On-line Learning-based Object Tracking Using Boosted Features

Bogdan Kwolek

Rzeszów University of Technology,
W. Pola 2, 35-959 Rzeszów, Poland
bkwolek@prz.rzeszow.pl

Abstract. The most informative and hard to classify examples are close to the decision boundary between object of interest and background. Gentle AdaBoost built on regression stumps focuses on hard examples that provide most new information during object tracking. They contribute to better learning of the classifier while tracking the object. The tracker is compared to recently proposed algorithm that uses on-line appearance models. The performance of the algorithm is demonstrated on freely available test sequences. The resulting algorithm runs in real-time.

1 Introduction

Object tracking is a central theme in computer vision with applications ranging from visual surveillance to human-computer interfaces. The goal is to find and to follow moving objects between consecutive frames. For applications where the observed appearance of a tracked object undergoes complex changes a learning algorithm can improve the tracking capabilities. Tracking algorithms should poses learning abilities to overcome drifting.

Several algorithms have been proposed for object tracking. To cope with observable appearance variations many algorithms incrementally adjust models to the changes of object or environment [1][2]. To improve the robustness of tracking many algorithms take into account the tracking environment and employ information about the background [3][4][5]. Recently visual tracking has been approached using classification methods such as support vector machines [3] or AdaBoost-based [4][5]. In such methods a learning of the classifier while tracking of the object takes place.

Automatically obtaining a set of both positive and negative examples for on-line learning is a difficult task. Levin *et al.* [6] begin with a small set of manually labeled data and then generate supplementary examples by applying co-training of two classifiers. In order to avoid hand labeling the usage of motion detection to obtain the initial training set was proposed by Nair and Clark [7]. In our approach Gentle AdaBoost built on regression stumps focuses on hard examples that provide most new information during object tracking. These pixels are sampled on the basis of the confidence map calculated by the strong classifier. In the training set we additionally include the most stable features as well as features from the initial object template. The learning algorithm considers the temporal coherence between images of object undergoing tracking.

2 Learning in object tracking

When learned off-line classifiers are employed in a visual tracker the tracking can be realized trough detection of the target. In [8], Okuma *et al.* proposes an approach that combines AdaBoost based detector using a color model to construct a proposal distribution for the particle filter. In many tracking systems a binary classifier that discriminates the object and the background is employed. In this context the Support Vector Machine that builds a hyperplane between two classes of examples based on the criterion of large margin can be employed. Considering tracking as binary classification, Avidan [3] proposed a support vector based tracker with learning capabilities built on the polynomial kernel. The score of support vector machine is maximized for every frame. In the work of Williams *et al.* [9] a system built on the relevance vector machine which employs temporal fusion is described. The AdaBoost that belongs to group of large margin classifiers is employed to learn the classifier in algorithm termed as ensemble tracking [4]. The classifier represents the appearance model that is updated by adding the recent features. Different approach is presented in work [10], which employs image pairs and temporal dependencies into a learned similarity function instead of learning a classifier to differentiate the object from the background. In [11], Oza and Russel propose on-line version of boosting which simulates the bootstrap process through updating each base model using multiple copies of each new example. Some work has been done in the past to enable automatic labeling of training data. Robust automatic labeling is a highly desirable property in any learning based tracking system. Levin *et al.* [6] propose the so called co-training approach which consists in starting with a small training set and increasing it by co-training of two classifiers, operating on different features. Ensemble methods such as boosting and bagging have demonstrated significant advantages in off-line settings. However little work has been done in exploring these methods in on-line settings. Nair and Clark [7] use the motion detection for constructing the initial training set and then the Winnow as a final classifier.

3 CamShift based object tracking

Bradski's CamShift [12] is representative of a group of algorithms that exploit the color cue to locate and subsequently track an object in video sequences. It is very fast, it can deal with irregular object motion arising due to perspective, uncalibrated cameras, image noise and so on. This general purpose tracking algorithm is based on a robust non-parametric technique called mean-shift to seek the nearest mode of probability distribution and requires minimal training. The major advantage of CamShift based algorithms is that they can work with cheap desktop cameras. Since our on-line learning based tracking algorithm is intended to spend small number of CPU cycles, we decided to base the tracking on 2D color cues, for example on RG or HS color components, similarly as in the mentioned above CamShift. Our tracking algorithm also employs the CamShift to track the object of interest, but it additionally learns the color distributions using boosting. CamShift tracking algorithm is based on a robust non-parametric technique called mean-shift to locate density extrema or modes of a given probability distribution without doing an exhaustive search. The locating starts from the final location in the previous frame and proceeds iteratively to find the nearest mode. Given a previous location (x, y) of the kernel the local mean shift vector represents a translation towards the nearest mode along the direction of maximal increase of the underlying density. The local density is estimated within a local neighborhood of this location via kernel density estimation where the kernel weights are multiplied by weights that are associated with data. The mean-shift based mode seeking can lead to fast trackers as well as effective kernel particle filters where a movement of particles towards the modes of the posterior probability density takes place [1][13].

