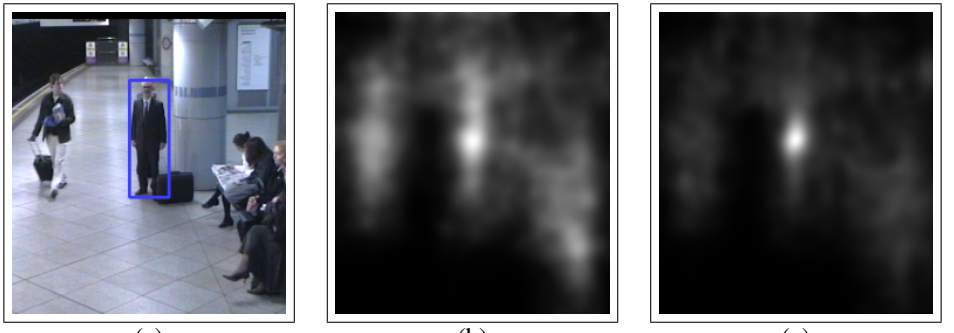
Abstract

Off-line trained class-specific object detectors are designed to detect any instance of the class in a given image or video sequence. In the context of object tracking, however, one seeks the location and scale of a target object, which is a specific instance of the class. Hence, the target needs to be separated not only from the background but also from other instances in the video sequence. We address this problem by adapting a class-specific object detector to the target, making it more instance-specific. To this end, we learn off-line a codebook for the object class that models the spatial distribution and appearance of object parts. For tracking, the codebook is coupled with a particle filter. While the posterior probability of the location and scale of the target is used to learn on-line the probability of each part in the codebook belonging to the target, the probabilistic votes for the object cast by the codebook entries are used to model the likelihood.

1 Introduction

Tracking an object in a monocular video sequence is very often a required preprocessing step for higher level video analysis and thus relevant for many practical applications. The most popular approaches rely on mean shift [11], Kalman [34], or particle filtering [18] where the appearance of the object is modeled by manually selected or learned image features. The appearance model is usually updated over time to cope with changing lighting conditions, background changes, and object transformations in the 3D space. Although state-of-the-art approaches as [19, 27] perform well in certain scenarios, the update of the appearance can lead to a significant drift, *i.e*. the appearance adapts to another object or background.

Fortunately, nearly all practical applications require the tracking of a restricted class of objects, *e.g*. humans or faces. This additional information has been well exploited by so-called *tracking-by-detection* approaches. Such approaches rely on a class-specific detector that is applied independently to all frames. The detections are then assembled to tracks, *e.g*. by using the detections or the confidence of the detector as observation for a particle filter. An off-line trained detector, however, tries to solve a much more difficult task than object tracking, namely to identify any instance of the class in any image. Since there is not a perfect object detector, the detectors provide even high confidence for parts of the



(a) (b) (c)

Figure 1: In order to track an instance of a class, like a certain person from the class pedestrians, we adapt on-line a class-specific codebook to the instance. (a) Blue box indicates the instance of interest. (b) Voting image obtained by an off-line trained codebook for pedestrians. (c) Voting image obtained by the proposed instance-specific codebook. The peak at the center of the instance is clearly visible and votes for background objects are reduced.

background, *i.e*. false positives, and cannot distinguish between several instances of the same class. A straightforward workaround is the combination of an off-line trained class-specific detector with an on-line learned identifier that distinguishes the target from other instances or from background objects with a high detector confidence. This, however, does not address the core of the problem, namely that detectors are not designed for tracking.

In this work, we demonstrate that an off-line trained class-specific detector can be transformed into an instance-specific detector on-the-fly. To this end, we make use of a codebook-based detector [14] that is trained on an object class. Codebooks model the spatial distribution and appearance of object parts. When matching an image against a codebook, a certain set of codebook entries is activated to cast probabilistic votes for the object. For a given object hypothesis, one can collect the entries that voted for the object. In our case, these entries can be regarded as a signature for the target of interest. Since a change of pose and appearance can lead to an activation of very different codebook entries, we learn the statistics for the target and the background over time, *i.e*. we learn on-line the probability of each part in the codebook belonging to the target. By taking the target-specific statistics into account for voting, the target can be distinguished from other instances in the background yielding a higher detection confidence for the target, see Fig. 1. To cope with the multi-modal probability for the location of the object, we couple the codebook with a particle filter.

2 Related Work

Codebook-based detectors learn the mapping from image features into a Hough space where detection hypotheses are obtained by local maxima. To this end, a codebook of local appearance is trained by clustering a training set of image features and storing their relative location with respect to the object center. While [21] clusters the sparse image features only based on appearance, the spatial distribution of the image features is used as cue for the clustering in [25, 28]. In [24], a max-margin framework is proposed to re-weight the votes for a better detection accuracy. Hough forests [14, 26] use a random forest framework [8] instead of clustering for codebook creation. Random forests have also been used for real-time tracking [22] where the forest is trained on a single target object. The approach, however, is

object-specific and does not generalize to other objects.

