MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Ensemble Tracking

Shai Avidan

TR2005-065 December 2005

Abstract

We consider tracking as a binary classification problem, where an ensemble of weak classifiers is trained on-line to distinguish between the object and the background. The ensemble of weak classifiers is combined into a strong classifier using AdaBoost. The strong classifier is then used to label pixels in the next frame as either belonging to the object or the background, giving a confidence map. The peak of the map, and hence the new position of the object, is found using mean shift. Temporal coherence is maintained by updating the ensemble with new weak classifiers that are trained on-line during tracking. We show a realization of this method and demonstrate it on several video sequences.

IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)

Ensemble Tracking

Shai Avidan
Mitsubishi Electric Research Labs
201 Broadway
Cambridge, MA 02139
avidan@merl.com

Abstract

We consider tracking as a binary classification problem, where an ensemble of weak classifiers is trained on-line to distinguish between the object and the background. The ensemble of weak classifiers is combined into a strong classifier using AdaBoost. The strong classifier is then used to label pixels in the next frame as either belonging to the object or the background, giving a confidence map. The peak of the map, and hence the new position of the object, is found using mean shift. Temporal coherence is maintained by updating the ensemble with new weak classifiers that are trained on-line during tracking. We show a realization of this method and demonstrate it on several video sequences.

1 Introduction

Visual tracking is a critical step in many machine vision applications such as surveillance [15], driver assistance systems [1] or human-computer interactions [3]. Tracking works by finding a region in the current image that matches the given object. But if the matching function takes into account only the object, and not the background, then it might not be able to correctly distinguish the object from the background and the tracking might fail.

We treat tracking as a classification problem and train a classifier to distinguish the object from the background. This is done by constructing a feature vector for every pixel in the reference image and training a classifier to separate pixels that belong to the object from pixels that belong to the background. Given a new video frame we use the classifier to test the pixels and form a confidence map. The peak of the map is where we believe the object moved to and we use mean shift [5] to find it.

If the object and background do not change over time then training a classifier when the tracker is initialized would suffice, but when the object and background change their appearance then the tracker must adapt accordingly. Temporal integration is maintained by constantly training new weak classifiers and adding them to the ensemble of weak classifiers. The ensemble thus achieves two goals. Each weak classifier is tuned to separate the object from the background in a particular frame and the ensemble as a whole ensures temporal coherence.

The overall algorithm proceeds as follows. We maintain an ensemble of weak classifiers that is used to create a confidence map of the pixels in the current frame and run mean-shift to find its peak, and hence the new position of the object. Then we update the ensemble by training a new weak classifier on the current frame and adding it to the ensemble.

The proposed method offers several advantages. It breaks the time consuming training phase into a sequence of simple and easy to compute learning tasks that can be performed on-line. It can automatically adjust the weights of different classifiers, trained on different feature spaces. It can also integrate off-line and on-line learning seamlessly. For example, if the object class to be tracked is known then one can train several weak classifiers off-line on large data sets and use these classifiers in addition to the classifiers learned on-line. Finally, integrating classifiers over time improves the stability of the tracker in cases of partial occlusions or illumination changes.

2 Background

Ensemble learning techniques combine a collection of *weak* classifiers into a single *strong* classifier. AdaBoost [9], for example, trains a weak classifier on increasingly more difficult examples and combine the result to produce a strong classifier that is better than any of the weak classifiers.

Treating tracking as a binary classification problem was also addressed by [5] in their mean-shift algorithm, where colors that appear on the object are down-weighted by colors that appear in the background. This was further extended by [4] that use on-line feature selection to switch to the most discriminative color space from a set of different color spaces.

Temporal integration methods include particle filtering [12] to properly integrate measurements over time, the

$\mathcal{WSL}$ tracker [13] that maintains short-term and long-term object descriptors that are constantly updated and re-weighted using on-line-EM, and the incremental sub-space approach [11] in which an adaptive sub-space is constantly updated to maintain a robust and stable object descriptor.

The work most closely related to ours is that of [4] that use on-line feature selection to find the best feature space to work in. We extend their work in several aspects. First, our classification framework automatically weights the different features, as opposed to the discrete nature of feature selection. We depart from histograms as means for generating the confidence map for mean-shift, meaning we can work with high-dimensional feature spaces, as opposed to the low-dimensional feature spaces often used in the mean-shift literature. Finally, by integrating multiple weak classifiers over time we offer a principled manner for temporal fusion.

