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.

Index Terms

AdaBoost, Visual Tracking, Video Analysis, Concept Learning.

Ensemble Tracking

Shai Avidan

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.

Index Terms

AdaBoost, Visual Tracking, Video Analysis, Concept Learning.

I. Introduction

Visual tracking is a critical step in many machine vision applications such as surveillance [22], driver assistance systems [1] or human-computer interactions [3]. Tracking finds 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 [6] 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.

Ensemble tracking extends traditional mean-shift tracking in a number of important directions. First, mean-shift tracking usually works with histograms of RGB colors. This is because gray-scale images do not provide enough information for tracking and high-dimensional feature spaces can not be modeled with histograms due to exponential memory requirements. By switching to general machine learning classifiers, ensemble tracking avoid both pitfalls. It can handle gray-scale images, by introducing local neighborhood information, and it does not suffer from exponential memory explosion because it is no longer restricted to working with histograms, as it can work with any type of classifier. Second, ensemble tracking gives a principled manner in which the classifiers are integrated over time. This is in contrast to existing methods that either represent the foreground object using the most recent histogram, or some ad-hoc combination of the histograms of the first and last frames.

In addition, 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. Also, integrating classifiers over time improves the stability of the tracker in cases of partial occlusions or illumination changes. Finally, on a higher level, one can view ensemble tracking as a method for training classifiers on time-varying distributions.

II. Background

Ensemble learning techniques combine a collection of *weak* classifiers into a single *strong* classifier. AdaBoost [13], 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 already considered in the past. Lin *et al.* [20] suggest an adaptive discriminative generative model where a Fisher Linear Discriminant function is constantly evaluated to discriminate the object from the background. A similar approach was taken by Nguyen *et al.* [21]. Comaniciu *et al.* [6] adopt this approach to 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 Collins *et al.* [5] 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 [16] to properly integrate measurements over time, the $\mathcal{WSL}$ tracker [17] 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 [15] in which an adaptive sub-space is constantly updated to maintain a robust and stable object descriptor.

It is instructive to compare these methods to ours. The $\mathcal{WSL}$ and incremental sub-space methods can be viewed as *generative* methods that aim to explain the foreground object while ignoring the background. Also, these methods are template based, meaning that they maintain spatial integrity of the object and thus are especially suited for handling rigid objects. Ensemble tracking, on the other hand, maintains an implicit representation of the foreground and the background, through the use of the classifiers. In addition, ensemble tracking works on a pixel level so global spatial relationships are not maintained. This is useful when the object deforms or undergoes severe appearance changes. Particle filtering maintains a probability distribution function over state space (i.e. what are the locations the object can be and what are the probabilities associated with each such hypothesis). This means that particle filtering can be used in conjunction with ensemble tracking, where the latter is used to form the measurements (i.e. the confidence map) that are used by the former.

A similar problem, termed “concept drift”, is considered in the data mining literature where the goal is to quickly scan large volumes of data and learn a concept (“object” in computer vision jargon). As the concept might drift the classifier must adapt as well. For example, [18] present “dynamic weighted majority” as a method to track concept drift for data mining applications, while [4] add change detection to concept drift to detect abrupt changes in the concept, much in the spirit of the $\mathcal{WSL}$ tracker [17].

The work most closely related to ours is that of [5] that use on-line feature selection to find the best feature space to work in. We extend their work in a number of important ways. First, our classification framework automatically weights the different features, as opposed to the discrete nature of feature selection. Second, 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, our ensemble tracking technique gives a general way of adaptively building discriminant functions over time varying distributions.

III. Ensemble Tracking

Ensemble tracking constantly updates 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 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.

Another way to look at ensemble tracking is to consider it as a method for building, and maintaining, a discriminant function over time varying distributions. In this case we deal with distributions of object and background pixels, but ensemble tracking can be used in other scenarios as well.

Our method constructs an ensemble classifier on-line. This bags the question what guarantees, if any, do we have on its errors over the training set as well as its generalization error? AdaBoost assumes a static distribution and an access to a weak learner that performs better than chance on this distribution. Ensemble tracking, on the other hand, assumes time-varying distributions. However, because we are dealing with video, we assume that the distribution changes slowly so past weak classifiers still perform better than chance on the new data which gives error bounds on the test error of ensemble tracking. In practice, AdaBoost was shown to perform much better than predicted by the theoretical analysis and we found the same to be true with our ensemble tracking algorithm.

