Online Multiple Classifier Boosting for Object Tracking

Tae-Kyun Kim[1] Thomas Woodley[1] Björn Stenger[2] Roberto Cipolla[1]
[1]Dept. of Engineering, University of Cambridge, Cambridge, UK
[2]Computer Vision Group, Toshiba Research Europe, Cambridge, UK

Abstract

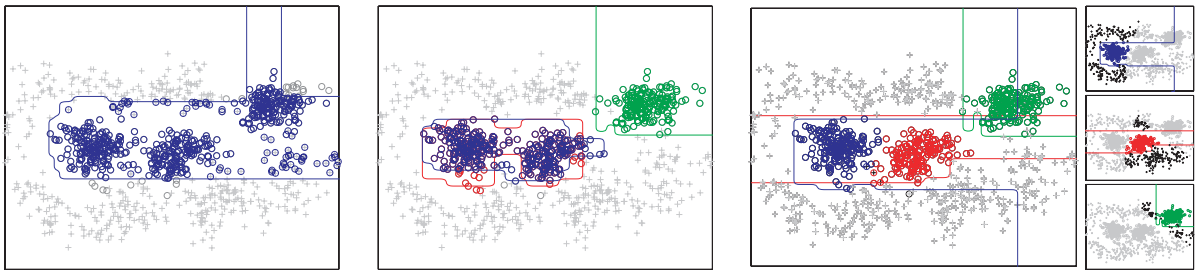
This paper presents a new online multi-classifier boosting algorithm for learning object appearance models. In many cases the appearance model is multi-modal, which we capture by training and updating multiple strong classifiers. The proposed algorithm jointly learns the classifiers and a soft partitioning of the input space, defining an area of expertise for each classifier. We show how this formulation improves the specificity of the strong classifiers, allowing simultaneous location and pose estimation in a tracking task. The proposed online scheme iteratively adapts the classifiers during tracking. Experiments show that the algorithm successfully learns multi-modal appearance models during a short initial training phase, subsequently updating them for tracking an object under rapid appearance changes.

1. Introduction

In object tracking a major challenge is handling appearance changes of the target object due to factors such as changing pose, illumination and deformation [18]. Recently a class of techniques using discriminative tracking has been shown to yield good results by treating tracking in a classification framework [1, 3, 5, 7]. A classifier is iteratively updated using positive and negative training samples extracted from each frame. Online boosted classifiers have been widely adopted owing to their efficiency and classification performance, which is required for tracking tasks [3, 7, 8]. However, as they maintain a single boosted classifier, they are limited to single view tracking or slow view changes of a target object. Tracking tends to fail during rapid appearance changes, because most weak learners of a boosted classifier do not capture the new feature distributions. Rapid adaptation of an online classifier in order to track these changes increases the risk of incorrectly adapting to background regions. A multi-modal object representation is therefore required.

Such a model can be either generative [4, 14] or discriminative [11]. Typically, in the latter case, distinct appearance clusters are found first and a classifier is trained on each [16]. Recent methods for multi-classifier boosting approach the problem by *jointly* clustering the positive samples and training multiple classifiers [2, 13]. These techniques have shown good results on learning multi-pose (or more generally multi-modal) classifiers for object detection, and our contribution is to formulate its online version for the task of multi-modal object tracking. However, this is not straightforward, the main reason being that in an online setting the number of positive and negative samples is not sufficient to ensure a good partitioning of the input space in terms of classifier expertise in the initial phase. Figure 1 illustrates the classification results of (a) standard Adaboost [6], (b) MCBoost [13] and (c) the proposed algorithm called MCBQ on a toy XOR classification problem. The positive class exhibits three clusters, but two of them actually form a single cluster in a discriminative sense as there are no negative points between them. Standard AdaBoost shows poor separation of the classes because it is unable to resolve XOR configurations. For the MCBoost algorithm and the proposed solution, we set the number of classifiers to be three. MCBoost successfully divides the two classes but shows overlapping areas of expertise for the two classifiers, since the two clusters without negative data points in-between can be correctly classified by a single boosting classifier. In contrast, the proposed algorithm shows improved partitioning of the input space. As a consequence, weak classifiers are used more efficiently. While tracking continues, additional negative samples are collected, eventually establishing three positive clusters in a discriminative sense in this example. However, in the case of MCBoost, the initially incorrectly assigned boosting classifiers are difficult to be correctly reassigned during online updates. We have observed this case when classifiers are initially trained on a short sequence that contains multi-views of a target object and are subsequently updated.