In CamShift a uniform kernel is employed and the algorithm operates on probability images. Each pixel value in the probability image $P(x, y)$ represents the probability of membership of the pixel to the object of interest. The object probability density image was extracted through thresholding the confidence score of the AdaBoost classifier.

The mean location of the distribution within the kernel is computed using moments [12]. It is given by:

$$x_1 = \frac{\sum_x \sum_y xP(x,y)}{\sum_x \sum_y P(x,y)}, \quad y_1 = \frac{\sum_x \sum_y yP(x,y)}{\sum_x \sum_y P(x,y)} \tag{1}$$

where x, y range over the kernel. Assuming an elliptical approximation of the underlying distribution the eigenvalues (major length and width) of the probability distribution are calculated as follows [12]:

$$l = 0.707\sqrt{(a+c) + \sqrt{b^2 + (a-c)^2}}, \; w = 0.707\sqrt{(a+c) - \sqrt{b^2 + (a-c)^2}} \tag{2}$$

where

$a = \frac{M_{20}}{M_{00}} - x_1^2, \;\; b = 2\frac{M_{11}}{M_{00}} - x_1 y_1, \;\; c = \frac{M_{02}}{M_{00}} - y_1^2, \;\; M_{00} = \sum_x \sum_y P(x,y),$

$M_{20} = \sum_x \sum_y x^2 P(x,y), \;\; M_{02} = \sum_x \sum_y y^2 P(x,y).$

Using the uniform kernel the zeroth and first order moments are computed implicitly.

The algorithm repeats the computation of the centroid and repositioning of the kernel until the position difference converges to some predefined value, that is, changes less than some assumed value. The size of the mean shift kernel is updated on the basis of l and w. Therefore the size of the kernel is adjusted according to the shape of the underlying distribution. The algorithm requires a selection of the initial location and size of the region of interest. The mean shift iterations are typically carried out starting from a smaller window size with regard to the final window size from the previous frame. The algorithm outputs the position, dimensions, and orientation of object undergoing tracking. The number of mean-shift iterations needed to find a mode is relatively small and ranges between 2 and 5. CamShift is able to handle noisy images without the need for extra filtering or adaptive smoothing.

4 Boosting

Boosting originates from a machine learning model known as Probably Approximately Correct (PAC). Boosting algorithms combine simple decision rules into more complex ones. They aim at finding an accurate classifier consisting of many base classifiers, which are only moderately accurate. A typical algorithm consists of a boosting algorithm and a learning algorithm. The boosting algorithm executes the base learning algorithm multiple times to achieve the desired classification performance. During iterations the weights are updated dynamically according to the errors in previous round of learning. The base learning algorithm takes into account a weight coupled with each training instance and attempts to find a learned hypothesis that minimizes the weighted classification error. The learning algorithm generates classification rules that are combined by the boosting algorithm into the final classification rule. In the first step a boosting algorithm constructs an initial distribution of weights over the training set. The weights are greater than zero, sum to one and constitute a distribution over the training set. Using the weighted training set the algorithm searches for a classification rule consisting in a selecting a base classifier that gives the least weighted error. The weights of the data that are misclassified by the selected base classifier are increased. This leads to selection of classifier that performs better on examples misclassified previously. Each weak classifier predicts the label of the data. In consequence, AdaBoost [14], which is the adaptive version of boosting minimizes the following exponential loss function:

$$J(F) = E(e^{-yF(x)}), \tag{3}$$

where E denotes the expectation and the strong classifier $F(x)$ is a linear combination of T weak classifiers $f_i(x)$:

$$F(x) = \sum_{i=1}^{T} \alpha_i f_i(x), \tag{4}$$

with parameters α_i to balance the evidence from each feature. The two terms in classification function, the set of decision rules $\{f_i\}_{i=1}^{T}$ and combining coefficients $\{\alpha_i\}_{i=1}^{T}$ are learned. AdaBoost was applied by Viola *et al.* to face detection [15] and recently pedestrian detection [16] with impressive results.