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 $T$ 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
  • Keep $K$ “best” weak classifiers
  • Train new $T - K$ weak classifiers on frame $I_j$ and add them to the ensemble
```

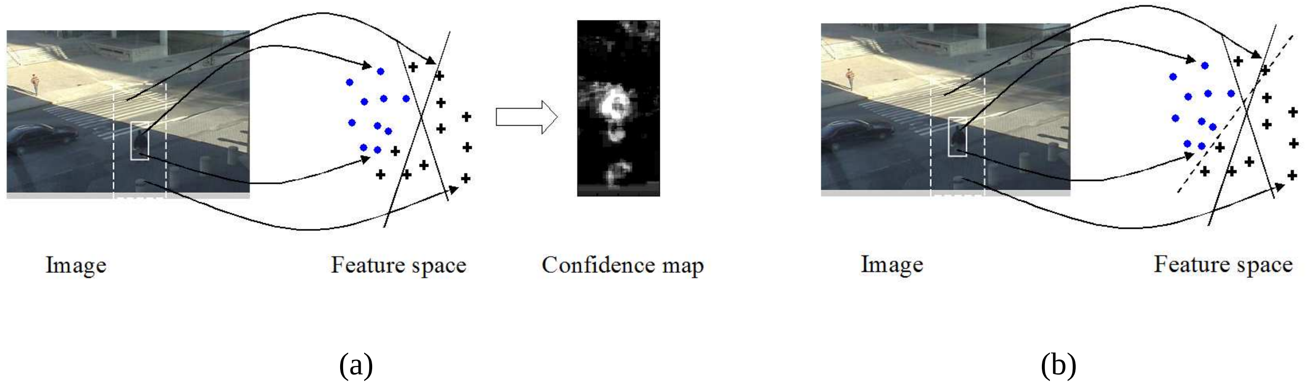


Fig. 1. Ensemble update and test. (a) The pixels of image at time $t-1$ are mapped to a feature space (circles for positive examples, crosses for negative examples). Pixels within the solid rectangle are assumed to belong to the object, pixels outside the solid rectangle and within the dashed rectangle 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.

A. 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 $h(\mathbf{x}) : \mathcal{R}^d \rightarrow \{-1, +1\}$ that is defined as:

$$h(\mathbf{x}) = sign(\mathbf{h}^T\mathbf{x})$$

where $\mathbf{h} \in \mathcal{R}^d$ is a separating hyperplane that is computed using weighted least square regression

$$\mathbf{h} = (\mathbf{A}^T\mathbf{W}\mathbf{A})^{-1}\mathbf{A}^T\mathbf{W}\mathbf{y}$$

Each row of the matrix $\mathbf{A}$, denoted $\mathbf{A_i}$, corresponds to one example $\mathbf{x_i}$ augmented with the constant 1, that is $\mathbf{A_i} = [\mathbf{x_i}, 1]$ and $\mathbf{W}$ is a diagonal matrix of the weights. We found it useful to scale the sum of weights of positive, as well as negative, examples to be equal to 0.5. This prevents bias to the negative examples if the area of the object is smaller that that of the background.

The temporal coherence of video is exploited by maintaining a list of T classifiers that are trained over time. In each frame we keep the K "best" weak classifiers, discard the remaining $T-K$ weak classifiers, train $T-K$ new weak classifiers 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 in the form of one or more weak classifiers that participate in the strong classifier, but can not be removed in the update stage.

Here we use the same feature space across all classifiers, but this does not have to be the case. Fusing various cues [7], [8] 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.

B. *Ensemble update*

In the update state, the algorithm keeps the "best" K weak classifiers, thus making room for $T-K$ new weak classifiers. However, before adding the new weak classifiers one needs to update the weight of the remaining K weak classifiers. This is done is step (7) of Algorithm 2. Instead of training a new weak classifier, the weak learner simply hands AdaBoost one weak classifier (from the existing set of T weak classifiers) at a time. By repeating this process K times we effectively choose the best K weak classifiers from the current ensemble of T classifiers. 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 in step (8) of the steady state in Algorithm 2 if err, calculated in

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) Keep best $K$ weak classifiers
  6) Initialize weights $\{w_i\}_{i=1}^N$ to be $\frac{1}{N}$
  7) For $t = 1...K$,     (Choose $K$ best classifiers and update their weights)
       a) Make $\{w_i\}_{i=1}^N$ a distribution
       b) Choose $h_t(\mathbf{x})$, with minimal error $err$, from $\{h_1(\mathbf{x}), ..., h_T(\mathbf{x})\}$
       c) update $\alpha_t$ and $\{w_i\}_{i=1}^N$
       d) Remove $h_t(\mathbf{x})$ from $\{h_1(\mathbf{x}), ..., h_T(\mathbf{x})\}$
  8) For $t = K+1...T$,     (Add new weak classifiers)
       a) Make $\{w_i\}_{i=1}^N$ a distribution
       b) Train weak classifier $h_t$
       c) Compute $err$ and $\alpha_t$
       d) Update example weights $\{w_i\}_{i=1}^N$
  9) The updated strong classifier is given by $sign(H(\mathbf{x}))$ where $H(x) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$
```