One benefit of codebook detectors is the robustness to occlusions. Since only a small set of local patches is required to locate the object, the detection is still reliable when the object is partially occluded. The idea of voting has been exploited for tracking in [1] where the template of the object is represented by a set of local patches. Each patch is tracked independently and the patches vote for the center of the object. While [1] use a static template model, it has been shown that updating the template is necessary in many scenarios [19, 27].

Modeling the appearance of an object has been addressed in several works. Most relevant are the *tracking-by-detection* approaches that train a model to separate the object from the background via a discriminative classifier. The approaches mainly differ in the classifier and the update strategy. For instance, an adaptive ensemble of classifiers based on support vector machines [3], adaptive feature selection [5, 10, 32, 33], incremental subspace learning [27], on-line boosting [16], or multiple instance learning [4] have been proposed. A combination of several classifiers has been used in [9, 23, 29, 30, 31, 35]. The classifiers have different properties such that the weakness of one is compensated by the others, *e.g.* by using off-line and on-line trained classifiers or by training the classifiers on different features. The fusion, however, is often performed in a non-probabilistic manner using a weighted sum or taking the maximum of the confidences. Our approach differs from these approaches in that it does not combine several binary classifiers but adapts a single class-specific codebook in a probabilistic manner.

3 Tracking with an Instance-specific Codebook

After a brief description of the off-line learning of the codebook (Section 3.1), we present a new approach to convert it into an instance-specific codebook (Section 3.2). The tracking framework with the on-line adaption is described in Section 3.3.

3.1 Class-specific Codebook

The class-specific codebook is implemented and trained as in [14]. For training, a set of images containing the object class, *e.g.* pedestrians, with bounding box annotation and a set of background images are required. After scaling the positive examples to a unit scale, *i.e.* $\max\{width, height\} = 100$, the Hough forest samples randomly 16x16 patches from the bounding boxes and negative examples. The training patches are then split in a tree structure such that each leaf L contains a set of positive patches $\mathscr{P}_L = \{P_i\}$ that are similar in appearance. Furthermore, the probability of the patches belonging to the object class $p\big(c{=}1|L\big)$ and the local spatial distribution of the patches with respect to the object center $p\big(\mathbf{x}|c{=}1,L\big)$ are stored. For detection, patches are sampled from an image and matched against the codebook, *i.e.* each patch $P(\mathbf{y})$ sampled from image location $\mathbf{y}$ ends at a leaf $L(\mathbf{y})$. The probability of an object centered at the location $\mathbf{x}$ is then given by

$$p\big(E(\mathbf{x})|L(\mathbf{y})\big) = p\big(\mathbf{y}-\mathbf{x}|\,c{=}1,L(\mathbf{y})\big)\cdot p\big(c{=}1|L(\mathbf{y})\big). \tag{1}$$

3.2 Instance-specific Codebook

For tracking, however, one is not interested in the probability $p\big(E(\mathbf{x})|L(\mathbf{y})\big)$, *i.e.* the evidence for any instance of the class, but in the probability $p\big(E_I(\mathbf{x})|L(\mathbf{y})\big)$ where $E_I(\mathbf{x})$ is the evidence for a given instance I, namely the tracking target.

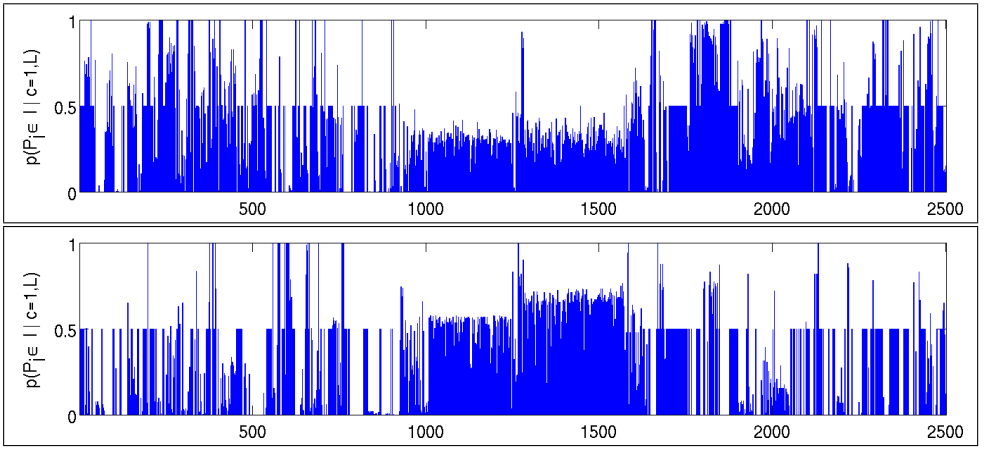


Figure 2: The probabilities $p\big(P_i \in I | c{=}1, L\big)$ for the first 2500 patches P_i of a single tree. The probabilities are on-line estimated by Equation (9) for two different persons after 100 frames. They give an instance-specific signature that can be used to improve tracking, see Fig. 1. While patches with probability > 0.5 are specific to the instance, probabilities < 0.5 indicate patches specific to the background. Patches with probability equal to 0.5 are mainly patches that have not been activated during tracking.