We therefore propose an extension of the multi-classifier boosting algorithm by introducing a weighting function $\mathcal{Q}$ that enforces a soft split of the input space. In addition, we present an online version of the algorithm to dynamically update the classifiers and the partitioning. The algorithm is applied to object tracking where it is used to learn dif-



(a) Standard AdaBoost [6] (b) MCBoost [13] (c) Proposed MCBQ

Figure 1. ***Learning cluster-specific classifiers on toy data***. *The positive class (circles) exhibits three clusters and is surrounded by data from the negative class (crosses). (a) The classification result using a standard boosting classifier shows errors due to the XOR configuration (colored circles denote classification as positive class). (b) The Multi-classifier boosting algorithm of [13] successfully divides the two classes but uses two boosting classifiers (blue and red line) in the same region, leading to inefficient use of weak classifiers. The two clusters with no negative data points between them can be correctly classified by a single boosting classifier. (c) The classification result of the proposed MCBQ algorithm shows improved classifier expertise.*

ferent appearance clusters during a short initial supervised learning phase.

The paper is organized as follows. We briefly review previous work on multi-classifier learning and object tracking. In section 3 we propose MCBQ, an algorithm for multi-classifier boosting using the weighting function $\mathcal{Q}$ to learn multi-modal appearance models. Section 4 shows an online version of MCBQ for tracking with a short initial learning phase and subsequently using an online update scheme. In the results section 5 we compare the proposed MCBQ with standard boosting and the MCBoost algorithm in [13]. We also compare the performance with that of two recent methods, Semi-supervised Online Boosting [7] and Online Multiple Instance Learning [3].

2. Prior Work

A number of online adaptation schemes have been proposed for object tracking [1, 5, 7]. The work in [5] introduced online feature selection for tracking, where in each frame the most discriminative features are chosen to compute likelihoods. *Ensemble Tracking* [1] takes a similar approach by combining a small number of weak classifiers using AdaBoost. Online boosting for tracking [7] introduced a scheme where features are selected from a pool of weak classifiers and combined into a strong classifier. Online schemes without any target model tend to suffer from drift. One solution is to introduce an object model that is learned prior to the tracking phase [8, 10]. This approach was first employed by Jebara and Pentland, who verified the tracker output with a classifier trained to detect the target object [10]. The work in [8] proposed semi-supervised learning, and included a boosted detector or simply the object region in the first frame as a prior to an online boosting scheme. A single AdaBoost classifier may not always be sufficient to capture multi-modal data, and one approach has been to cluster the data and train separate classifiers [11].

Recently multi-classifier boosting was introduced, where clustering and classifier training is performed jointly [2, 13]. These methods have so far been applied to object detection, where the full training set is available from the beginning. However, direct application to the online tracking domain may lead to significant overlap of the classifier regions due to the small amount of initial data as explained in the previous section. In the case of classifier overlap, weak classifiers are used less efficiently and initially overlapping classifiers in the algorithm are difficult to separate during subsequent updates.

This paper takes the view that MCBoost is an example of a more general algorithm, where clustering can be based on desired properties. In order to achieve this we introduce a function $\mathcal{Q}$, which weights the contributions of each classifier on a particular sample, similar to gating functions in Mixture of Experts models [12].