3 Ensemble Tracking

Ensemble tracking works by constantly updating a collection of weak classifiers to separate the foreground object from the background. The weak classifiers can be added or removed at any time to reflect changes in the object appearance or incorporate new information about the background. Hence, we do not represent an object explicitly, instead we use an ensemble of classifiers to determine if a pixel belongs to the object or not.

Each weak classifier is trained on positive and negative examples where, by convention, we term examples coming from the object as positive examples and examples coming from the background as negative examples. The strong classifier, calculated using AdaBoost, is then used to classify the pixels in the next frame, producing a confidence map of the pixels, where the classification margin is used as the confidence measure. The peak of the map is where we believe the object is, and we use mean shift to find it. Once the detection for the current frame is completed we train a new weak classifier on the new frame, add it to the ensemble, and repeat the process all over again. Figure 1 gives an overview of the system, a general algorithm is given in Algorithm 1.

Algorithm 1 General *Ensemble Tracking*

Input: n video frames $I_1, ..., I_n$
Rectangle r_1 of object in first frame
Output: Rectangles $r_2, ..., r_n$

Initialization (for frame I_1):

- Train several weak classifiers and add them to the ensemble

For each new frame I_j do:

- Test all pixels in frame I_j using the current strong classifier and create a confidence map L_j
- Run mean shift on the confidence map L_j and report new object rectangle r_j
- Label pixels inside rectangle r_j as object and all those outside it as background
- Remove old weak classifiers
- Train new weak classifiers on frame I_j and add them to the ensemble

3.1 The weak classifier

The ensemble tracking framework is a general framework that can be implemented in different ways. We report the particular decisions we made in our system.

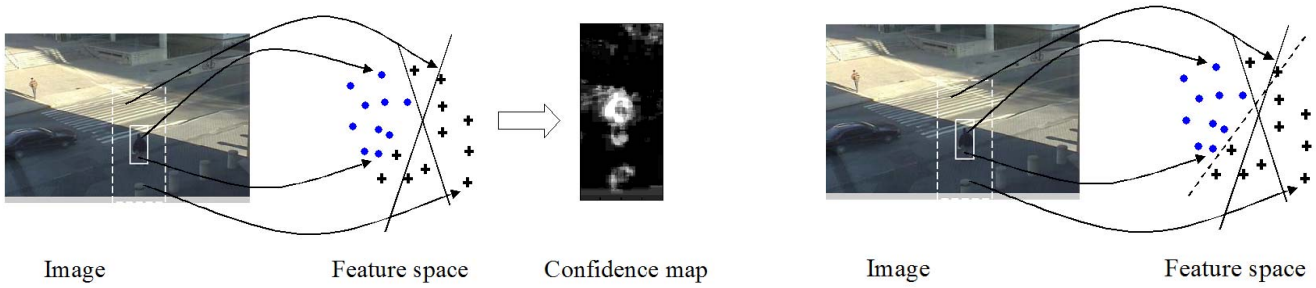
Let each pixel be represented as a d-dimensional feature vector that consists of some local information and let $\{\mathbf{x_i}, y_i\}_{i=1}^{N}$ denote N examples and their labels, respectively, where $\mathbf{x_i} \in \mathcal{R}^d$ and $y_i \in \{-1, +1\}$. The weak classifier is given by $h(\mathbf{x}) : \mathcal{R}^d \rightarrow \{-1, +1\}$, where $h(\mathbf{x})$ is the sign of a linear classifier trained in a least-squares manner. In particular, we use a $11D$ feature vector that is formed by the combination of local orientation histogram and pixel colors. These features are easy to compute and convey rich information for detection purposes as well [14]. Other features, such as the response to filter banks, can be used as well.

Least-squares solutions are slower to compute than color-histograms that are often used in mean shift tracking, however they scale to high dimensions. By using an ensemble of classifiers and running mean shift on their output we indirectly apply mean shift to high-dimensional data. This can be viewed as an alternative to the recently suggested locality-sensitive hashing [10] that maps high dimensional data to a low dimensional space where mean shift is carried out.

Of course, other classifiers, such as stumps (single node decision trees) or perceptrons can be used instead of the least-squares based classifier presented here.

The temporal coherence of video is exploited by maintaining a list of T classifiers that are trained over time. In each frame we discard the oldest weak classifier, train a new weak classifier on the newly available data, and reconstruct the strong weak classifier.

Prior knowledge about the object to be tracked can be incorporated into the tracker as one or more weak classifiers that participate in the strong classifier, but can not be removed in the update stage.