Before expressing $p\big(E_I(\mathbf{x})|L(\mathbf{y})\big)$, we rewrite Equation (1). Since each patch P_i has been observed at a relative position $\mathbf{d}_i$ with respect to object center, the spatial distribution $p\big(\mathbf{y}-\mathbf{x}|\, c{=}1, L(\mathbf{y})\big)$ can be approximated by a sum of Dirac measures $\delta_{\mathbf{d}_i}$:

$$p\big(E(\mathbf{x})|L(\mathbf{y})\big) = \frac{1}{|\mathscr{P}_{L(y)}|}\left(\sum_{P_i \in \mathscr{P}_{L(y)}} p\big(c{=}1|L(\mathbf{y})\big)\cdot\delta_{\mathbf{d}_i}(\mathbf{y}-\mathbf{x})\right). \tag{2}$$

For each vote cast by P_i, the vote is weighted by its probability of belonging to the object class, $p\big(c{=}1|L(\mathbf{y})\big)$. For $p\big(E_I(\mathbf{x})|L(\mathbf{y})\big)$, we are interested in its probability of belonging to the instance, *i.e.* $p\big(P_i \in I|L(\mathbf{y})\big)$. Hence, we have

$$p\big(E_I(\mathbf{x})|L(\mathbf{y})\big) = \frac{1}{|\mathscr{P}_{L(y)}|}\left(\sum_{P_i \in \mathscr{P}_{L(y)}} p\big(P_i \in I|L(\mathbf{y})\big)\cdot\delta_{\mathbf{d}_i}(\mathbf{y}-\mathbf{x})\right) \tag{3}$$

$$= \frac{1}{|\mathscr{P}_{L(y)}|}\left(\sum_{P_i \in \mathscr{P}_{L(y)}} p\big(P_i \in I|\, c{=}1, L(\mathbf{y})\big)\cdot p\big(c{=}1|L(\mathbf{y})\big)\cdot\delta_{\mathbf{d}_i}(\mathbf{y}-\mathbf{x})\right). \tag{4}$$

As we will see, Equation (4) is easy to calculate since $p\big(P_i \in I|\, c{=}1, L(\mathbf{y})\big)$ takes only the object class patches and not all patches into account. Note that the other terms are already computed for the off-line creation of the codebook (2).

3.3 Tracking

For tracking, we couple the codebook with a particle filter [12]. In the following, we describe the dynamical model (Section 3.3.1), the computation of the likelihood (Section 3.3.2), and

finally the computation of $p\big(P_i \in I | c{=}1, L(\mathbf{y})\big)$ (Section 3.3.3).

3.3.1 Prediction

We estimate the position $\mathbf{x}$, velocity $\mathbf{v}$, acceleration $\mathbf{a}$, and the scale s of the object in pixels. Assuming a constant acceleration, we have:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \Delta t \cdot \mathbf{v}_{t-1} + \frac{1}{2}(\Delta t)^2 \cdot \mathbf{a}_{t-1} + \eta_{\mathbf{x}}(\hat{s}_{t-1}) \qquad s_t = s_{t-1} \cdot (1 + \eta_s) \tag{5}$$

$$\mathbf{v}_t = \mathbf{v}_{t-1} + \Delta t \cdot \mathbf{a}_{t-1} + \eta_{\mathbf{v}}(\hat{s}_{t-1}) \qquad \mathbf{a}_t = \mathbf{a}_{t-1} + \eta_{\mathbf{a}}(\hat{s}_{t-1}). \tag{6}$$

The noise terms η depend on the estimated scale $\hat{s}_{t-1}$ of the previous frame and are modeled by zero mean Gaussians with standard deviation $\sigma_{\mathbf{x}} = \max\{4(\hat{s}_{t-1}/s_u), 4\}$, $\sigma_{\mathbf{v}} = 0.5\Delta t \cdot \sigma_{\mathbf{x}}$, $\sigma_{\mathbf{a}} = \sigma_{\mathbf{v}}^2$, and $\sigma_s = 0.04$, where s_u is the scale of the training data (Section 3.1).