4.1 Gentle AdaBoost

We employ in our tracking algorithm a version of boosting called Gentle AdaBoost [17], because it requires fewer iterations to achieve similar classification performance in comparison with other methods. Given a set of training instances $\mathcal{X}$ and a corresponding weight distribution D the boosting algorithm calculates a weak hypothesis $f : \mathcal{X} \mapsto R$, where the sign of f determines the predicted label y of the instance $x \in \mathcal{X}$. The magnitude $|f(x)|$ expresses the confidence of the prediction. Suppose we have a current ensemble hypothesis $F(x) = \sum_{t=1}^{T} f_t(x)$ and seek better one $F + f$ by minimizing the following criterion:

$$J(F + f) = E[e^{-y(F(x)+f(x))}], \tag{5}$$

where E denotes the expectation. Gentle AdaBoost minimizes this equation by employing adaptive Newton steps [17], which corresponds to minimizing at each step a weighted squared error. At each step m the current ensemble hypothesis F is updated as follows $F(x) \leftarrow F(x) + f_m$, where f_m is selected to minimize a second order Taylor approximation of the cost function. Replacing the weighted conditional expectation $E[y|x]$ in (5) with an empirical expectation over the training data leads to minimizing the weighted squared error:

$$J = \sum_{i=1}^{N} w_i (y_i - f_m(x_i))^2, \tag{6}$$

where $w_i = e^{-y_i F(x_i)}$ and the summation is over the training exemplars.

4.2 Regression stumps based week learner

The week learners we employ in our approach are regression stumps of the following form:

$$f_m(x) = a\delta(x^{(k)} > \theta) + b \tag{7}$$

where $x^{(k)}$ denotes the k-th coordinate of K dimensional feature vector x, δ is the Kronecker delta function, θ is a threshold, and a, b are regression parameters. Such binary regression stumps were employed in [15][18]. The following parameters of a regression stump minimize the function (6):

$$\begin{aligned} b &= \frac{\sum_j w_j y_j \delta(x_j^{(k)} \le \theta)}{\sum_j w_j \delta(x_j^{(k)} \le \theta)} \\ a &= \frac{\sum_j w_j y_j \delta(x_j^{(k)} > \theta)}{\sum_j w_j \delta(x_j^{(k)} > \theta)} - b. \end{aligned} \tag{8}$$

This means that in each iteration m we should determine four parameters of the regression stump (7), namely a, b, θ and k. At the beginning, we determine parameters a and b with respect to each possible threshold $\theta_i^{(k)} = x_i^{(k)}$, i.e. for $i = 1, 2, ..., N$ and $k = 1, 2, ..., K$:

$$\begin{aligned} b_i^{(k)} &= \frac{\sum_{j=1}^{N} w_j y_j \delta(x_j^{(k)} \le x_i^{(k)})}{\sum_{j=1}^{N} w_j \delta(x_j^{(k)} \le x_i^{(k)})} \\ a_i^{(k)} &= \frac{\sum_{j=1}^{N} w_j y_j \delta(x_j^{(k)} > x_i^{(k)})}{\sum_{j=1}^{N} w_j \delta(x_j^{(k)} > x_i^{(k)})} - b_i^{(k)}. \end{aligned} \tag{9}$$

Next, we calculate the error according to the following formula:

$$e_i^{(k)} = \sum_{j=1}^{N} w_j (y_j - a_i^{(k)} \delta(x_j^{(k)} > x_i^{(k)}) + b_i^{(k)})^2. \tag{10}$$

Then for each dimension k we seek for thresholds $\theta^{(k)} = x_{i^{*(k)}}^{(k)}$, which minimize the error function given by (10). This operation can be expressed as follows:

$$i^{*(k)} = \arg \max_{i=1,2,\ldots N} \{e_i^{(k)}\}. \tag{11}$$

In the last step of selecting the best regression stump we determine the coordinate k^* for which the error function (10) takes the minimal value:

$$k^* = \arg \max_{k=1,2,\ldots K} \{e_{i^{*(k)}}^{(k)}\}. \tag{12}$$

In order to speed up the process of selecting θ the computations were conducted using K sorted vectors x. In order to reduce number of the summations during fitting of the regression stumps we employed the cumulative sums of w_j and $w_j y_j$.