(a) (b)

Figure 1: Ensemble update and test. (a) The pixels of image at time $t-1$ are mapped to a feature space (circles for positive examples, crosses for negative examples). Pixels within the solid rectangle are assumed to belong to the object, pixels outside the solid rectangle and within the dashed rectangle are assumed to belong to the background. The examples are classified by the current ensemble of weak classifiers (denoted by the two separating hyper-planes). The ensemble output is used to produce a confidence map that is fed to the mean shift algorithm. (b) Now we train a new weak classifier (the dashed line) on the pixels of the image at time t and add it to the ensemble.

Here we use the same feature space across all classifiers, but this does not have to be the case. Fusing various cues [6, 7] was proved to improve tracking results and ensemble tracking provides a flexible framework to do so.

The margin of the weak classifier $h(\mathbf{x})$ is mapped to a confidence measure $c(\mathbf{x})$ by clipping negative margins to zero and re-scaling the positive margins to the range $[0, 1]$. The confidence value is then used in the confidence map that is fed to the mean shift algorithm. The specific algorithm we use is given in Algorithm 2.

3.2 Ensemble update

In the update state, the algorithm removes K old weak classifiers to make room for K new weak classifiers. However, before adding new weak classifiers one needs to update the weight of the remaining weak classifiers. This can be done either by forming a least-squares problem in which the weights are unknown and the weak classifiers are known, or by running AdaBoost. We chose the latter. Step (7) of Algorithm 2 in the update state updates the weights of the remaining weak classifier. This is done by changing the role of the weak learner. Instead of training a new weak classifier, the weak learner simply hands AdaBoost one weak classifier (from the existing set of weak classifiers) at a time. This saves training time and creates a strong classifier as well as a sample distribution that can be used for training the new weak classifier, as is done in step (8).

Care must be taken when adding or re-weighting a weak classifier that do not perform much better than chance. If, during weight re-calculation, the weak classifier performs worse than chance then we set its weight to zero. During step (8), we require the new weak classifier to perform significantly better than chance. Specifically, we abort the loop

(a) (b) (c)

Figure 2: Outlier rejection. (a) The input image. The solid rectangle marks the object, the dashed one marks the background. (b) The confidence map with outlier rejection. (c) confidence map without outlier rejection. The outlier rejection process produces cleaner confidence maps that lead to a more stable tracking process. The confidence maps correspond to the dashed rectangle.

in step (8) of the steady state in Algorithm 2 if err, calculated in step (8c), is above some threshold, which is set to 0.4 in our case. This is especially important in case of occlusions or severe illumination artifacts where the weak classifier might learn data that does not belong to the object but rather to the occluding object or to the illumination.

3.3 Outlier rejection

If the object to be tracked is not a pure rectangle then the bounding box that we use for tracking will include some pixels that are labeled as positive, while in fact they should be labeled negative. It was shown that AdaBoost is sensitive to outliers [8] and hence an outlier rejection scheme is needed. A simple approach is to treat too "difficult" examples as outliers and change their label.