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.

Note that even during step (7), of choosing the K best weak classifier, we might encounter a case where some of the existing weak classifiers do not perform much better than chance. We allow up to two existing weak classifiers to be removed this way because a larger number might be the sign of occlusion and hence we keep the ensemble unchanged for this frame.

IV. Implementation issues

There are several implementation issues that we found helpful in tracking.

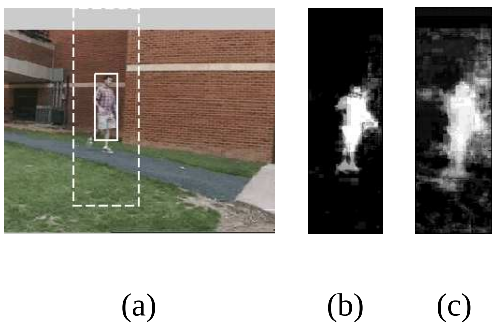


Fig. 2. Outlier rejection. (a) The input image. The solid rectangle marks the object, the dashed one marks the background. (b) The confidence map with outlier rejection. (c) confidence map without outlier rejection. The outlier rejection process produces cleaner confidence maps that lead to a more stable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

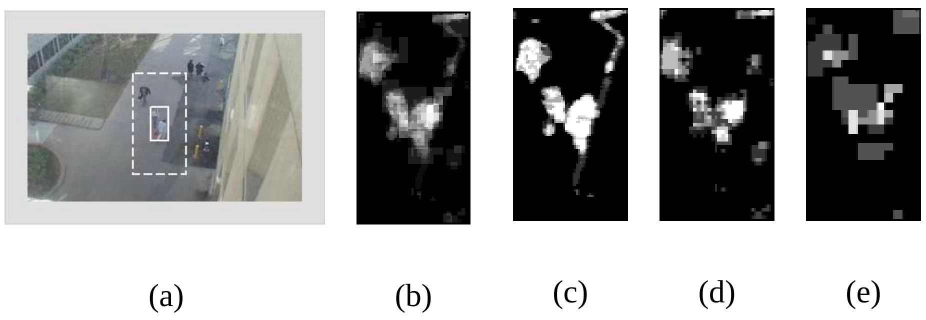


Fig. 3. Integrating multi-scale confidence maps. Combining features across multiple scales improves the object/background separation. (a) input image with the solid rectangle defining the object and dashed rectangle defining the background region. (b) The confidence map computed as a weighted average 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.

A. 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 [9] 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 can be written as follows:

