On-line Multi-View Forests for Tracking*

Christian Leistner, Martin Godec, Amir Saffari and Horst Bischof

Institute for Computer Graphics and Vision
Graz University of Technology, Austria
{leistner,godec,saffari,bischof}@icg.tugraz.at

Abstract. A successful approach to tracking is to on-line learn discriminative classifiers for the target objects. Although these tracking-by-detection approaches are usually fast and accurate they easily drift in case of putative and self-enforced wrong updates. Recent work has shown that classifier-based trackers can be significantly stabilized by applying semi-supervised learning methods instead of supervised ones. In this paper, we propose a novel on-line multi-view learning algorithm based on random forests. The main idea of our approach is to incorporate multi-view learning inside random forests and update each tree with individual label estimates for the unlabeled data. Our method is fast, easy to implement, benefits from parallel computing architectures and inherently exploits multiple views for learning from unlabeled data. In the tracking experiments, we outperform the state-of-the-art methods based on boosting and random forests.

1 Introduction

Tracking of a priori unknown objects is still a big challenges in computer vision. Despite the huge amount of research spent on this task it is still hard to design robust tracking systems that can achieve human-level performance. Visual trackers have to cope with all variations that occur in natural scenes such as shape and appearance changes, different illuminations as well as varying poses or partial occlusions. Numerous methods to approach the tracking tasks have been proposed, such as global template-based trackers, shape-based methods, probabilistic models using mean-shift, particle filtering, local key-point based trackers, or flow-based trackers. See also [1] for a detailed review.

A recently dominating trend is to apply classifiers – trained on object versus background – to track objects because they are able to deliver highly accurate results in real-time. Such tracking-by-detection systems [2] usually train a classifier at the very first frame versus its local background and perform re-detection in the succeeding frames. In order to handle rapid appearance and illumination changes, they use on-line classifiers that learn the target object based on their own predictions, *e.g.*, [3]. However, as these classifiers perform self-learning it is difficult to decide autonomously where exactly to take the positive and negative

* This work has been supported by the Austrian FFG project MobiTrick (825840) and Outlier (820923) under the FIT-IT program.

updates, respectively. Even if the object is tracked correctly, the alignment may not be perfect, which can lead to slightly wrong updates of the tracker (*a.k.a* label jitter). If these errors accumulate over time and self-reinforce the classifier in its wrong decisions, the tracker can easily drift [4].

Recent approaches try to tackle the drifting problem by formulating the tracking-by-detection task as one-shot semi-supervised learning. For instance, Grabner *et al.* [5] proposed an on-line semi-supervised boosting algorithm (On-line SemiBoost) where supervised updates are only performed at the beginning, *i.e.*, the first frame. All the subsequent frames are considered as unlabeled data used in order to regularize the learner with an unsupervised loss function. Although this method has shown to be less susceptible to drifting and is still more adaptive than a static classifier, it looses the capability of self-learning classifiers to adapt fast in case of rapid appearance changes. Also highlighting this problem of Online SemiBoost, recently Babenko *et al.* [6] formulated the tracking task as a multiple-instance learning (MIL) problem. Using MIL, the classifier in principle is still performing self-learning; however, the allowed positive update area around the current tracker can be increased and the classifier resolves the ambiguities by itself, yielding a tracker that is more robust than a pure supervised learner but less inertial than SemiBoost.

Another semi-supervised learning method that has been recently applied to tracking [7] is co-training, where the main idea is that two classifiers train each other on unlabeled data using distinct views [8]. Co-training can be very powerful but has the main drawback that it requires classifiers which are conditional independent given the class in order to converge, which is hard to fulfill in practice. One way to weaken this condition is to use two different classifiers [9] instead of different sufficient views. However, since this is still often not stable enough [10] showed that for such an approach it is necessary to take at least three classifiers. In practice, using different kinds of classifiers is complicated because it is still an open research problem how to compare the outputs. That is, the classifiers need to yield comparable performance in order to train each other. Also, the computing overhead grows with the number of classifiers and not for all of the learners on-line algorithms exist. Hence, what we need in practice is a single classifier approach that is able to emulate the multi-view robustness of using several classifiers.