Specifically, step (4) of the steady state in Algorithm 2

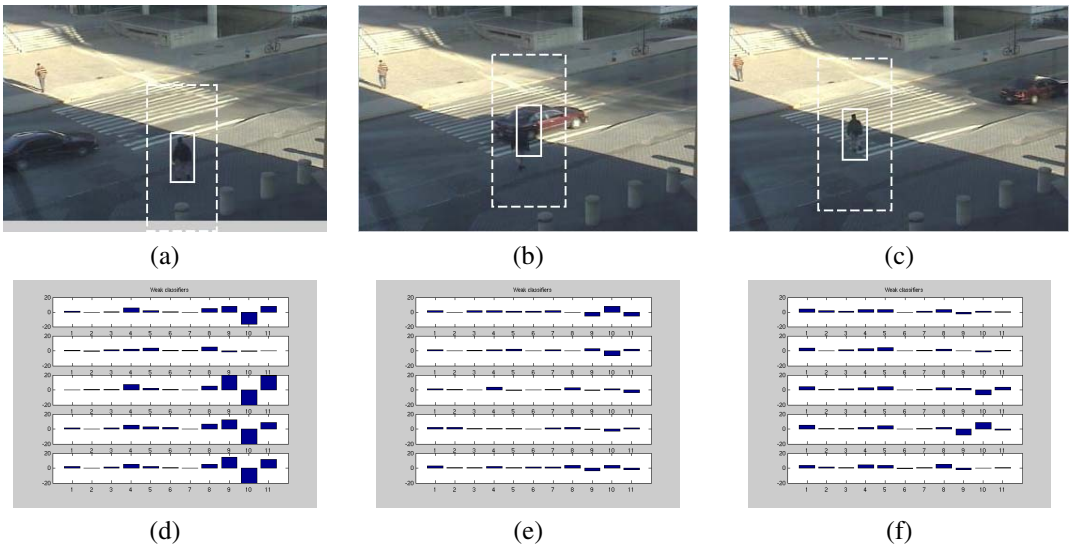


Figure 4: Adapting the weak classifiers. Top row shows frames 10, 40 and 70 from a 100-long video sequence. Bottom row shows the ensemble classifiers used in each frame. There are five weak classifiers for each frame, shown in reverse temporal order (i.e. top classifier was trained on the current frame, the one below it was trained on the previous frame and so on). The first 8 bins of each classifier are of a 5×5 local orientation histogram calculated around each pixel, the last three bins are of the pixel color. The magnitude of the bars indicate the weight of the feature. As can be seen, the color (right-most three bars) plays an important role in the tracking, but when the pedestrian stands in front the of the car, the weight of the oriented edges increase to provide better object/background separation.

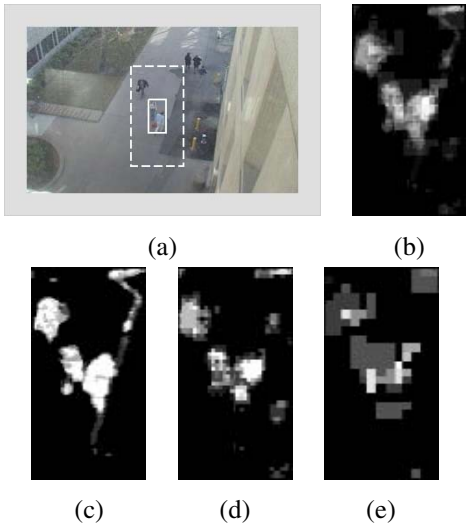


Figure 3: Integrating multi-scale confidence maps. Combining features across multiple scales improves the object/background separation. (a) input image with the solid rectangle defining the object and dashed rectangle defining the background region. (b) The confidence map computed as a weighted average of the confidence maps (c-e). (c-e) are confidence maps that are computed on different levels of the image pyramid. (c) confidence map of original image. (d) confidence map of half-size image. (e) confidence map of quarter-size image. The confidence maps correspond to the dashed rectangle.

can be written as follows:

$$y_i = \begin{cases} +1 & inside(r_j, p_i) \\ -1 & otherwise \end{cases}$$

where r_j is the current rectangle, p_i is the pixel position of example i and $inside(r, p)$ is a predicate that is true if pixel p is inside rectangle r. The outlier rejection version will look as follows:

$$y_i = \begin{cases} +1 & inside(r_j, p_i) \ \wedge (w_i < \Theta) \\ -1 & otherwise \end{cases}$$

where w_i is the weight of the pixel p_i after running the strong classifier and Θ is some predefined threshold which, in our case, is set to $\Theta = \frac{3}{N}$, where N is the number of examples. That is, pixels inside the rectangle are assumed to be positive examples, unless they are too "difficult" to classify and then their label is changed to negative.

Figure 2 show the contribution of the outlier rejection process. The confidence maps are much cleaner, leading to a better and more stable tracking.

3.4 Multi-resolution tracking

We run ensemble tracking in a multi-scale framework. This enables the tracker to capture feature at multiple scales. For each level of the pyramid we run an independent ensemble

Algorithm 2 Specific *Ensemble Tracking*

Input: n video frames $I_1, ..., I_n$
Rectangle r_1 of object in first frame
Output: Rectangles $r_2, ..., r_n$

Initialization (for frame I_1):

1. Extract $\{\mathbf{x_i}\}_{i=1}^N$ examples with labels $\{y_i\}_{i=1}^N$
2. Initialize weights $\{w_i\}_{i=1}^N$ to be $\frac{1}{N}$
3. For $t = 1...T$,
   (a) Make $\{w_i\}_{i=1}^N$ a distribution
   (b) Train weak classifier h_t
   (c) Set $err = \sum_{i=1}^N w_i |h_t(\mathbf{x_i}) - y_i|$
   (d) Set weak classifier weight $\alpha_t = \frac{1}{2} log \frac{1-err}{err}$
   (e) Update example weights $w_i = w_i e^{(\alpha_t |h_t(\mathbf{x_i}) - y_i|)}$
4. The strong classifier is given by $sign(H(\mathbf{x}))$ where $H(x) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$