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

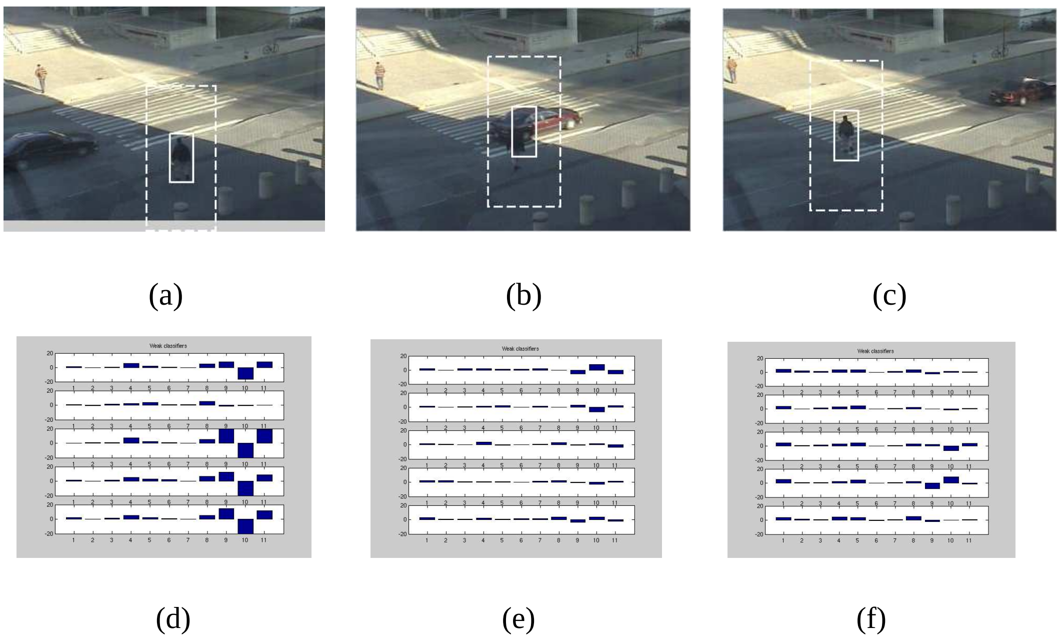


Fig. 4. Adapting the weak classifiers. Top row shows frames 10, 40 and 70 from a 100-long video sequence. Bottom row shows the ensemble classifiers used in each frame. There are five weak classifiers for each frame, shown in reverse temporal order (i.e. top classifier was trained on the current frame, 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 histogram of oriented gradients 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.

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.

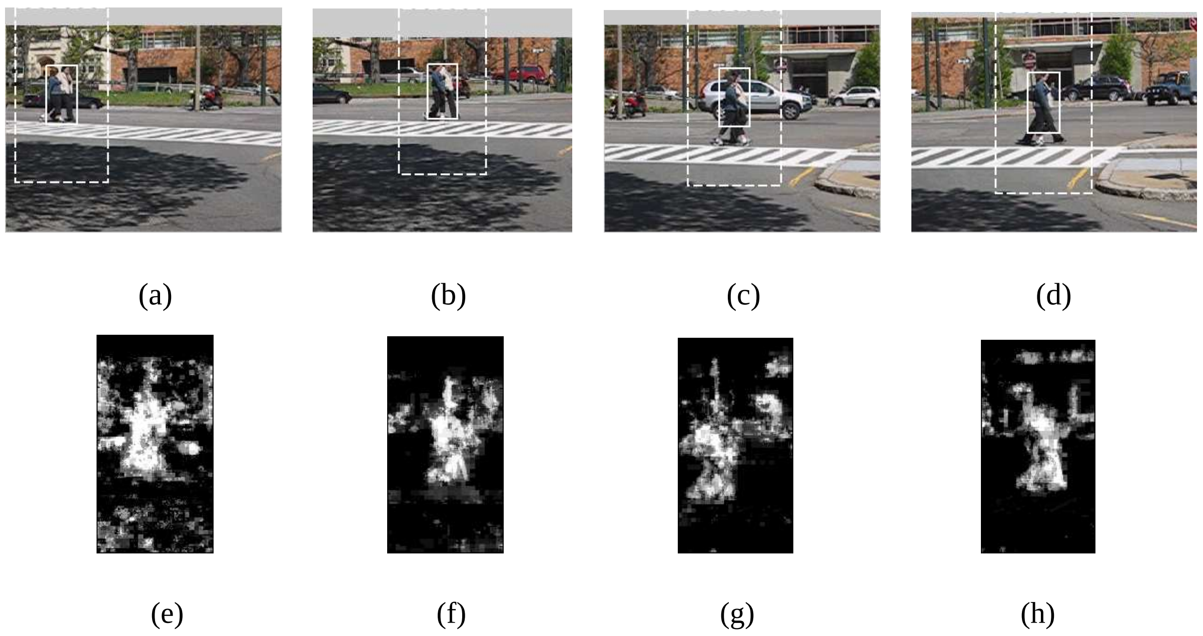


Fig. 5. Ensemble Tracking with a moving camera. (a-d) Frames 0,40,68 and 80 from a 80-frame long sequence. (e-h) The confidence map for each frame. The confidence maps correspond to the dashed rectangle.

B. Multi-resolution tracking

We run ensemble tracking in a multi-scale framework. This enables the tracker to capture features at multiple scales. For each level of the pyramid we run an independent ensemble tracking that outputs a confidence map. The maps are then combined to form a single confidence map that is used by the mean shift tracker.

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.