In this work, we propose a novel on-line semi-supervised learning approach based on random forests. The method is inherently able to learn from multiple views and is thus called *MVForests*. The motivation for taking random forests [11] for our approach stems from the facts that they are fast and accurate learners, inherently parallel and multiclass capable and less suscpetible to class-label noise. We grow a common on-line random forest, similar to the recently proposed method by Saffari et al [12], and hence during evaluation our algorithm is identical to [12]. However, for learning, our method is able to exploit both labeled and unlabeled data, where the latter one is necessary in order to increase the stability of the tracking results. In order to incorporate the unlabeled data, we create a multi-view learning setting for each of the individual

trees; that is, each tree is trained individually with a possibly different set of label predictions for the unlabeled data. In particular, each tree is trained by a random sub-set of the remaining trees. Thereby, we guarantee that no single tree is performing self-learning and due to the random selection of trees we further achieve that no single tree is provided with the same label estimates for the unlabeled data set, which preserves the diversity among the trees. We incorporate multiple features into learning by restricting each tree to a subset of feature types. For instance, if we use color features and simple Haar features, trees of the type Haar features are trained by color-trees and vice versa. This setting can be extended to an arbitrary number of features. However, as we will show in the tracking experiments, our method is able to deliver highly accurate results even when using a single feature type. Our algorithm has several advantages: First, it provides an easy, fast an inherent way to learn from multiple views. This is necessary in order to ensure repeatability and real-time performance. Second, since we use more than two learners, we have weaker theoretical conditions in order to show convergence of our method [10]. As we will show in the experiments, our method outperforms the state-of-the-art tracking methods based on boosting and, on average, performs better than using fully self-learned random-forests [12] or off-line random forests [13].

The reminder of this paper is as follows. In the following section, we will introduce the basic notations and shortly review related work. Then, in Section 3, we will introduce our novel on-line learning method. We will evaluate our method on the task of visual object tracking in Section 4. Section 5 concludes the paper and discusses ideas for future work.

2 Notations and Related Work

In supervised learning one deals with a labeled dataset $\mathcal{D}^L \subseteq \mathcal{X} \times \mathcal{Y} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{|\mathcal{D}^L|}, y_{|\mathcal{D}^L|})\}$, where $\mathbf{x}_i \in \mathcal{X} = \mathbb{R}^P$ and $y_i \in \mathcal{Y} = \{+1, -1\}$. In contrast, unsupervised methods aim to find an interesting (natural) structure in $\mathcal{X}$ using only unlabeled input data $\mathcal{D}^U \subseteq \mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_{|\mathcal{D}^U|}\}$. Although supervised learners usually yield better results, most of the time unlabeled data can be obtained significantly easier than labeled samples. Hence, there exist increased interest in semi-supervised learning methods [14], such as co-training [8], which are able to exploit both labeled $\mathcal{D}^L$ and unlabeled $\mathcal{D}^U$ data. In co-training, the main idea is that two initial classifiers h_1 and h_2 are trained on labeled data $(\mathbf{x}_i^1, y_i), (\mathbf{x}_i^2, y_i) \in \mathcal{D}^L$, where $\mathbf{x}^1$ and $\mathbf{x}^2$ shows two views of the same data point. Then, these classifiers update each other using the unlabeled data set $\mathcal{D}^U$, if one classifier is confident on a sample whereas the other one is not. Co-training classifiers minimize the error on the labeled samples while increasing the agreement on the unlabeled data. Thus, the unlabeled data helps to improve the margin of the classifiers and to decrease the generalization error [15]. The approach has proven to converge, if two assumptions hold: (i) the error rate of each classifier is low, which means each classifier is in principle able to solve the given task, and (ii) the views must be conditionally independent [8]. Especially the second

condition is more of a theoretical requirement and is hard to fulfill in practice. However, the independence condition can be relaxed (*e.g.*, [16, 10, 15]), for instance, by applying more than two classifiers. If more than two classifiers are used, co-training becomes multi-view learning[1]. For practical usage, this means that co-training can even be applied if the learners are slightly correlated.