For each new frame I_j do:

1. Extract $\{\mathbf{x_i}\}_{i=1}^N$ examples
2. Test the examples using the strong classifier $H(\mathbf{x})$ and create confidence image L_j
3. Run mean-shift on L_j with r_{j-1} as the initial guess. Let r_j be the result of the mean shift algorithm
4. Define labels $\{y_i\}_{i=1}^N$ with respect to the new rectangle r_j
5. Remove K oldest weak classifiers
6. Initialize weights $\{w_i\}_{i=1}^N$ to be $\frac{1}{N}$
7. For $l = K+1...T$, (Update weights)
   (a) Make $\{w_i\}_{i=1}^N$ a distribution
   (b) Choose $h_t(\mathbf{x})$, with minimal error err, from $\{h_{K+1}(\mathbf{x}), ..., h_T(\mathbf{x})\}$
   (c) update α_t and $\{w_i\}_{i=1}^N$
   (d) Remove $h_t(\mathbf{x})$ from $\{h_{K+1}(\mathbf{x}), ..., h_T(\mathbf{x})\}$
8. For $t = 1...K$, (Add new weak classifiers)
   (a) Make $\{w_i\}_{i=1}^N$ a distribution
   (b) Train weak classifier h_t
   (c) Compute err and α_t
   (d) Update example weights $\{w_i\}_{i=1}^N$
9. The updated strong classifier is given by $sign(H(\mathbf{x}))$ where $H(x) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$

tracking that outputs a confidence map. The maps are then combined to form a single confidence map that is used in the mean shift step.

Specifically, in each frame we train a weak classifier for each pyramid level, and maintain one strong classifier for each such level. Each strong classifier generates a confidence map and all the confidence maps are resized to the size of the original image and averaged to form the confidence map that is used by the mean shift algorithm.

Figure 3 shows a typical confidence map, accumulated across multiple scales. We computed a confidence for the original, half-size and quarter-size images, then we rescaled all confidence maps to the same size and combined them based on the classification score of the classifier at each level.

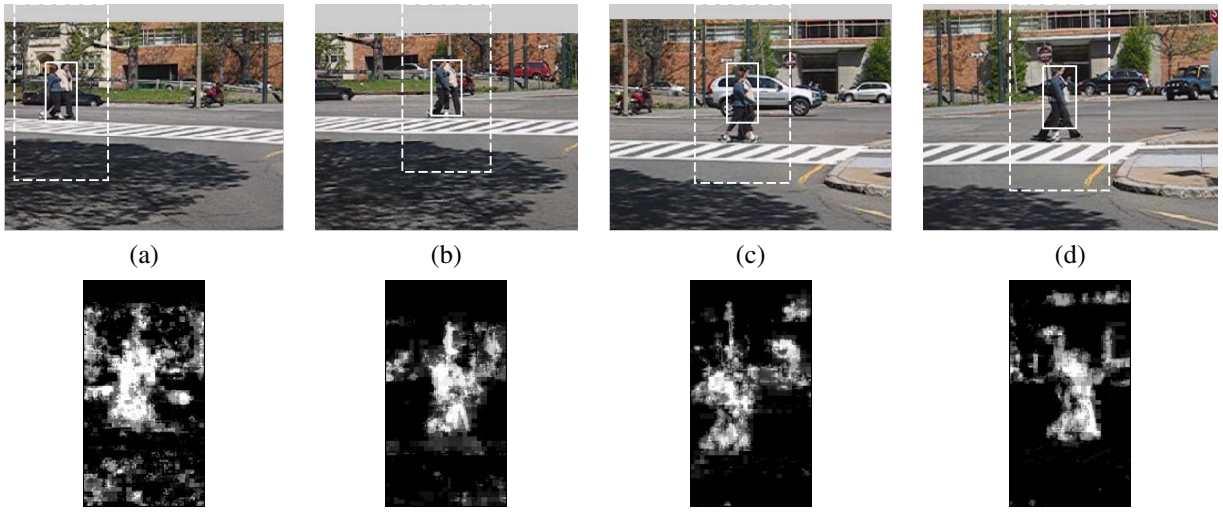
4 Experiments

We implemented the proposed method in Matlab and tested it on several video sequences. No parameters were changed from one experiment to the next and in all cases the initial rectangle was supplied manually. In all cases we use a $11D$ feature vector per pixel that consists of an 8-bin local orientation histogram calculated on 5×5 window as well as the pixel R, G and B values. To improve robustness we only count edges that are above some predefined threshold, which in our case was set to 10 intensity values. A similar approach was taken in [14] for the problem of face detection. We run the tracker, in parallel, on three levels of the pyramid, combine the confidence maps and run mean-shift on the resultant confidence map. In each frame we train one weak classifier (i.e. $K = 1$). The algorithm runs at a few frames per second. In all cases we never use a static background assumption and allow the camera to move freely.