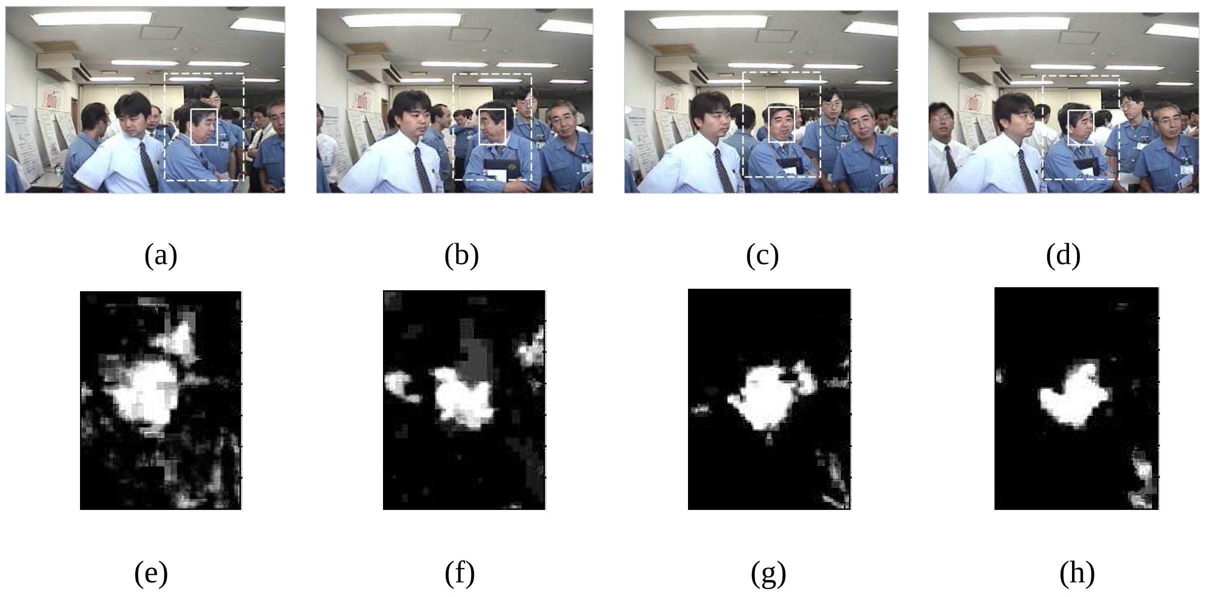


Fig. 6. Ensemble Tracking. (a-d) Frames 0,20,40 and 70 from a 90-frame long sequence. (e-h) The confidence map for each frame. The confidence maps correspond to the dashed rectangle.