Computer vision naturally offers many physical "real-world" views, which can be exploited by co-training and multi-view learning. For instance, [17, 18] combined different simple cues based on shape, appearance, or motion to train visual classifiers. Co-training has also been applied to tracking. For instance, Tang et al. [19] used an SVM-based co-training approach that was later extended by Yu *et al.* [20]. Recently [7] presented an on-line boosting approach which outperforms the previous methods based on SVMs.

2.1 On-line Random Forests

Random Forests (RFs) were originally proposed by Amit and D. Geman [21], extended by Breiman [11] and consist of ensembles of T independent decision trees $f_t(\mathbf{x}) : \mathcal{X} \rightarrow \mathcal{Y} = \{1, \ldots, K\}$. For a forest $\mathcal{F} = \{f_1, \cdots, f_T\}$ the predictive confidence can be defined as $F_k(\mathbf{x}) = \frac{1}{T} \sum_{t=1}^{T} p_t(k|\mathbf{x})$, where $p_t(k|\mathbf{x})$ is the estimated density of class labels of the leaf of the t-th tree, where sample $\mathbf{x}$ resides. A decision is made by simply taking the maximum over all individual probabilities of the trees for a class k with $C(\mathbf{x}) = \arg\max_{k \in \mathcal{Y}} F_k(\mathbf{x})$. Breiman [11] showed that the generalization error of random forests is upper bounded by

$$GE \leq \bar{\rho} \frac{1 - s^2}{s^2}, \tag{1}$$

where $\bar{\rho}$ is the mean correlation between pairs of trees in the forest[2] and s is the strength of the ensemble (*i.e.*, the expected value of the margin over the entire distribution). In order to decrease the correlation of the trees, each tree is provided with a slightly different subset of training data by subsampling with replacement from the entire training set, *a.k.a* bagging [22]. Trees are trained recursively, where each split node randomly selects binary tests from the feature vector and selects the best according to an impurity measurement. The information gain after node splitting is usually measured with

$$\Delta H = -\frac{|I_l|}{|I_l| + |I_r|} H(I_l) - \frac{|I_r|}{|I_l| + |I_r|} H(I_r), \tag{2}$$

where I_l and I_r are the left and right subsets of the training data, respectively. $H(I) = -\sum_{i=1}^{K} p_i^j \log(p_i^j)$ is the entropy of the classes in the node and p_i^j is

[1] Note that in the literature the term "multi-view learning" is mainly used for learning from two views or classifiers, respectively. In this work, we distinguish between the two terms in a way that "co-training" only uses two views and is a special case of multi-view learning, where in the latter case always more than two views are used.

[2] The correlation is measured in terms of the similarities of the predictions.

the label density of class i in node j. The recursive training continues until a maximum depth is reached or no further information gain is possible.

RFs have demonstrated to be better or at least comparable to other state-of-the-art methods in both classification [11, 23] and clustering [24]. Especially, the speed in both training and evaluation is one of their main appealing properties. Additionally, since the trees are trained and evaluated independently, RFs can easily be parallelized, which makes them interesting for multi-core and GPU implementations [25]. Finally, compared to boosting and other ensemble methods, RFs are more robust against label noise [11]. This resistance to noise, is especially important when learning from unlabeled data where wrong label estimates are an inherent problem. These advantages of random forests have also led to increased interest in the computer vision domain. For instance, recently [26] presented an efficient object detection framework based on random forests, [27] presented a real-time algorithm for semantic segmentation based on randomized trees, and [28] presented state-of-the-art categorization results using RFs. Randomized trees have also successfully been applied to visual tracking, either in batch mode using keypoints [13] or on-line using tracking-by-detection [12].