Other related work is multiple instance learning (MIL) [17]. The algorithm learns with 'bags of examples' which in the positive case only need to contain at least one positive example, thus training data does not have to be aligned. The MIL boosting algorithm simultaneously detects positive samples in bags in order to train weak classifiers. The MCBoost algorithms in [2, 13] can be seen as a multi-class extension of multiple instance learning where multiple classifiers, each of which simultaneously detects favored positive samples and learns weak learners for them, are trained. Both MIL and MCBoost are derived from the interpretation of boosting algorithms as gradient descent [15]. There is an online version of MIL Boosting for tracking [3] and our proposed method in this paper can be seen as a multi-class extension of [3].

3. Joint Boosting And Clustering

This section first briefly reviews multi-classifier boosting as proposed by [2, 13], then details our improvements

in the MCBQ algorithm. In both cases the following notation is used: Given is a set of n training samples $\mathbf{x}_i \in \mathcal{X}$, where $\mathcal{X}$ is the input domain (in our case image patches), with labels $y_i \in \{-1, +1\}$ corresponding to non-object and object, respectively. Additionally, each of the object samples can be considered belonging to one of K groups where the class membership is a priori unknown.

3.1. Multi-Classifier Boosting

In order to discriminate between object and non-object a boosting framework is used to train K strong classifiers H^k, where $\mathrm{H}^k(\mathbf{x}_i) = \sum_t \alpha_t^k \, h_t^k(\mathbf{x}_i), k = 1, ..., K$, and h_t^k is the t-th weak classifier of the k-th strong classifier weighted by α_t^k. Each weak classifier comprises a simple visual feature and threshold, and each strong classifier $\mathrm{H}^k(\mathbf{x}_i)$ is trained to focus its expertise on one of the K groups. The key is the use of a *noisy OR* function to combine the output of strong classifiers. This function classifies a sample as positive if any of the K strong classifiers does so, and negative otherwise:

$$p(\mathbf{x}_i) = 1 - \prod_k (1 - p^k(\mathbf{x}_i)), \tag{1}$$

where $p^k(\mathbf{x}_i) = 1/(1 + \exp(-\mathrm{H}^k(\mathbf{x}_i)))$. Following standard AdaBoost [6], a distribution of weights for the training samples is maintained, one distribution per strong classifier, and at each round the algorithm chooses a new weak classifier h_t^k with associated weight for each strong classifier, and updates the sample weights for the next round. For given weights, the algorithm finds K weak classifiers at the t-th round of boosting, to maximize $\sum_i w_i^k \, h_t^k(\mathbf{x}_i), h_t^k \in \mathcal{H}$, where $h_t^k \in \{-1, +1\}$ and $\mathcal{H}$ is a set of weak classifiers. The weak classifier weights $\alpha_t^k, k = 1, ..., K$ are then found by minimizing $\mathcal{L}(\mathrm{H} + \alpha_t^k h_t^k)$ by line search, where $\mathcal{L}$ is a loss function. Applying the AnyBoost method [15], the sample weights are set as the negative gradient of the loss function $\mathcal{L}$ with respect to the classifier score. Choosing $\mathcal{L}$ to be the negative log likelihood, the weight of k-th classifier over i-th sample is updated by

$$w_i^k = \frac{\partial \mathcal{L}}{\partial \mathrm{H}^k(\mathbf{x}_i)} = \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)} \, p^k(\mathbf{x}_i). \tag{2}$$

Clearly the choice of K, the number of strong classifier, is important for good performance. One method, suggested in [13], is to start with large values of K and select the number of distinctive clusters.