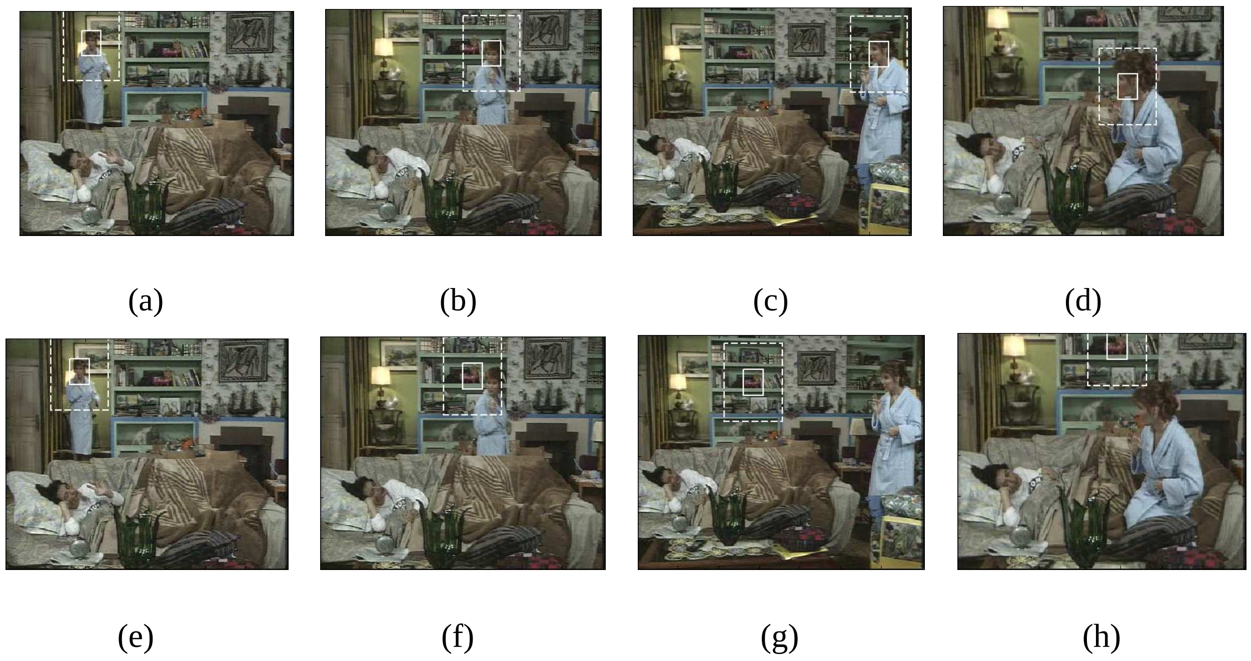


Fig. 7. Ensemble Tracking with and without update. Tracking *with* weak classifier update (a-d). Tracking *without* weak classifier update (e-h). In the latter case, we train 5 weak classifiers on the first frame and never update them. In the former case, we update the weak classifier according to the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

V. 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. We experimented with several different feature spaces. The first version uses 5 weak classifiers each working on an $11D$ feature vector per pixel that consists of an 8-bin local histogram of oriented gradients 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. The histogram of oriented gradients is easy to compute and convey rich information that was used in the past for detection and recognition purposes [11], [19], [10]. Other features, such as the response to filter banks, can be used as well.

In the traffic and zodiac sequences, a gray scale and IR sequences presented later in this section, we found that the original feature space was not stable enough and used a non-linear version of that feature space instead. The non-linear feature space is defined as $[\mathbf{x_i}, \mathbf{x_i}^2, \mathbf{x_i}^3]$, where $\mathbf{x_i}$ is the original feature vector and $\mathbf{x_i}^d$ is taken to be a shorthand for raising each element of the vector $\mathbf{x_i}$ to the power d. We found this to be a cheap way of introducing non-linear kernel-like performance into the system. Of course, other non-linear classifiers can be used as well, provided they can work in real time. With the non-linear feature vector we used only 3, instead of 5, weak classifiers.

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 drop one weak classifier and add a newly trained weak classifier. We allow the tracker to drop up to two weak classifiers per frame, because dropping more than that might be a sign of occlusion and we therefor do not update the ensemble in such a case. The algorithm runs at a few frames per second. Currently we use every pixel of the object and background for the ensemble update. This can probably be greatly accelerated if we sample the pixels (because the feature vector associated with each pixel already captures some local information) or if we ignore samples with low weight, as was suggested by Friedman *et al.* [12]. In all cases we never use a static background assumption and allow the camera to move freely.

A. Results on color sequences

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 histogram of oriented gradients, 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 histogram of oriented gradients 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 90-frame long sequence.

In the next experiment, shown in figure 8, we track a red car that is undergoing out-of-plane rotations and partial occlusions. This scenario is challenging to template based methods as the object change its appearance completely over time. The sequence is 200 frames long and the size of each image is 240×320 pixels. In this case we used the basic $11D$ feature vector (RGB color and the 8-bin histogram), in a single scale, and an ensemble of 3 classifiers. This was enough to obtain robust and stable tracking.

Next, we analyzed 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.

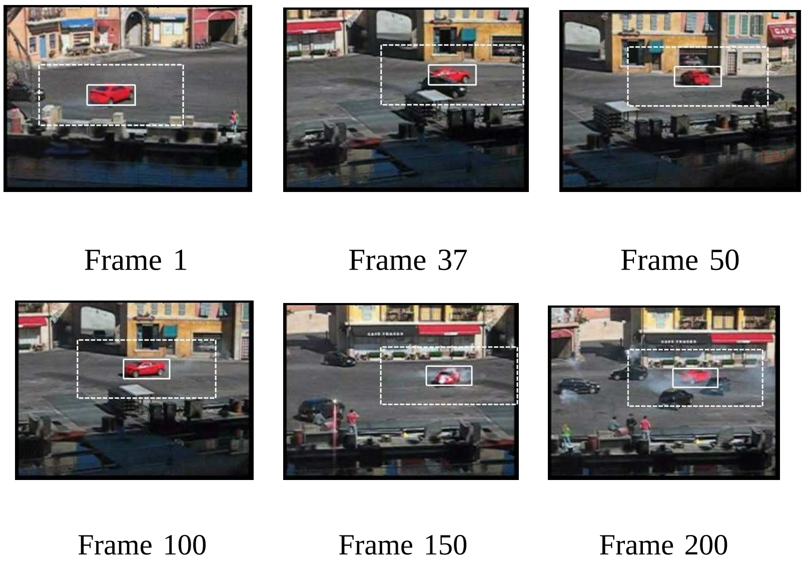


Fig. 8. Another tracking example. Ensemble tracking tracks a red car that is undergoing out-of-plane rotations and partial occlusions. The sequence was taken with a hand-held camera.