Random forests, as reviewed above, is an off-line learning method. Recently, Saffari *et al.* [12] proposed an on-line version of RFs, which allows to use them as on-line classifiers in tracking-by-detection systems. Since recursive training of decision trees is hard to do in on-line learning, they propose a tree-growing procedure similar to evolving-trees [29]. The algorithm starts with trees consisting only of root nodes and randomly selected node tests f_i and thresholds θ_i. Each node estimates an impurity measure based on the Gini index ($G_i = \sum_{i=1}^{K} p_i^j(1 - p_i^j)$) on-line, where p_i^j is the label density of class i in node K. Then, after each on-line update the possible information gain ΔG during a potential node split is measured. If ΔG exceeds a given threshold β, the node becomes a split node, *i.e.*, is not updated any more and generates two child leaf nodes. The growing proceeds until a maximum depth is reached. Even when the tree has grown to its full size, all leaf nodes are further on-line updated. The method is simple to implement and has shown to converge fast to its offline counterpart. Additionally, [12] also showed that the classifier is faster and more noise-robust compared to boosting, which makes it an ideal candidate for our tracking system.

3 On-line Multi-view Training

As we have seen in the previous section, co-training is a popular approach in order to incorporate unlabeled data and has been used in many computer vision tasks. In this section, we will introduce a novel multi-view learning approach called *MVForests*, which extends the idea of co-training to an arbitrary number of views using random forests.

In particular, consider a random forest $\mathcal{F} = \{f_1, \cdots, f_T\}$, where T is the number of trees. Further, assume an on-line setting, where the training samples $\mathbf{x}_i$ arrive sequentially. If $\mathbf{x}_i$ is provided along with its class label y_i we can simply update the forest using [12]. If $\mathbf{x}_i$ is an unlabeled sample, we have to estimate

its label $\tilde{y}_i$; however, without using self-learning to reduce drifting. Therefore, we propose the following strategy: For each tree f_t we randomly select with replacement a sub-forest $\mathcal{F}_t^*$, with $|\mathcal{F}_t^*| = T$, *i.e.*, the forest consists of the same amount of trees as the original forest. This strategy can be interpreted as performing bagging on trees and results in forests where some trees of $\mathcal{F}$ are used multiple times, whereas on average one third (see [22]) of the total available trees are left out. We call the sub-forest $\mathcal{F}_t^*$ for the t^{th} tree *parent forest*. Note that for each tree its corresponding parent forest indices are created at the beginning of the learning and are kept fix during the on-line learning. Then, each f_t uses its corresponding parent forest in order to predict the label for $\mathbf{x}_i$; *i.e.*,

$$\tilde{y}_i^t = \arg\max_{k \in \mathcal{Y}} \mathcal{F}_{t,k}^*(\mathbf{x}_i). \tag{3}$$

We further use the confidence-rated predictions of each parent in order to encode uncertainties about a label in form of weight estimates. In particular, we take the confidence of the parent ensemble as weights in form of $\tilde{w}_i = \frac{1}{T}\sum_{t=1}^{T} p_t(\tilde{y}_i|\mathbf{x})$. For evaluation and testing, we take the original forest $\mathcal{F}$. The overall algorithm is depicted in detail in Algorithm 1.

Algorithm 1 On-line Multi-View Forests

Require: Sequential training example $\langle \mathbf{x}_i, y_i \rangle$ or $\langle \mathbf{x}_i \rangle$
Require: The size of the forest: T
// For all trees
for t from 1 to T **do**
  //sub-sample parent tree ensemble
  $\mathcal{F}_t^* \leftarrow$ SubSampleTreeIndices(T)
end for
// For each arriving sample $\mathbf{x}_i$
for t from 1 to T **do**
  if $\exists y_i$ **then**
    $f_t \leftarrow updateTree(f_t, \mathbf{x}_i, y_i)$
  else
    // Estimate label and weight
    $\tilde{y}_i^t = \arg\max_{k \in \mathcal{Y}} evalForest(\mathcal{F}_t^*, \mathbf{x}_i)$
    $\tilde{w}_i^t = getForestConfidence(\mathcal{F}_t^*, \mathbf{x}_i, \tilde{y}_i^t)$
    $f_t \leftarrow updateTree(f_t, \mathbf{x}_i, \tilde{y}_i^t, \tilde{w}_i^t)$
  end if
end for
Output the forest $\mathcal{F}$.