5 On-line learning during tracking

The most informative and hard to classify examples are close to the decision boundary between object of interest and background. Therefore an on-line AdaBoost that is employed in our algorithm focuses on hard examples that provide most new information than easy ones. They cause the base learner to concentrate on unseen examples. In this context the major difference of our work from relevant research is that weak classifiers are not trained from the same data sets within rectangles covering the object and background, but only a small portion of the newly available training sets. This makes a difference between our learning based tracking algorithm and algorithms relying on linear adaptation or learning, where updating of the object model is done on the basis of all newly extracted pixels.

An on-line learning algorithm does not need all the training data processed so far to calculate a current hypothesis, rather it process data as it become available without the need for storage, through reusing previously learned weak classifiers to learn new classifier from new data. In our algorithm we train the classifier on the labeled pixels and then apply the classifier to the sampled pixels from unlabeled and newly available ones. The foreground and background pixels are extracted using center-surround approach in which an internal rectangle covers the object, while a larger surrounding rectangle represents the background. The classifier is trained both off-line and on-line using the weak learner that was described in subsection 4.2.

The algorithm that has been proposed in work [4] maintains a list of classifiers that are trained over time. During tracking the algorithm removes old classifiers, trains new classifiers on a confidence map generated by the strong classifier and then adds them to the ensemble. However, through removing the oldest classifiers this algorithm omits important information contained in the initial object template [19] as well it is not able to detect features being stable during tracking. The importance of such stable features during tracking has been highlighted by several authors, among others by [2]. In the algorithm described in [5] the selectors are updated when a new training sample is available. This operation needs considerable computations since the strong classifier contains 50 selectors and each can choose from 250 selectors. This in turn can even lead to slower boosting algorithm in comparison with an off-line algorithm applied to learn on-line. The average number of calculations per feature in this algorithm is far larger than in an off-line AdaBoost.

Before starting the tracking the foreground and background pixels are extracted using center-surround approach. The initial object template is constructed on the basis of the internal rectangle covering the object of interest. A number of representative pixels that are sampled from the object of interest are then utilized during tracking. Such an object template holds information about initial object appearance and prevents from model drift. A strong classifier is used to label the pixels as either belonging to the object of interest or background. On the basis of the confidence map calculated by the strong classifier we sample from the current frame a set of foreground pixels that are hardest to classify. Using a histogram holding information about all pixels seen so far in the object rectangle we extract a set of the most stable pixels in the current frame and add it to the set representing the current frame. Having in disposal a set of pixels from the previous frame that were extracted in the same manner the algorithm considers the temporal coherence between images of object undergoing tracking. The background is represented by pixels laying in close to decision boundary as well as collection of uniformly sampled pixels both from the current and previous frame. During tracking a simple procedure is responsible for removing the pixels belonging to previous frame and inserting the pixels from the new frame as well as maintaining proportions between the mentioned above ingredients of the training vector at possibly the same level. The length of the list containing training pixels is constant.

During boosting iterations the weights that are employed by weak learner are calculated as follows:

$$w \leftarrow w \exp(-y\, f_m) \tag{13}$$

The total score produced by AdaBoost is normalized through soft identity function to range between -1 and 1 in the following manner:

$$s = \tanh(F(x)) = \tanh(\sum_{m=1}^{T} f_m(x)) \tag{14}$$

Such a normalized score can be used as a measure of prediction confidence [20]. Therefore we employ it to determine the object likelihood images.

6 Vision system

The presented algorithm has been integrated into our system providing a support for human-machine interaction. The system consists of a mobile robot Pioneer 2DX, which is equipped with SRI's binocular Megapixel Stereo Head as well as Sony EVI-D31 PTZ (pant/tilt/zoom) camera. Our experimental platform is also supplied with a laser range finder as well as an on-board laptop computer equipped with Intel Core Duo, 2.16 GHz processor. All tasks and threads run asynchronously, but can be synchronized via messages or events. The system can operate autonomously but in a number of configurations, especially when the self-localization of the robot is needed, parallel computations can be realized on external computers with an integration support of Common Object Request Broker Architecture (CORBA). The system is capable to detect human faces, select a person for interaction and then perform several tasks consisting in person following, simultaneous tracking of user's face and hands. On the basis of a 3D-model the user's head pose can be determined as well.