3.3.2 Update

To get the posterior probability, the particles $S_k = (\mathbf{x}_k, \mathbf{v}_k, \mathbf{a}_k, s_k)$ are weighted by the likelihood $w_k \propto p\big(\mathscr{I}_t | S_k\big)$, where $\mathscr{I}_t$ is the current image. To this end, each patch $P(y)$ of the image $\mathscr{I}_t$ is matched against the codebook by passing it through the trees where it ends at a leaf $L(y)$. For each $P_i \in \mathscr{P}_{L(y)}$, weighted votes are then cast to a 3D Hough space $\mathscr{H}(\mathbf{h}, s)$. For a given scale s, the votes are given by

$$\mathbf{h} = \mathbf{y} - \frac{s}{s_u}\mathbf{d} \quad \text{with weight} \quad w_{\mathbf{h}} = \frac{1}{|\mathscr{P}_{L(y)}|} p\big(P_i \in I | c{=}1, L(\mathbf{y})\big) \cdot p\big(c{=}1 | L(\mathbf{y})\big). \tag{7}$$

Figure 1 (c) shows a 2D Hough image that accumulates the weights for a single scale. Since the previous scale $\hat{s}_{t-1}$ is known, we collect the votes only for scales around $\hat{s}_{t-1}$[1].

After all patches are mapped to the Hough space, the weights $w_k \propto p\big(\mathscr{H} | S_k\big)$ are computed for each particle S_k:

$$w_k = \sum_{(\mathbf{h},s) \in \mathscr{H}} w_{\mathbf{h}} \cdot K_{(\mathbf{x}_k, s_k)}(\mathbf{h}, s), \tag{8}$$

i.e. we sum the weights of the votes in the neighborhood of the particle weighted by a Gaussian kernel K[2]. An example of the posterior approximated by a weighted set of particles is given in Figure 3 (a). Before the weights are normalized, *i.e.* $\sum_k w_k = 1$, and the particles are re-sampled [12], the unnormalized weights are used for updating the codebook.

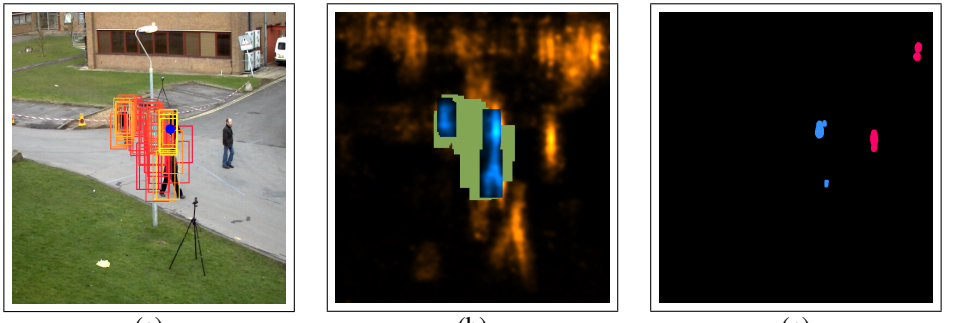
3.3.3 Codebook Update

According to (4), the codebook is updated by estimating the probability $p\big(P_i \in I | c{=}1, L(\mathbf{y})\big)$. To this end, we count the number of times a patch $P_i \in \mathscr{P}_{L(y)}$ votes for the target instance $\big\{y | P_i \in I \cap \mathscr{P}_{L(y)}\big\}$ and the number of times it votes for other objects $\big\{y | P_i \notin I \cap \mathscr{P}_{L(y)}\big\}$:

$$p\big(P_i \in I | c{=}1, L(\mathbf{y})\big) = \begin{cases} 0.5, & \text{if } \big|\big\{y | P_i \in I \cap \mathscr{P}_{L(y)}\big\}\big| + \big|\big\{y | P_i \notin I \cap \mathscr{P}_{L(y)}\big\}\big| = 0 \\ \dfrac{\big|\big\{y | P_i \in I \cap \mathscr{P}_{L(y)}\big\}\big|}{\big|\big\{y | P_i \in I \cap \mathscr{P}_{L(y)}\big\}\big| + \big|\big\{y | P_i \notin I \cap \mathscr{P}_{L(y)}\big\}\big|}, & \text{otherwise.} \end{cases} \tag{9}$$

[1] We use the sales $(0.8, 0.9, 1, 1.1, 1.2, 1.3, 1.4) \cdot \hat{s}_{t-1}$. In our experiments, the additional scales 1.3 and 1.4 proved to be beneficial. Furthermore, a lower bound on the scale prevents objects getting arbitrary small.

[2] For computational efficiency, we use two 2D Gaussian kernels with kernel width 31 for the two nearest scales centered at $\mathbf{x}_k$. The summed votes for the two scales are then linearly interpolated.