In the following experiment we analyzed the update regime taken by our method. Specifically, we were interested in several aspects of the method. How often are the weak classifiers updated? How does their weight change over time? and how does this method compare with a standard AdaBoost classifier that trains all its weak classifiers on a given frame. The results are shown in figure 9.

We found that the first weak classifier is kept for a very long period of time, providing an anchor for the tracker. The rest of the weak classifiers are updated more often, according to the particular sequence at hand. Note also that in some cases (frames 79,91 and 92) the algorithm dropped two weak classifiers in one frame and then added a new one. This is because both weak classifiers produced error rates that are close to chance and hence were removed. One new weak classifier was trained, and added to the ensemble, in their place. We also recorded the weight of the first weak classifier over time and, as can be seen, the weight decreases overtime, indicating that the associated weak classifier is slowly losing its ability to properly classify the data.

Our method constructs an on-line classifier and it is therefor reasonable to compare this classifier to a classifier that is built from scratch in each frame. Specifically, in each frame we trained an independent strong classifier using the same number of weak classifiers from scratch given the same training data that was used by ensemble tracking to train the new weak classifier. As can be seen, the standard AdaBoost classifier outperforms ensemble tracking by at most 8%. The large discrepency at the begining of the sequence stems from the fact that ensemble tracking trains one weak classifier per frame, whereas vanilla AdaBoost trains five weak classifiers per frame. Over time, this discrepancy diminishes. This experiment also shows the robustness of our approach to, at least, short occlusions of the object. The ensemble did not lock onto the pole because the new weak classifier trained at this frame had a high training error and thus the ensemble remained unchanged until the pedestrian crossed the pole entirely.

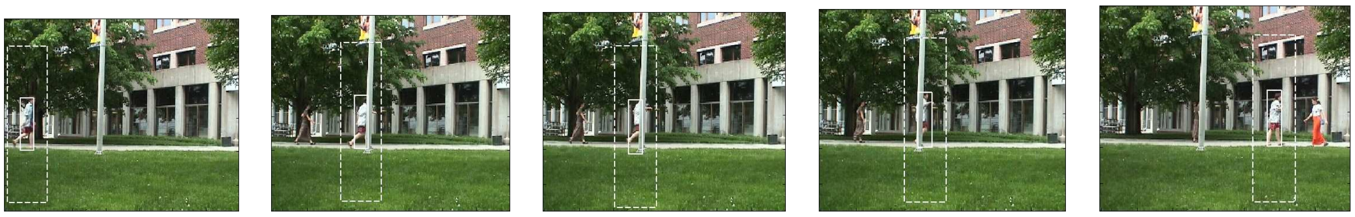
B. Results on gray-scale and IR sequences

An important advantage of ensemble tracking, over mean-shift methods, is its ability to work on images other than color images. Here we show two examples; one on gray-scale and another on IR. The only modification made to the algorithm was to change the feature space that is used to represent each pixel, as we describe next.

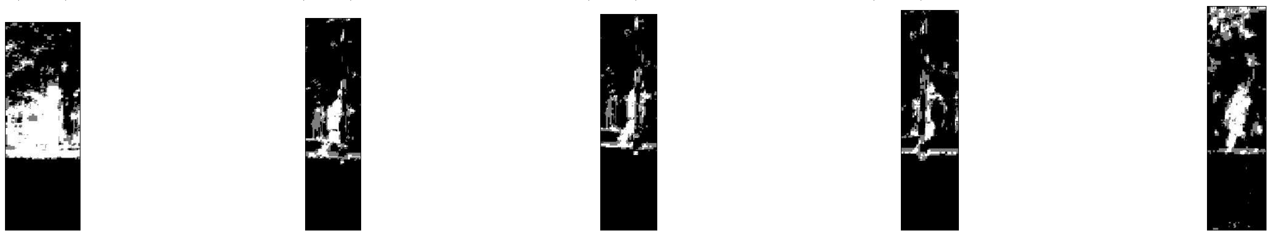
In one experiment we tracked a car over 225 frames of a gray scale, not color, video sequence [1]. While a $9D$ feature space (the 8-bin local histogram of oriented gradients and the gray scale intensity value) managed to track the car, we found that the non-linear feature space (which is $27D$ in this case) was much more robust to the position of the initial rectangle and in general performed much better. 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 to be a problem for our system. Some of the frames can be seen in figure 10.