3.2. Classifier Assignment

Multi-classifier Boosting creates strong classifiers with different areas of expertise. However, it relies on the training data set containing negative samples which separate the positive samples into distinct regions in the classifiers' discriminative feature space. This also implies that there is no guarantee of pose-specific clustering. In fact there is no constraint in the algorithm that enforces strong classifiers to focus on a unique area of expertise, and there is no concept of a metric space on which perceived clusters can be formed. We make the classifier assignment explicit by defining functions $\mathcal{Q}^k(\mathbf{x}_i) : \mathcal{X} \rightarrow [0, 1]$ which weight the influence of strong classifier k on a sample $\mathbf{x}_i$. By mapping $\mathbf{x}_i$ into a suitable metric space, we can impose any desired clustering regime on the training set, thus $\mathcal{Q}$ defines a soft partitioning of the input space. The choice of $\mathcal{Q}$ is dependent on the application domain. In principle any function can be used that captures the structure of the input domain, i.e. that maps the samples to meaningful clusters. In this paper $\mathcal{Q}$ is defined by a K-component Gaussian mixture model in the space of the first d principal components of the training data. The k-th GMM mode defines the area of expertise of the k-th strong classifier. The GMM is updated using a EM-like algorithm alongside the weak classifiers in the boosting algorithm:

Algorithm 1 Updating Weighting Function $\mathcal{Q}^k$

1. Calculate the likelihood of each of the samples under the k-th strong classifier, $p^k(\mathbf{x}_i)$

2. Set the new probability of the sample being in the k-th GMM component as its current $\mathcal{Q}$ value scaled by the likelihood from the classifier, $\mathcal{Q}^k(\mathbf{x}_i)p^k(\mathbf{x}_i)$

3. Update the k-th cluster by the mean and covariance matrix of the samples under this probability.

The new noisy-OR function in Equation 1 becomes:

$$p(\mathbf{x}_i) = 1 - \prod_k (1 - \mathcal{Q}^k(\mathbf{x}_i) \, p^k(\mathbf{x}_i)), \tag{3}$$

leading to the new weight update equation:

$$w_i^k = \frac{\partial \mathcal{L}}{\partial \mathrm{H}^k(\mathbf{x}_i)} = \frac{y_i - p(\mathbf{x}_i)}{p(\mathbf{x}_i)} \, \frac{\mathcal{Q}^k(\mathbf{x}_i) \, p^k(\mathbf{x}_i) \, (1 - p^k(\mathbf{x}_i))}{1 - \mathcal{Q}^k(\mathbf{x}_i) \, p^k(\mathbf{x}_i)}. \tag{4}$$

The full MCBQ algorithm is summarized in Algorithm 2. Note that compared to the original multi-classifier boosting algorithm additional steps 1, 2, and 8 are required and step 7 is modified.

4. Online MCBQ for Object Tracking

The goal is to learn an object-specific appearance model using a short initial training sequence in order to guide the tracker [8, 14]. The number of training samples is limited, but is sufficient to bootstrap the classifier. Subsequently, we would like the tracker to remain flexible to some appearance changes while using the learned model as an anchor.

Algorithm 2 Multi-classifier Boosting with Weighting Function $\mathcal{Q}$ (MCBQ)

Input: Data set $(\mathbf{x}_i, y_i)$, set of pre-defined weak learners.
Output: Multiple strong classifiers $H^k(\mathbf{x}_i)$, weighting function $Q^k(\mathbf{x}_i)$.
1. Initialize Q with a Gaussian mixture model
2. Initialize weights w_i^k to the values of $Q^k(\mathbf{x}_i)$.
3. Repeat for $t = 1, ..., T$
4. Repeat for $k = 1, ..., K$
5. Find weak learners h_t^k maximizing $\sum_i w_i^k \mathrm{h}_t^k(\mathbf{x}_i)$.
6. Compute weights α_t^k maximizing $\mathcal{L}(\mathrm{H}^{\mathrm{k}} + \alpha_t^k \mathrm{h}_t^k)$.
7. Update weights by Equation 4.
8. Update weighting function $Q^k(\mathbf{x}_i)$ by Algo 1.
9. End
10. End

This motivates the following approach of iteratively adapting multiple strong classifiers with MCBQ.