(a) (b) (c)

Figure 3: On-line adaption of the codebook. (a) After updating the particles, the multi-modal posterior distribution is approximated. The weights of the particles are indicated by color (*yellow: high, red: low*). The target is marked by a *blue dot*. (b) Based on the posterior, the voting space is clustered (*blue: foreground, red: background, green: uncertain*). Note that the intensity of the background (*red*) has been increased for a better visibility. In reality, the maximum of the background is much lower than for the foreground. (c) Votes that contributed to the detected local maxima are used to update the instance-specific statistics. Note that there is no local maximum for the most left person of the foreground (*blue*) since the values are very low compared to the global maximum of this cluster.

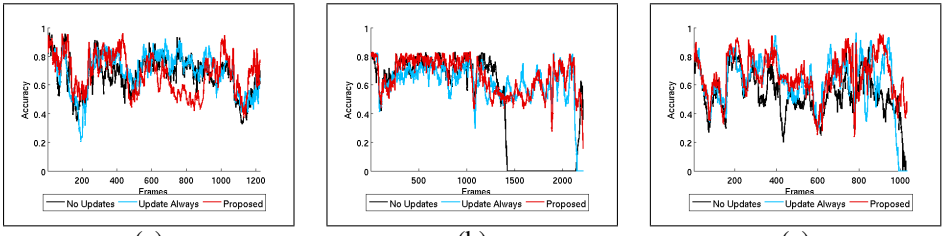
When the patch has not been previously activated for voting, we assume a fifty-fifty chance that the patch belongs to the instance I, see Fig. 2.

In order to compute (9), we have to estimate $\{y|P_i \in I \cap \mathscr{P}_{L(y)}\}$ and $\{y|P_i \notin I \cap \mathscr{P}_{L(y)}\}$. To this end, we assign a label to each $\mathbf{h}$ based on the posterior distribution as illustrated in Fig. 3. Namely 1 (*blue*) or -1 (*red*) if we are confident that it either belongs to the instance or it does not. When the posterior is greater than zero but relatively low, we assign the label 0 (*green*) to it. In practice, we sort the particles according to their weights and compute the bounding box for each particle. Starting with the particle with the lowest weight, we assign the label 0 to each pixel inside the bounding box if the weight is lower than a given threshold or 1 otherwise[3]. Finally, unlabeled pixels are set to -1. The obtained 2D mask provides a clustering of the voting space, see Fig. 3 (b). Since the scale is not important for this step, we only consider the voting space at scale $\hat{s}_{t-1}$, *i.e.* $\mathscr{H}^{\hat{s}}(\mathbf{h}) = \mathscr{H}(\mathbf{h}, \hat{s}_{t-1})$. However, the full voting space could also be clustered if necessary.

After clustering, we search local maxima in the positive and the negative cluster using a Gaussian kernel as in (8). The elements of the cluster labeled with 0 are discarded. By taking for each cluster all maxima that exceed at least 75% of the global maximum, we collect the votes that contributed to the local maxima and add them to the corresponding sets $\{y|P_i \in I \cap \mathscr{P}_{L(y)}\}$ and $\{y|P_i \notin I \cap \mathscr{P}_{L(y)}\}$. Finally, we re-compute for each leaf L the probability $p\big(P_i \in I | c{=}1, L\big)$ according to Equation (9).

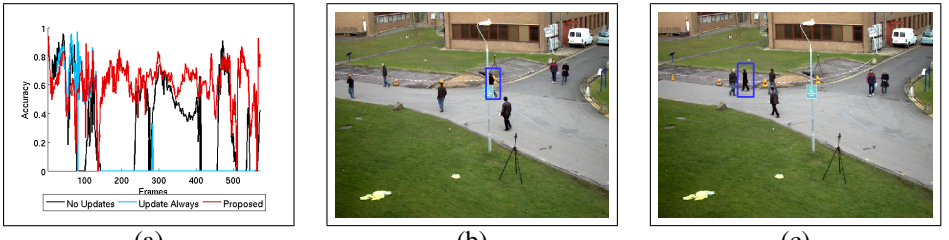
Note that the update neither changes the relative position d_i of the patches nor adds new patches to the leaves. The update only performs a re-weighting of the votes. On the one hand, the localization accuracy does not suffer from the updates as it might be the case for other on-line learning approaches, where the appearance model drifts away from the target due to poor localization in the previous frames. In the worst case, the tracker is confused by other instances of the class or better to say by objects that are very similar according

[3] Currently, the threshold is given by 60% of the maximum weight of the first frame.