Discussion Although a random forest acts from the outside as a single classifier, it consists already of a committee of classifiers, *i.e.*, its trees. This suggests to bring multi-view learning inside a forest. However, there are two important

things that have to be considered: first, in order to get reliable label predictions for training each tree, we have to create sub-trees or parents that are strong enough to deliver accurate predictions. It is clear that for a tree f_t the strongest prediction that it can get out of the forest, though excluding itself, consists of the averaged prediction of the rest of the trees, *i.e.*, $\sum_m^T \mathbb{I}(t \neq m) f_t$, where $\mathbb{I}$ is the indicator function. However, it is also clear that if T is a large number, leaving out one tree will not change the overall predictions at all, which means that using this strategy each tree will get the same label estimates for $\mathbf{x}_i$. Therefore, MVForests create the parent forests in form of bagged classifiers from the entire forest, which results in parents where some trees are taken eventually several times and some trees are not taken at all. On average, $1 - \frac{1}{e}$ non-identical trees form a parent ensemble. We enforce the agreement of the trees on the unlabeled data, which overall increases the classification margin and improves the generalization. This strategy ensures that the predictions are reliable but not the same, thus yielding a typical multi-view setting, which overall preserves the diversity among the trees (see also Eq. (1)). To the best of our knowledge, this is the first approach that applies the bagging idea of [22] on sampling from a large amount of classifiers and not data, as in the typical setting.

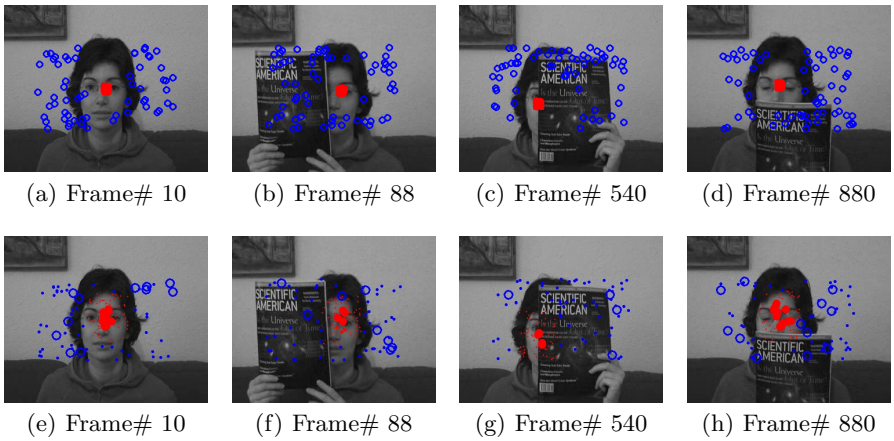
Our work is related to the tri-training algorithm [10], where the main idea is to take three classifiers h_i and the i^{th} classifier is trained by the remaining two classifiers if they agree on the prediction of an unlabeled datum and simulatenously each h_i has an error below a given threshold. MVForests differ from tri-training in three important aspects: (i) MVForests are not limited to three views but perform on an arbitrary number of views, only limited by the number T of trees that form an ensemble. (ii) Each unlabeled sample is incorporated, regardless of the agreement and the error of concomitant trees, which makes our approach much simpler. (iii) MVForest is an on-line algorithm. A second approach, which is related to MVForests is the recently proposed DAS-RF algorithm [30]; however, in this work an off-line optimization procedure is used which needs several parameters to be set, and it is not designed to learn on-line from multiple views.

4 Experiments

Within this section, we demonstrate the performance of our learning method for the task of object tracking, where we assume no prior knowledge about the object class available except its initial position. We use eight publicly available sequences including variations in illumination, pose, scale, rotation and appearance, and partial occlusions. The sequences *Sylvester* and *David* are taken from [31] and *Face Occlusion 1* is taken from [32], respectively. *Face occlusion 2*, *Girl*, *Tiger1,Tiger2* and *Coke* are taken from [6]. All video frames are gray scale and of size 320×240. To show the real accuracy of the compared tracking methods, we use the overlap-criterion of the VOC Challenge [33], which is defined as $A_{overlap} = R_T \cap R_{GT} / R_T \cup R_{GT}$, where R_T is the tracking rectangle and R_{GT} the groundtruth. We compare MVForests to supervised on-line