During person following the controller of the active camera keeps the face on desired location in the image. The current pan angle of the camera is used as input for the orientation controller of the robot. This controller in turn minimizes the angle between the axis of the camera and of the robot. The mentioned above control strategy guaranties smooth behaviors of the robot in response to a rapid movement of the tracked person.

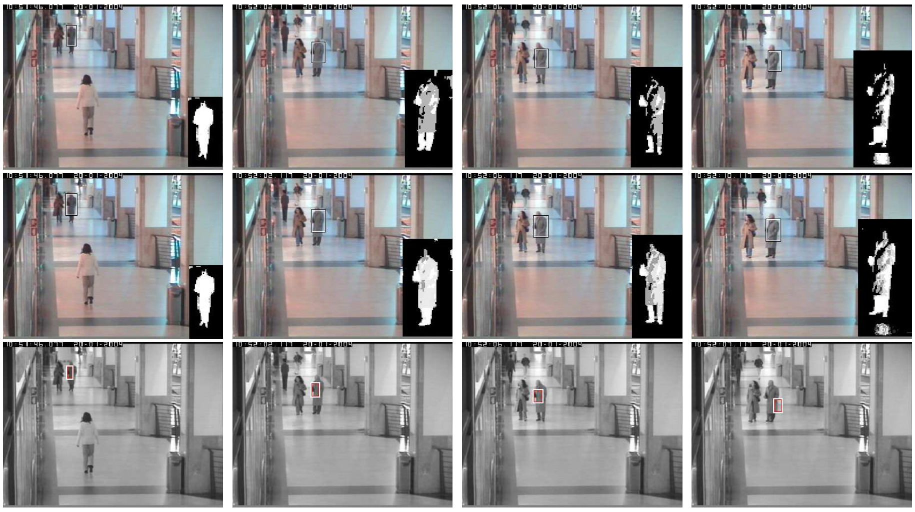
The probability image which is peaked at the correct location of the target allows us to integrate our tracking algorithms. It confirmed also its great usefulness in evaluating performance of different tracking algorithms, but operating on the same set of cues. On the basis of probability images of foreground and background the selection of discriminative histograms takes place.

7 Experiments

To test the proposed method we performed various experiments on real images. Figure 1. depicts some tracking results that were obtained on sequence[1] of images 288 high and 384 pixels wide. It can be observed that our on-line learning algorithm performs better than the algorithm with no learning. The probability images illustrate the potential of our learning algorithm as well as how the confidence maps picks the person's shape over time. In frames that were generated by learning-based algorithm the jitter of rectangular ROI is smaller and it is located near the true location in most frames, see also Fig. 1. and compare images from the last column in rows 1 and 2. Bottom row in Fig. 1. depicts the behavior of the tracking algorithm [2] that has been initialized in the same manner and configured to run with 100 particles. The algorithm looses track of the pedestrian in frame #1226. Figure 2. demonstrates tracking results that were obtained with on-line learning in another sequence[2].

[1] Downloaded from site at: http://groups.inf.ed.ac.uk/vision/CAVIAR/

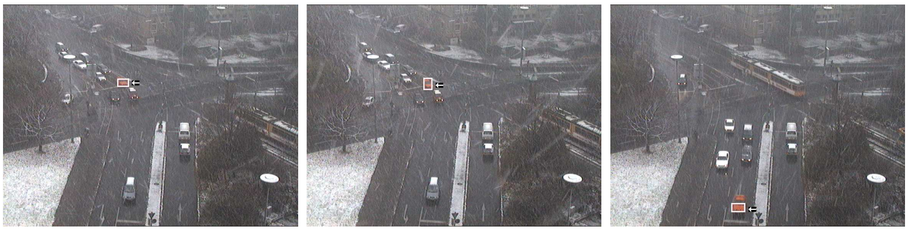
[2] Downloaded from site at: http://i21www.ira.uka.de/image_sequences/



#700 #1100 #1200 #1300/#1226

Fig. 1. Pedestrian tracking with no learning (upper row), on-line learning (middle row, last frame #1300) and using algorithm [2] (bottom row, last frame #1226)

.



#1 #5 #210

Fig. 2. Car tracking with on-line learning on noisy images

Our algorithm is about 2.2 times slower than [2] and runs with 320 × 240 images at about 10 fps on 2.4 GHz Pentium IV. It can be easily extended to run with other features, for example integral images or orientation histograms. A modification consisting in replacing the CamShift by a particle filter operating on the probability images is also straightforward.