The first experiment is on a video sequence of a pedestrian crossing the street. Halfway through the sequence the pedestrian is standing in front of a car that has the same color as he does. The tracker manages to track the pedestrian through the entire sequence. Figure 4 shows several frames from the sequence. The top row shows the actual images, while the bottom row shows the weak classifier behavior (for the bottom level of the pyramid only). Recall that the feature vector consists of an 8-bin local orientation histogram, followed by the R, G and B colors of each pixel. As can be seen, at first the color features are prominent in the classification, but as the background changes, so are the classifiers and the role of the orientation histogram increases.

In the second experiment we track a couple walking with a hand-held camera. Figure 5 show several frames from this 80-frame long sequence.

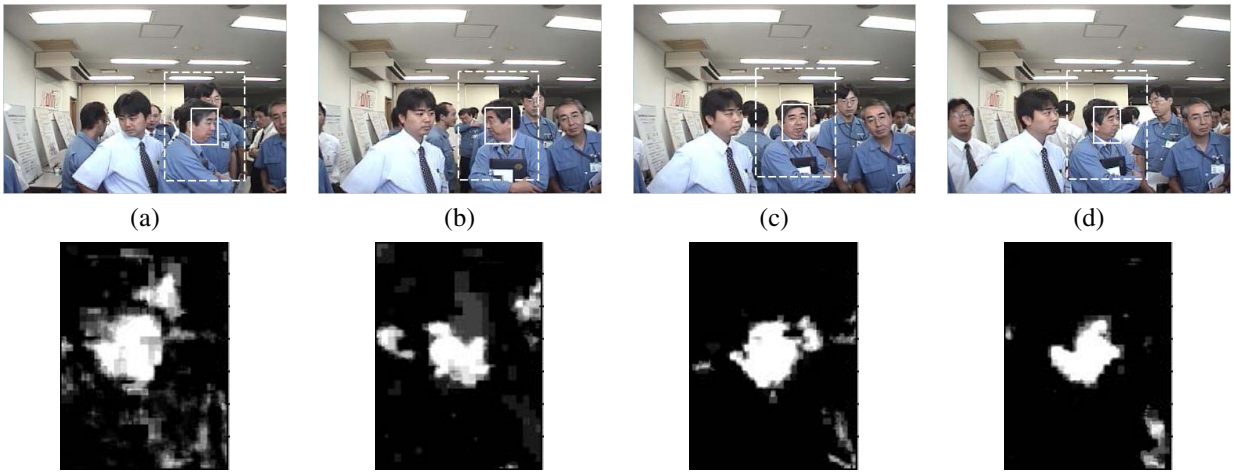
In the third experiment we track a face exhibiting out-of-plane rotations. Figure 6 show several frames from this



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 5: Ensemble Tracking with a moving camera. (a-d) Frames 0,40,68 and 80 from a 80-frame long sequence. (e-h) The confidence map for each frame. The confidence maps correspond to the dashed rectangle.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 6: Ensemble Tracking. (a-d) Frames 0,20,40 and 70 from a 90-frame long sequence. (e-h) The confidence map for each frame. The confidence maps correspond to the dashed rectangle.