In order to move MCBQ into an online setting we need a mechanism for rapid feature selection and incremental updates of the weak classifiers as new training samples become available. The online boosting algorithm [7] addresses this issue, allowing for the continuous learning of a strong classifier from training data. The key step is, at each boosting round, to maintain error estimates from samples seen so far, for a pool of weak classifiers. At each round t a *selector* S_t maintains these error estimates for weak classifiers in its pool, and chooses the one with the smallest error to add to the strong classifier.

To summarize, our tracking algorithm contains two-stages: Firstly, training data is assembled in a supervised learning stage, where the system is given initial samples which span the extent of all appearances to be classified. An initial MCBQ classifier is then built rapidly from this data. Secondly, additional training samples are supplied to update the classifier with new data during tracking.

4.1. Weak Learning and Selection

All weak classifiers use a single Haar-like feature. For online learning from a feature f and labeled samples $(\mathbf{x}_i, y_i)$ we create a decision threshold θ_m^k with parity p_m^k from the mean of feature values seen so far for positive and negative samples, where each feature value is weighted by the corresponding image weight:

$$h_{t,m}^k(\mathbf{x}_i) = p_m^k \operatorname{sign}(f(\mathbf{x}_i) - \theta_m^k), \tag{5}$$

$$\theta_m^k = (\mu^{k,+} + \mu^{k,-})/2, \quad p_m^k = \operatorname{sign}(\mu^{k,+} - \mu^{k,-}), \tag{6}$$

$$\mu^k = \frac{\Sigma_i |w_i^k| f(\mathbf{x}_i)}{\Sigma_i |w_i^k|}. \tag{7}$$

The error of the weak classifier is then given as the normalized sum of the weights of mis-classified samples:

$$e_{t,m}^k = \frac{\sum_i \mathbf{1}(h_{t,m}^k(x_i) \neq y_i)|w_i^k|}{\sum_i |w_i^k|}. \tag{8}$$

A weak classifier can then be chosen from a pool as the one giving the minimum error.

4.2. Supervised Learning

During the supervised learning stage, we have a set of weighted samples, and a global feature pool $\mathcal{F}$. Weight distributions are initialized to randomly assign positive samples to a strong classifier k, and at each round t and strong classifier k the equations 5, 6, 7, 8 are applied to initialize and select a weak classifier based on exact errors. In order to facilitate selection at the incremental update stage, we store in each selector S_t^k, for the positive and negative samples (1) for each feature value, the sum of weights of samples with that value, and (2) the sum of image weights.

To improve speed, each selector only keeps the best M performing weak classifiers for use in the incremental update stage. After each round of boosting, image weights are updated as in Equation 4, and voting weights calculated based on the error of the chosen weak classifier.

4.3. Incremental Update

Once the initial classifier has been created, it can be updated with new samples. Weights for positive samples are initialized based on their classification responses from each of the component strong classifiers in the MCBQ classifier, and the sample is passed through the boosting framework. The summations stored in each selector can be updated from the new sample, and thus the new classification thresholds for the weak classifiers calculated using equations 5, 6, 7. The error values from Equation 8 are used to choose the best weak classifier to add to the strong classifier. Finally, the worst-performing weak classifier is replaced with a new randomly-generated one. Note that in the case of $\mathcal{Q}$ being defined as a Gaussian mixture in PCA space, we update the PCA space by the algorithm of Hall *et al*. [9] before updating $\mathcal{Q}$. Pseudo-code is given in Algorithm 3.