(a) (b) (c)

Figure 4: Tracking accuracy for the i-Lids sequences [7] over time: (a) *easy*; (b) *medium*; (c) *hard*. Mean and standard deviation are given in Table 1. On the *easy* sequence, the class-specific codebook and the on-line adaption perform well. In sequences *medium* and *hard*, the scene is more crowded, see Fig. 6 for an example image. This is a situation where on-line adaption outperforms the class-specific codebook. Taking the confidence into account for clustering improves slightly the accuracy compared to the update-always-strategy.



(a) (b) (c)

Figure 5: Tracking accuracy for the sequence S2.L1 (view 1) of PETS09 [13]. Neither the off-line codebook nor the update-always-strategy are able to track the person since they are misled by other pedestrians in the scene (a). The proposed approach tracks the target (*blue box*) despite of occlusions and similarity with other instances. Two frames are shown in (b) and (c). Note that the accuracy is zero for a few frames where the strongest mode of the multi-modal posterior distribution does not overlap with the target, see Fig. 3 (a).

to the off-line trained detector. On the other hand, instances that are not localized a-priori by the codebook detector cannot be tracked since new observations are not added. In our experiments, we will demonstrate that the tracker handles difficult classes like humans that have a high frame-to-frame variation due to pose and many similarities between instances due to similar clothing.

4 Experiments

For a quantitative evaluation, we use two standard datasets i-Lids [7] and PETS09 [13] that have been recorded in an underground station and a public place. The sequences contain several instances (persons) of the class (pedestrians). For comparison, we apply the proposed tracker with on-line adaption, a particle filter without any adaption, *i.e.* using the class-specific codebook only, and a third version where the voting space is clustered only in foreground and background (Sec. 3.3.3), *i.e.* the codebook is always updated. As class-specific codebook, we use 5 pre-trained trees for the TUD pedestrian dataset [2] which are public available [14]. All trackers run with 50 particles and are initialized by a given bounding box. As an estimate, we take the strongest mode of the posterior. The accuracy is measured by

Accuracy (%)	Proposed	Update All	No Update	[15]	[16]	[4]
i-Lids(easy)	67.4 ± 13.5	$\mathbf{69.2} \pm 13.9$	66.9 ± 12.8	25.1 ± 21.3	0.8 ± 8.6	28.5 ± 18.9
i-Lids(medium)	$\mathbf{65.4} \pm 12.2$	60.8 ± 14.7	45.9 ± 33.9	23.1 ± 31.0	6.6 ± 21.1	35.9 ± 36.6
i-Lids(hard)	$\mathbf{65.9} \pm 15.0$	61.4 ± 19.6	53.2 ± 15.2	21.3 ± 32.4	6.8 ± 21.7	34.8 ± 37.7
PETS09	$\mathbf{60.3} \pm 15.3$	15.6 ± 29.0	31.3 ± 29.9	8.1 ± 20.5	12.0 ± 24.6	8.4 ± 22.2

Table 1: Mean and standard deviation of the tracking accuracy.

Feature Extraction	Particle Filter	Voting (Proposed)	On-line Adaption	Voting (Class-specific)
180*msec.*	0.3*msec.*	235*msec.*	63*msec.*	851*msec.*

Table 2: Since votes with zero probability are not cast, voting with on-line adaption is 2.8 times faster than voting with the class-specific codebook.

the PASCAL VOC criterion[4] that takes scale and position of the bounding box into account. The results in Figs. 4 and 5 show the benefit of the on-line adaption of a class-specific codebook to the target instance. While simple sequences without ambiguities can be handled by a class-specific codebook, more complex scenes with several instances cannot be tracked without the proposed on-line adaption. Note that the proposed approach is also faster, see Table 2. The results for some on-line boosting approaches [4, 15, 16] are given in Table 1. However, we have to emphasize that the public available implementations [4, 15, 16] neither handle scale nor make use of any off-line training.

We have also applied our algorithm to face tracking. To this end, we trained a codebook with 5 trees on a face dataset [20] where we used clutter as background images [17]. The sequences we have tested on are shown in Fig. 6. Our approach successfully tracks some standard sequences that have been used in the literature [1, 4, 6, 27]. Occlusions and changing lighting conditions do not impose a burden on our algorithm. Although the training data contains only frontal faces, the 360 degree head rotation in the sequence Girl is well captured. Sometimes the tracker follows the face of the girl with a slight delay which is a limitation of our dynamical model. Since these research sequences are not as challenging as real world data, we have taken a few sequences from YouTube[5]. The sequences cover a variety of challenges like low resolution, strong compression artifacts, motion blur, occlusions, and extreme lighting conditions ranging from very dark to very bright. In some frames, the face is very difficult to recognize without context. In addition, we have run the tracker on a very long sequence with 14166 frames and sequences with multiple faces.