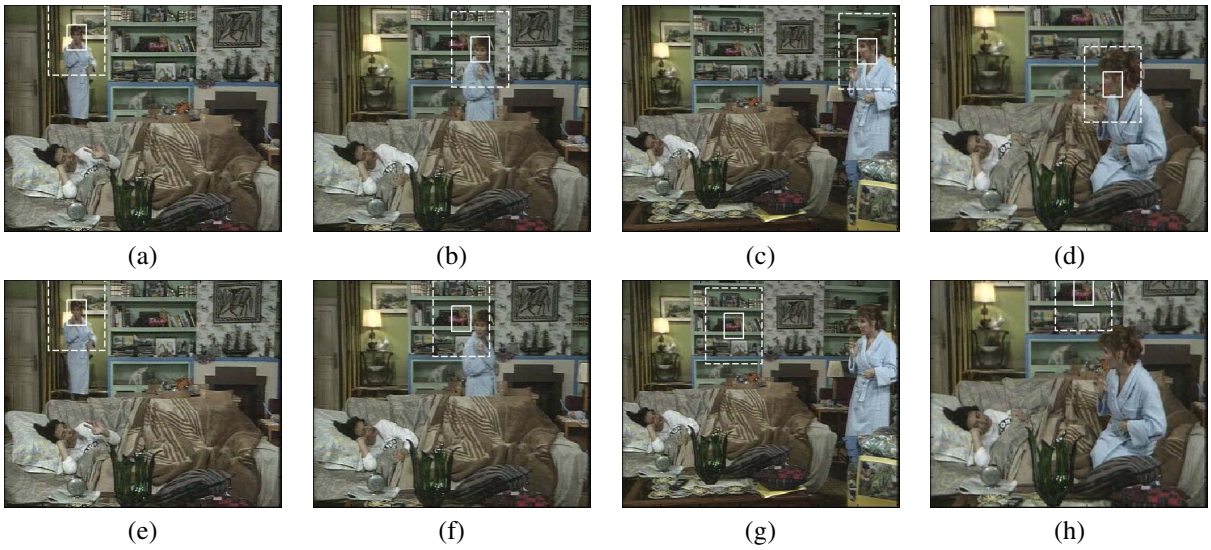


Figure 7: Ensemble Tracking with and without update. Tracking *with* weak classifier update (a-d). Tracking *without* weak classifier update (e-h). In the latter case, we train 5 weak classifiers on the first frame and never update them. In the former case, we update the weak classifier according to the scheme presented in this paper.

90-frame long sequence.

Next, we compared the importance of the update scheme for tracking. Figure 7 show the results of two trackers on the same sequence. In the first case we use an "adaptive" tracker based on the framework presented in this paper. In the second case we use a "static" tracker that trains five weak classifiers on the first frame of the sequence and fix it for the entire length of the sequence. At frame 30 the "static" tracker locks on the background while the "adaptive" tracker keeps tracking successfully.

Finally, the last sequence is a gray scale sequence [1], not color, and we track a car over 225 frames. The feature space, accordingly, is $9D$ feature space (the 8-bin local orientation histogram and the gray scale intensity value). Gray scale images are usually difficult to track using traditional mean-shift algorithms because a single color channel does not provide enough information for tracking. However this did not prove a problem for our system. Some of the frames can be seen in figure 8.

5 Conclusions

We treat tracking as a binary classification problem. An ensemble of weak classifiers is trained on-line to distinguish between features of the object and features of the background. We form a strong classifier from the ensemble using AdaBoost. The strong classifier is then used to compute a confidence map of the next frame. The peak of the map, and hence the new position of the object, is found using mean shift algorithm. The tracker adjusts to appearance changes by training a new weak classifier per frame and updating the strong classifier, giving robustness to the tracker at a low computational cost.

[1] Downloaded from the Karlsruhe university site at: http://i21www.ira.uka.de/image_sequences

References

[1] Avidan S., Support Vector Tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2004.

[2] Black, M. J. and Jepson, A. EigenTracking: Robust matching and tracking of articulated objects using a view-based representation. International Journal of Computer Vision, 26(1), pp. 63-84, 1998.

[3] Bobick, A., S.Intille, J.Davis, F.Baird, C.Pinhanez, L.Campbell, Y.Ivanov, A.Schutte, and A.Wilson. The KidsRoom. In *Communications of the ACM*, 43(3). 2000

[4] Collins T. R., Liu, Y. On-Line Selection of Discriminative Tracking Features. *Proceedings of the International Conference on Computer Vision* (ICCV '03), France, 2003.

[5] Comanciu, D., Visvanathan R., Meer, P. Kernel-Based Object Tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (PAMI), 25:5, pp 564-575, 2003.