5. Results

Pose Clustering. For this experiment we captured short training and testing sequences (about 100 frames each) of a face rotating from left to right, see Fig. 2. We trained classifiers using MCBoost [13] and the MCBQ algorithm on face images and random patches sampled from the training sequence. In both cases the number of strong classifiers K is set to 3 by hand. The $\mathcal{Q}$ function is defined by a 3-component Gaussian mixture on the first

Algorithm 3 Online MCBQ – Incremental Update

Require: Labeled training image $(\mathbf{x}_i, y_i)$, $y_i \in \{-1, +1\}$.
Require: MCBQ classifier $\mathrm{H}^k(\mathbf{x}_i)$, $k = 1, ..., K$.
// Initialize sample weight
$w_i^k = \mathcal{Q}^k(\mathbf{x}_i) / \sum_k \mathcal{Q}^k(\mathbf{x}_i)$

// For each round of boosting
for $t = 1, \ldots, T$ **do**
  // For each strong classifier, update selector $\mathbf{S}_t^k$
  for $k = 1, \ldots, K$ **do**
    // Update the selector's weak classifiers
    for $m = 1, 2, \ldots, M$ **do**
      // Update cached weight sums from sample's feature value, for positive and negative samples
      // Update classification threshold and parity
      Update $\left(h_{t,m}^k, (\mathbf{x}_i, y_i), w_i^k\right)$
      // Calculate new error $e_{t,m}^k$
      $e_{t,m}^k = \sum_i \mathbf{1}(h_{t,m}^k(\mathbf{x}_i) \neq y_i)|w_i^k|$
    end for
    // Choose the weak classifier with the lowest error
    $m^* = \mathrm{argmin}_m \left(e_{t,m}^k\right)$, $h_t^{k*} = h_{t,m^*}^k$ and $e_t^{k*} = e_{t,m*}^k$
    // Calculate voting weight
    $\alpha_t^k = \frac{1}{1+\exp\{-\ln\left(\frac{1-e_t^{k*}}{e_t^{k*}}\right)\}}$
    // Replace the weak classifier with the highest error
    $m^- = \mathrm{argmax}_m \left(e_{t,m}^k\right)$ and replace h_{t,m^-}^k
  end for

  // Update $\mathcal{Q}^k(\mathbf{x}_i)$ function
  // Update importance weights by Equation 4, then renormalize.
end for

30 principal components. The graph in Fig. 2 shows the contribution of each strong classifier on the test sequence. The MCBoost algorithm shows no clear pose-specific response, while MCBQ has successfully captured three distinct pose clusters, left, right, and center, as shown by the changes in classifier weights.

Tracking Performance. In order to evaluate the performance on the multi-appearance tracking problem, we captured four sequences where the target object rapidly changes its pose. The sequences are *toyface* (452 frames), *handball* (210 frames), *cube* (357 frames), and *face* (185 frames). We also compared on the public *Sylvester* sequence (1345 frames). The performance was evaluated against manually labeled ground truth. We compared AdaBoost, MCBoost and MCBQ trackers (both manually set to $K = 2$), as well as two publicly available trackers, Semi-supervised Boosting [8] and MIL tracking [3]. For each

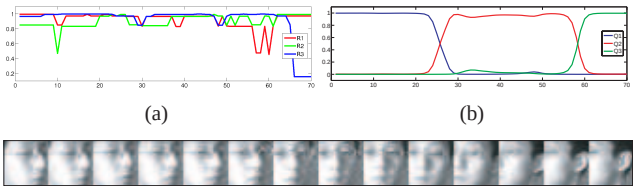


Figure 2. ***Improved pose expertise:*** *Plots of the contributions of three strong classifiers given the image input (bottom row).* **(a)** *MCBoost [13] shows no clear separation of expertise over different poses, while* **(b)** *MCBQ has learned pose-specific classifiers, corresponding to left, center and right view of the face.*

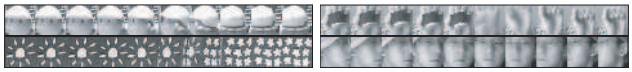


Figure 3. ***Positive class samples for training.*** *A subset of the positive samples is shown for the four sequences.*