Sequence	*MVForest*	CoBoost	On-line RF	Off-line RF	MILBoost	SemiBoost
sylv	<u>0.54</u>	0.53	0.53	0.50	**0.60**	0.46
david	**0.71**	0.52	<u>0.69</u>	0.32	0.57	0.31
faceocc2	<u>0.78</u>	**0.79**	0.72	**0.79**	0.65	0.63
tiger1	**0.51**	<u>0.41</u>	0.38	0.34	0.49	0.17
tiger2	<u>0.45</u>	0.13	0.43	0.32	**0.53**	0.08
coke	0.28	**0.41**	<u>0.35</u>	0.15	0.33	0.08
faceocc1	**0.79**	<u>0.78</u>	0.71	0.77	0.60	0.71
girl	**0.77**	0.69	0.70	<u>0.74</u>	0.53	0.69

Table 1. Accuracy comparison of different approaches using single views measured using the Pascal VOC overlap criterion. Best performing method marked bold-face. Second best method marked underlined.



(a) Frame# 10 (b) Frame# 88 (c) Frame# 540 (d) Frame# 880

(e) Frame# 10 (f) Frame# 88 (g) Frame# 540 (h) Frame# 880

Fig. 1. Comparison of supervised updates ((a) to (d)) and MVForest's updates ((e) to (h)) (red circles: positive updates; blue circles: negative updates; circle radius corresponds to sample update weights). MVForests inherently update with smaller weight if the sample is noisy whereas supervised updates are hand-crafted and always weighted equally high (best viewed in color).

random forests (On-line RF) [12], off-line random forests (Off-line RF) [13], MILBoost [6], SemiBoost [5] and CoBoost [7]. We skip the related SVM-based co-training approaches as they were all outperformed by CoBoost. All methods are implemented in C++ and run in real-time, *i.e.*, $> 25fps$. Note that although MVForest are able to incorporate an arbitrary number of features, to ensure fair comparison in our experiments we evaluate the tracking performance of the different approaches using only Haar-features. We use forest sizes of 100 trees, with a maximum depth of 15. For the boosting classifiers, we use 2×50 selectors for CoBoost and the original settings for MILBoost [6]. We initialize the classifiers using virtual samples generated out of the first frame [13], 10 samples for on-line approaches and 500 for off-line approaches, respectively.

As depicted in Table 1, our approach is able to automatically cover the gap between supervised on-line training [12] and off-line training [13]. MVForests perform best in four out of eight sequences, and second best in three. Notably, we frequently outperform MILBoost, which is currently known to be the best performing method on these sequences. We also outperform CoBoost, the current state-of-the-art method for on-line co-training. Please refer to supplamentary material for the result videos.

Discussion Semi-supervised tracking methods virtually increase the tracking robustness by updating with lower weights in case of reduced confidence. The dilemma, however, arises in case of rapid appearance changes because this also results in lower confidence measurements. In such cases, semi-supervised trackers usually perform inferior to supervised ones [6]. The tracking results suggest that our method is a good compromise in this ambivalent setting, in terms that MVForest reduce their update weights in case of occlusions but due to the multi-view set-up are also adaptive when it comes to appearance changes. See also Figure 1 for a further illustration of MVForest's update behavior.

5 Conclusion and Future Work

In this paper, we have proposed a novel on-line multi-view learning algorithm using random forests called MVForests. MVForests learn from unlabeled data by emulating a multi-view setting inside the random forests, where each tree receives label estimates by a randomly selected sub-set of the trees forming the forest. We outperform the state-of-the-art learning methods on the task of visual object tracking. It should be noted that our multi-view learning approach is not limited to RFs, and in principle, can be applied to any ensemble of classifiers. MVForests are by no means limited to tracking. Hence, in future work, we plan to apply our method to additional vision problems such object detection and classification.