[6] Crowley, J., Berard, F. Multi-Modal Tracking of Faces for Video Communications. *Proceedings of the Con-*

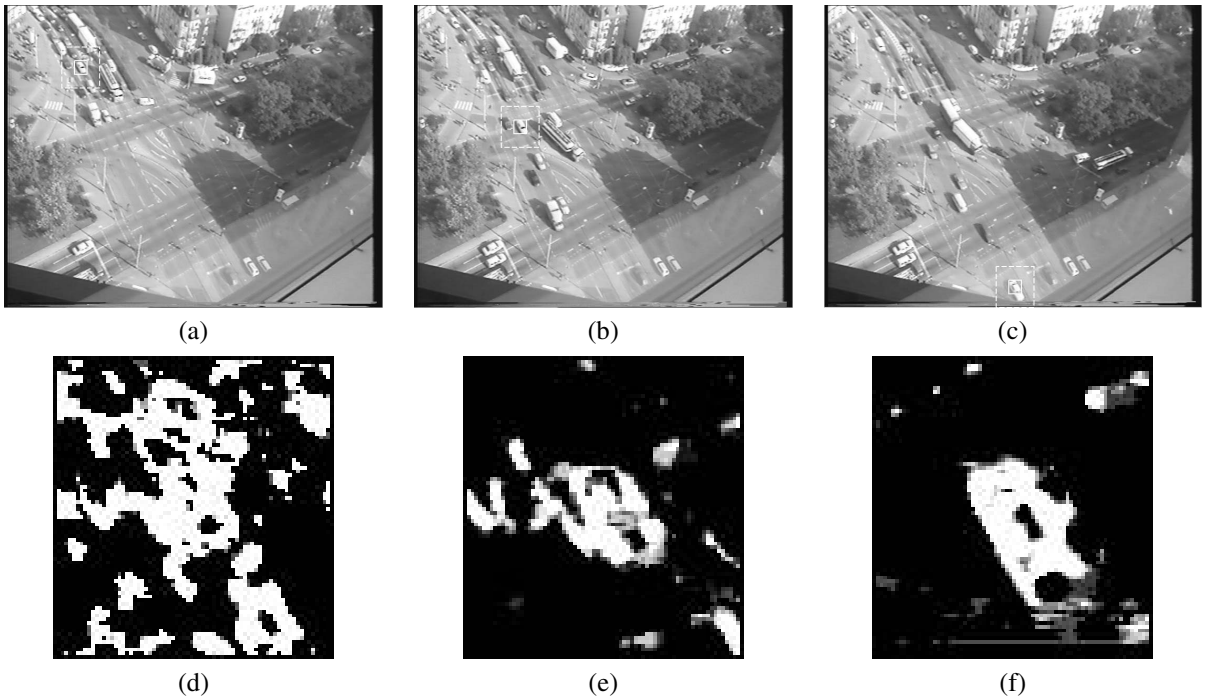


Figure 8: Ensemble Tracking. (a-e) Frames 0,100 and 225 from a 225-frame long sequence. We track a car from the upper left part of the image to the middle bottom of the image. (d-f) The confidence map for each frame. The confidence map corresponds to the dashed rectangle. Note how the confidence map picks the car's shape over time.

ference on Computer Vision and Pattern Recognition (CVPR '97), Puerto Rico 1997.

[7] Darrell, T., Gordon, G., Harville, M., and Woodfill, J. Integrated person tracking using stereo, color, and pattern detection. *Proceedings of the Conference on Computer Vision and Pattern Recognition* (CVPR '98), pp. 601-609, Santa Barbara, June 1998.

[8] Freund, Y. An adaptive version of the boost by majority algorithm. In *Machine Learning*, 43(3):293-318, June 2001.

[9] Freund, Y. Schapire, R. E. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Eurocolt* 95, pp 23-37, 1995.

[10] Georescu, B., Shimshoni, I., Meer, P. Mean Shift Based Clustering in High Dimensions: A Texture Classification Example. *Proceedings of the International Conference on Computer Vision* (ICCV '03), France, 2003.

[11] Ho, J., Lee K., Yang, M., Kriegman, D. Visual Tracking Using Learned Linear Subspaces. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2004.

[12] Isard, M., Blake, A. CONDENSATION - Conditional Density Propagation for Visual Tracking, *International Journal of Computer Vision*, Vol 29(1), pp:5-28, 1998.

[13] Jepson, A.D., Fleet, D.J. and El-Maraghi, T. Robust. on-line appearance models for vision tracking. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25(10):1296-1311.

[14] Levi, K., Weiss, Y. Learning Object Detection from a Small Number of Examples: The Importance of Good Features. In *IEEE Conf. on Computer Vision and Pattern Recognition*, 2004.

[15] Stauffer, C. and E. Grimson, *Learning Patterns of Activity Using Real-Time Tracking*, PAMI, 22(8):747-757, 2000.