8 Conclusions

We have presented an approach for on-line learning during tracking. The elaborated method focuses on hard examples that provide more new information than easy ones. The major difference of our work from relevant research is that weak classifiers are not trained from the same data but only a portion of newly available pixels. To avoid drift the on-line training is conducted using pixels of the object template. During learning we also employ stable pixels seen so far.

Acknowledgment

This work has been supported by Polish Ministry of Education and Science (MNSzW) within the projects 3 T11C 057 30 and N206 019 31/2664.

References

1. Han, B., Comaniciu, D., Zhu, Y., Davis, L.: Incremental density approximation and kernel-based bayesian filtering for object tracking. In: Int. Conf. on Comp. Vision and Pattern Rec., Washington, DC (2004) 638–644
2. Zhou, S.K., Chellappa, R., Moghaddam, B.: Appearance tracking using adaptive models in a particle filter. In: Proc. Asian Conf. on Comp. Vision. (2004)
3. Avidan, S.: Support vector tracking. In: Int. Conf. on Comp. Vision and Pattern Rec., Hawaii (2001) 184–191
4. Avidan, S.: Ensemble tracking. In: Int. Conf. on Comp. Vision and Pattern Rec. (2005) vol. 2, 494–501
5. Grabner, H., Grabner, M., Bischof, H.: On-line boosting and vision. In: Int. Conf. on Comp. Vision and Pattern Rec. (2006) vol. 1, 260– 267
6. Levin, A., Viola, P., Freund, Y.: Unsupervised improvement of visual detectors using co-training. In: Proc. Int. Conf. on Comp. Vision. (2004) vol. 1, 626–633
7. Nair, V., Clark, J.J.: An unsupervised, online learning framework for moving object detection. In: Int. Conf. on Comp. Vision and Pattern Rec. (2004) vol. 2, 317–324
8. Okuma, K., Teleghani, A., Freitas, N.D., Little, J., Lowe, D.G.: A boosted particle filter: Multitarget detection and tracking. In: Proc. ECCV. (2004) 29–39
9. Williams, O., Blake, A., Cipolla, R.: A sparse probabilistic learning algorithm for real-time tracking. In: Int. Conf. on Comp. Vision, Nice, France (2003) 353–360
10. Zhou, S.K., Shao, J., Georgescu, B., Comaniciu, D.: Boostmotion: Boosting a discriminative similarity function for motion estimation. In: Proc. of Int. Conf. on Comp. Vision and Pattern Rec., New York (2006) vol. 2, 1761– 1768
11. Oza, N.C., Russell, S.: Online bagging and boosting. In: 8th Int. Workshop on Artificial Intelligence and Statistics, Morgan Kauffman (2001) 105–112
12. Bradski, G.R.: Computer vision face tracking as a component of a perceptual user interface. In: Proc. IEEE Workshop on Appl. of Comp. Vision. (1998) 214–219
13. Stoessel, D., Sagerer, G.: Kernel particle filter for visual quality inspection from monocular intensity images. In: DAGM06, LNCS, vol. 4174 (2006) 597–606
14. Freund, Y., Schapire, R.: Experiments with a new boosting algorithm. In: Proc. of Int. Conf. on Machine Learning, San Francisco, Morgan Kauffman (1996) 148–156
15. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Int. Conf. on Comp. Vision and Pattern Rec. (2001) vol. 1, 511–518
16. Viola, P., Jones, M., Snow, D.: Detecting pedestrians using patterns of motion and appearance. In: Proc. Int. Conf. on Comp. Vision. (2003) vol. 2, 734–741
17. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Annals of Statistics **38(2)** (2000) 337–374
18. Torralba, A., Murphy, K., Freeman, W.: Sharing features: efficient boosting procedures for multiclass object detection. In: Int. Conf. on Comp. Vision and Pattern Rec. (2004) vol. 2, 762–769
19. Matthews, I., Ishikawa, T., Baker, S.: The template update problem. IEEE Trans. on Pattern Analysis and Machine Intelligence **26** (2004) 810–815
20. Schapiere, R., Freund, Y., Bartlett, P., Lee, W.S.: Boosting the margin: A new explanation for the effectiveness of voting methods. The Annals of Statistics **26(5)** (1998) 1651–1686