5 Discussion

We have demonstrated that a class-specific codebook can be transformed into a more instance-specific codebook in a probabilistic manner. Coupled with a particle filter, the codebook becomes already a powerful instance tracking method without the use of additional classifiers to distinguish several instances during tracking. Compared to a class-specific codebook, the accuracy is not only increased but the computation time is also reduced. Compared to on-line learning approaches, tracking is much more reliable subject to an off-line trained codebook. Although this prevents tracking arbitrary objects, it is not a practical limitation since the objects of interest usually belong to a well defined class. Real-time performance is not yet achieved, but it seems to be feasible by speeding-up feature extraction and voting.

[4] $Accuracy(est) = (A_{gt} \cap A_{est})/(A_{gt} \cup A_{est})$, where A_{est} and A_{gt} are the areas of the estimated and the ground truth bounding box, respectively. When the areas do not overlap the value is 0, when they are identical the value is 1.

[5] www.youtube.com

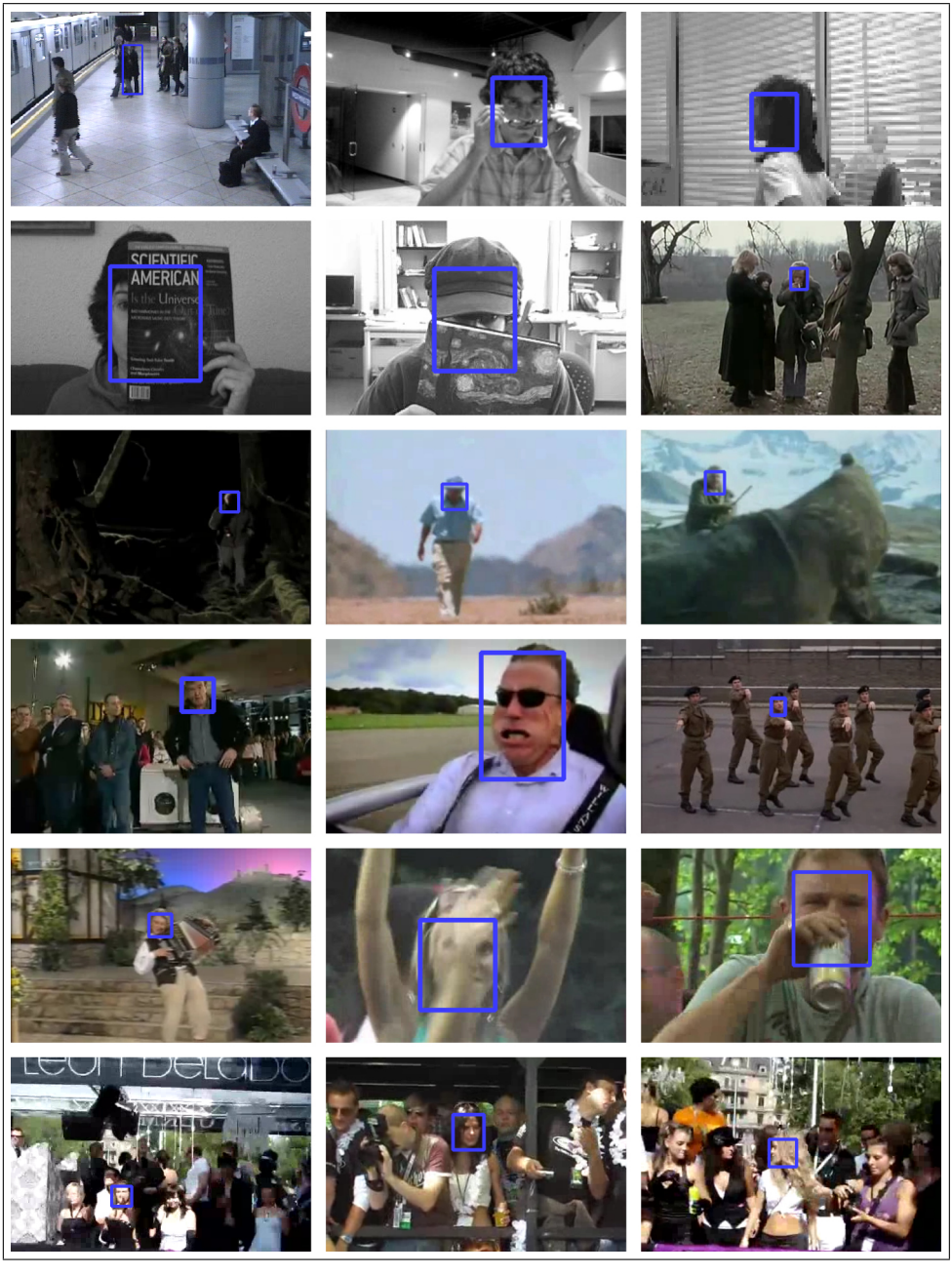


Figure 6: From top to bottom. Standard test sequences: i-Lids *hard* [0] (surveillance, crowded scene); David Indoor [0] (changing lighting conditions); Girl [0] (fast movements, 360 degree head rotation); Occluded Face [0] (occlusions); Occluded Face2 [0] (strong occlusions). YouTube test sequences: Klaus Kinski 1971 (camera zoom, 14166 frames); David Attenborough Night (very dark); Desert (strong shadow); Elephant Seal (fast movement); Top Gear (multiple faces); Top Gear Ariel Atom (face deformations); Monty Python Military (multiple faces); Florian Silbereisen (fast movements, low resolution). The last five are from public street parades (occlusions, multiple faces, low resolution, difficult lighting conditions).

Acknowledgments The work was partially funded through the SNF projects CASTOR (200021-118106) and Vision-supported Speech-based Human Machine Interaction (200021-130224).

References

[1] A. Adam, E. Rivlin, and I. Shimshoni. Robust fragments-based tracking using the integral histogram. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 798–805, 2006.

[2] M. Andriluka, S. Roth, and B. Schiele. People-tracking-by-detection and people-detection-by-tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[3] S. Avidan. Ensemble tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(2):261–271, 2007.

[4] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[5] V. Badrinarayanan, P. Perez, F. Le Clerc, and L. Oisel. Probabilistic color and adaptive multi-feature tracking with dynamically switched priority between cues. In *International Conference on Computer Vision*, 2007.

[6] S. Birchfield. Elliptical head tracking using intensity gradients and color histograms. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 232–237, 1998.

[7] Home Office Scientific Development Branch. Imagery library for intelligent detection systems i-lids. http://www.elec.qmul.ac.uk/staffinfo/andrea/avss2007_d.html.

[8] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[9] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool. Robust tracking-by-detection using a detector confidence particle filter. In *IEEE International Conference on Computer Vision*, 2009.