sequence the initial classifier was trained on a short initial training set (25-40 frames), capturing the appearance variation, and updated online during tracking. Examples of positive training samples are shown in Fig. 3. Because such a training set is generally not available for public tracking sequences, the training data for the *Sylvester* sequence was constructed by randomly sampling 30 frames from the whole sequence. For AdaBoost, MCBoost and MCBQ 50 random patches per frame were collected as negative class samples. We stopped boosting rounds when the classification error reached zero on the training samples. The public code for semi-supervised Boosting and MIL tracking was modified so that these methods can also be trained on the initial set, otherwise their default parameters were used. Parameter settings were unchanged for all experiments. Fig. 4 shows the tracking errors on the five sequences, and Table 1 shows the mean error for each tracker (for SemiBoost the best of five runs is shown). While none of the algorithms was able to successfully track the target in all sequences, MCBQ showed the best overall performance, in particular outperforming AdaBoost and MCBoost. The MIL tracker performed best on two sequences, however, was not able to recover from drift in two of the other sequences. Overall, the single classifier trackers tend to adapt to a current appearance mode forgetting previous appearance modes, which often makes them fail when target objects rapidly change appearance modes. Fig. 5 shows example frames from the test sequences.

6. Conclusion

This paper proposed MCBQ, a multi-classifier boosting algorithm with a soft partitioning of the input space. This is achieved with a weighting function $\mathcal{Q}$ ensuring that coherent clusters are formed. We applied the method to simultaneous tracking and pose estimation. The learned model allows tracking during rapid pose changes, since it cap-

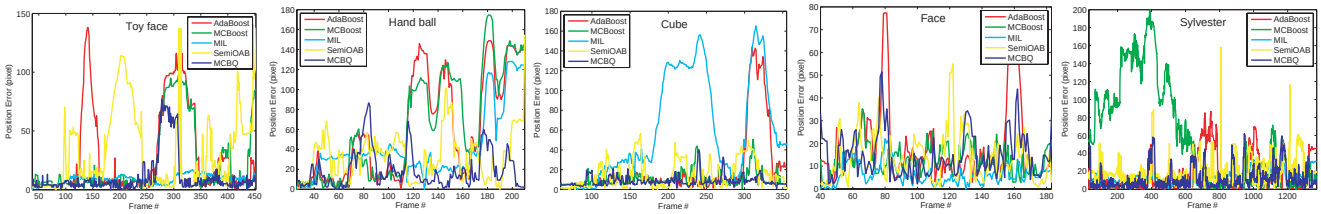


Figure 4. **Tracking error on test sequences.** *The plots show the tracking error over time on four test sequences for AdaBoost (red) MCBoost (green), MCBQ (blue), MILTrack (cyan), and SemiBoost (yellow). MCBQ shows the best overall performance.*

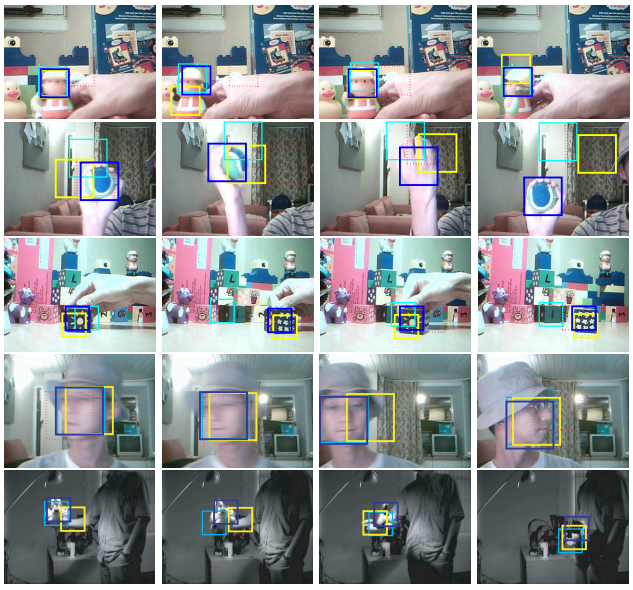


Figure 5. **Example tracking results on test sequences.** *The comparison shows tracking results for MCBQ (blue), AdaBoost (red), MILTrack (cyan), and SemiBoost (yellow) in the evaluation. See text for details.*