References

1. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Comput. (2006)
2. Avidan, S.: Ensemble tracking. PAMI **29**(2) (2007) 261–271
3. Grabner, H., Bischof, H.: On-line boosting and vision. In: CVPR. (2006)
4. Matthews, I., Ishikawa, T., Baker, S.: The template update problem. IEEE Trans. on Pattern Analysis and Machine Intelligence **26** (2004) 810 – 815
5. Grabner, H., Leistner, C., Bischof, H.: On-line semi-supervised boosting for robust tracking. In: ECCV. (2008)
6. Babenko, B., Yang, M.H., Belongie, S.: Visual tracking with online multiple instance learning. In: CVPR. (2009)
7. Liu, R., Cheng, J., Lu, H.: A robust boosting tracker with minimum error bound in a co-training framework. In: ICCV. (2009)
8. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proc. COLT. (1998) 92–100

9. Goldman, S., Zhou, Y.: Enhancing supervised learning with unlabeled data. In: ICML. (2000)
10. Zhou, Z.H., Li, M.: Tri-training. exploiting unlabeled data using three classifiers. IEEE Transactions on Knowledge and DAta Engineering **17**(11) (2005) 1529–1541
11. Breiman, L.: Random forests. Machine Learning (2001)
12. Saffari, A., Leistner, C., Godec, M., Santner, J., Bischof, H.: On-line random forests. In: OLCV. (2009)
13. Lepetit, V., Fua, P.: Keypoint recognition using randomized trees. In: CVPR. (2006)
14. Zhu, X., Goldberg, A.B.: Introduction to Semi-Supervised Learning. Third edn. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool (January 2009)
15. Abney, S.: Semi-Supervised Learning for Computational Linguistics. First edn. Computer Science and Data Analysis. Chapman & Hall/CRC (2007)
16. Balcan, M.F., A.Blum, Yang, K.: Co-training and expansion: Towards bridging theory and practice. In: NIPS, MIT Press (2004)
17. Levin, A., Viola, P., Freund, Y.: Unsupervised improvement of visual detectors using co-training. In: Proc. IEEE ICCV. Volume I. (2003) 626–633
18. Javed, O., Ali, S., Shah, M.: Online detection and classification of moving objects using progressively improving detectors. In: CVPR
19. Tang, F., Brennan, S., Zao, Q., Tao, W.: Co-tracking using semi-supervised support machines. In: ICCV. (2007)
20. Yu, Q., Dinh, T.B., Medioni, G.: Online tracking and reacquisition co-trained generative and discriminative trackers. In: ECCV. (2008)
21. Geman, Y.A.D.: Shape quantization and recognition with randomized trees. Neural Computation (1996)
22. Breiman, L.: Bagging predictors. Machine Learning **2**(24) (1996) 49–64
23. Caruana, R., Karampatziakis, N., Yessenalina, A.: An empirical evaluation of supervised learning in high dimensions. In: ICML. (2008)
24. Moosmann, F., Triggs, B., Jurie, F.: Fast discriminative visual codebooks using randomized clustering forests. In: NIPS. (2006) 985–992
25. Sharp, T.: Implementing decision trees and forests on a gpu. In: ECCV. (2008)
26. Gall, J., Lempinsky, V.: Class-specific hough forests for object detection. In: CVPR. (2009)
27. Shotton, J., Johnson, M., Cipolla, R.: Semantic texton forests for image catergorization and segmentation. In: CVPR. (2008)
28. Bosch, A., Zisserman, A., Munoz, X.: Image classification using random forests and ferns. In: ICCV. (2007)
29. Pakkanen, J., Iivarinen, J., Oja, E.: The evolving tree—a novel self-organizing network for data analysis. Neural Process. Lett. **20**(3) (2004) 199–211
30. Leistner, C., Saffari, A., Santner, J., Bischof, H.: Semi-supervised random forests. In: ICCV. (2009)
31. Ross, D., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. IJCV (2008)
32. Adam, A., Rivlin, E., Shimshoni, I.: Robust fragments-based tracking using the integral histogram. In: CVPR. (2006)
33. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A.: The pascal visual object class challenge 2007. In: VOC. (2007)