[10] R. Collins, Y. Liu, and M. Leordeanu. On-line selection of discriminative tracking features. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 27(1):1631 – 1643, 2005.

[11] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 142–149, 2000.

[12] A. Doucet, N. De Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, New York, 2001.

[13] J. Ferryman, J. Crowley, and A. Shahrokni. Pets 2009 benchmark data. http://www.cvg.rdg.ac.uk/PETS2009/a.html.

[14] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

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

[16] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *European Conference on Computer Vision*, LNCS. Springer, 2008.

[17] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[18] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *European Conference on Computer Vision*, LNCS, pages 343–356. Springer, 1996.

[19] A. Jepson, D. Fleet, and T. El-Maraghi. Robust online appearance models for visual tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10): 1296–1311, 2003.

[20] N. Kumar, P. Belhumeur, and S. Nayar. FaceTracer: A Search Engine for Large Collections of Images with Faces. In *European Conference on Computer Vision*, LNCS, pages 340–353. Springer, 2008.

[21] B. Leibe, A. Leonardis, and B. Schiele. Robust object detection with interleaved categorization and segmentation. *International Journal of Computer Vision*, 77(1-3):259–289, 2008.

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

[23] Y. Li, H. Ai, T. Yamashita, S. Lao, and M. Kawade. Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1728–1740, 2008.

[24] S. Maji and J. Malik. Object detection using a max-margin hough transform. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[25] A. Opelt, A. Pinz, and A. Zisserman. Learning an alphabet of shape and appearance for multi-class object detection. *International Journal of Computer Vision*, 80(1):16–44, 2008.

[26] R. R. Okada. Discriminative generalized hough transform for object dectection. In *International Conference on Computer Vision*, 2009.

[27] D. Ross, J. Lim, R.-S. Lin, and M.-H. Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1):125–141, 2008.

[28] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(7):1270–1281, 2008.

[29] S. Stalder, H. Grabner, and L. Van Gool. Beyond semi-supervised tracking: Tracking should be as simple as detection, but not simpler than recognition. In *On-line learning for Computer Vision Workshop*, 2009.

[30] F. Tang, S. Brennan, Q. Zhao, and H. Tao. Co-tracking using semi-supervised support vector machines. In *IEEE International Conference on Computer Vision*, 2007.

[31] R. Verma, C. Schmid, and K. Mikolajczyk. Face detection and tracking in a video by propagating detection probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25:1215–1228, 2003.

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

[33] T. Woodley, B. Stenger, and R. Cipolla. Tracking using online feature selection and a local generative model. In *British Machine Vision Conference*, 2007.

[34] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19: 780–785, 1997.

[35] Q. Yu, T. Dinh, and G. Medioni. Online tracking and reacquisition using co-trained generative and discriminative trackers. In *European Conference on Computer Vision*, LNCS, pages 678–691. Springer, 2008.