Sequence	SemiBoost	MIL	AdaBoost	MCBoost	MCBQ
Toy face	31	**9**	28	22	*10*
Hand ball	*33*	41	59	62	**19**
Cube	16	59	20	*10*	**8**
Face	15	**6**	18	*12*	14
Sylvester	20	*15*	17	58	**13**
Cumulative	22.1	*21.6*	22.9	41.9	**12.3**

Table 1. **Tracking error.** Average center location errors rounded to nearest integer (in pixels). Algorithms compared are SemiBoost [8] (best of 5 runs), MILTrack [3], our implementations of AdaBoost, MCBoost [13] and MCBQ trackers. Bold font indicates best performance, italic second best. Cumulative errors are weighted by the number of frames per sequence.

tures multiple appearances. Existing single classifier trackers tend to adapt to a single appearance mode, forgetting previous modes. MCBQ can be seen as an extension of MCBoost [13] for the online setting, or a multi-class extension of the MIL tracker [3]. Future work includes a more principled selection of the number of strong classifiers and exploring other choices for the weighting function.

References

[1] S. Avidan. Ensemble tracking. *IEEE Trans. Pattern Analysis and Machine Intell.*, 29(2):261–271, 2007. 1, 2

[2] B. Babenko, P. Dollár, Z. Tu, and S. Belongie. Simultaneous learning and alignment: Multi-instance and multi-pose learning. In *Workshop on Faces in Real-Life Images*, October 2008. 1, 2

[3] B. Babenko, M.-H. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *CVPR*, Miami, FL, June 2009. 1, 2, 5, 6

[4] M. J. Black and A. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *Proc. ECCV*, pages 329–342, Cambridge, UK, 1996. 1

[5] R. Collins, Y. Liu, and M. Leordeanu. Online selection of discriminative tracking features. *PAMI*, 27(10):1631–1643, 2005. 1, 2

[6] Y. Freund and R. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *J. of Computer and System Sciences*, 55(1):119–139, 1997. 1, 2, 3

[7] H. Grabner and H. Bischof. On-line boosting and vision. In *Proc. CVPR*, volume 1, pages 260–267, 2006. 1, 2, 4

[8] H. Grabner, C. Leistner, and H. Bischof. Semi-supervised on-line boosting for robust tracking. In *Proc. ECCV*, Marseille, France, October 2008. 1, 2, 3, 5, 6

[9] P. Hall, D. Marshall, and R. Martin. Merging and splitting eigenspace models. *Trans. PAMI*, 22(9):1042–1049, 2000. 4

[10] T. Jebara and A. Pentland. Parameterized structure from motion for 3d adaptive feedback tracking of faces. In *CVPR*, pages 144–150, June 1997. 2

[11] M. Jones and P. Viola. Fast multi-view face detection. Technical Report 96, MERL, 2003. 1, 2

[12] M. I. Jordan and R. A. Jacobs. Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994. 2

[13] T.-K. Kim and R. Cipolla. MCBoost: Multiple classifier boosting for perceptual co-clustering of images and visual features. In *NIPS*, Vancouver, Canada, December 2008. 1, 2, 3, 4, 5, 6

[14] K.-C. Lee, J. Ho, M.-H. Yang, and D. Kriegman. Visual tracking and recognition using probabilistic appearance manifolds. *Computer Vision and Image Understanding*, 99(3):303–331, 2005. 1, 3

[15] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Boosting algorithms as gradient descent. In *NIPS*, pages 512–518, 2000. 2, 3

[16] A. Torralba, K. P. Murphy, and W. T. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. In *CVPR*, pages 762–769, Washington, DC, July 2004. 1

[17] P. Viola, J. C. Platt, and C. Zhang. Multiple instance boosting for object detection. In *NIPS*, pages 1417–1426, 2006. 2

[18] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *ACM Journal of Computing Surveys*, 38(4):1–45, 2006. 1