In a similar experiment we used one of the PETS data sets to track a zodiac boat in a 671 frames long IR sequence. The results are shown in figure 11. Note that the system manages to discriminate the object from the background despite the small size of the object, that means a small sample set for training. Here as well as in the previous experiment, we used the non-linear feature space.

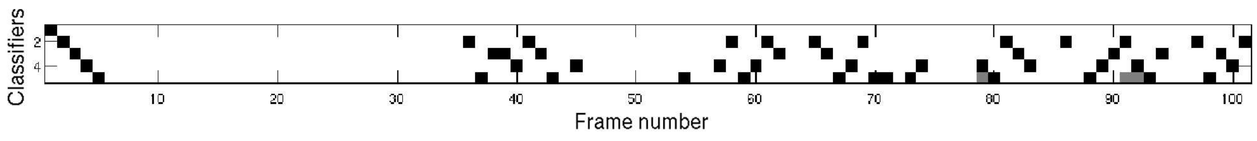
[1]Downloaded from the Karlsruhe university site at: http://i21www.ira.uka.de/image_sequences



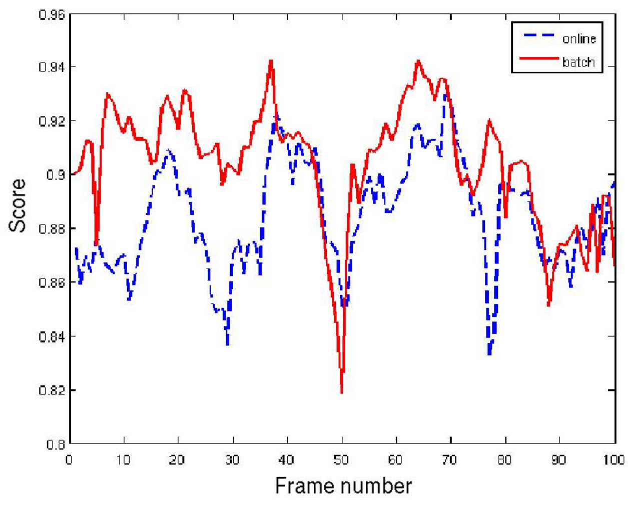
(a-1) (a-2) (a-3) (a-4)

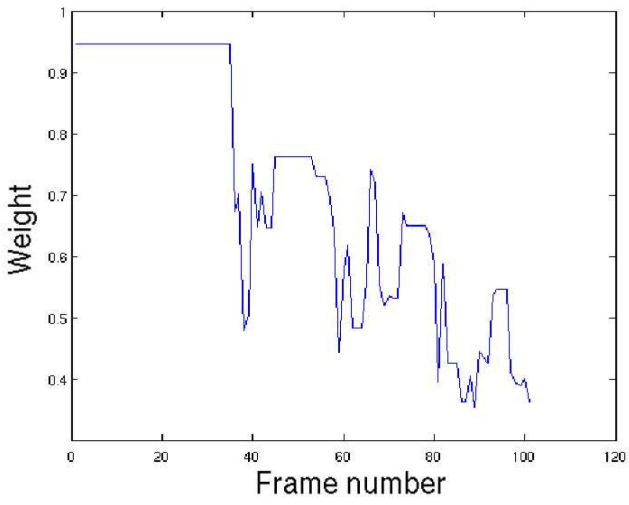


(b-1) (b-2) (b-3) (b-4)



(c)





(d) (e)

Fig. 9. Ensemble update rate. (a) some of the frames from this 100 frame sequence. (b) The corresponding confidence map. Note ho the ensemble correctly mark the pole as being part of the background (b-3). (c) The ensemble update rate. The x axis represent the fran number and each row correspond to a different classifier. Black means the classifier was updated, Gray means the classifier was remove Note that the first classifier is maintained through the entire sequence. Note that at frames 79,91 and 92 the ensemble removed two wea classifiers per frame and added only one in their place. (d) ensemble tracking (online) Vs. AdaBoost (batch). The x axis represent the fran number and the y axis represent the classification score using AdaBoost or ensemble tracking on each frame. See text for further detail (e) the weight of the first weak classifier, over the entire sequence. As can be seen the weight of the classifier decreases over time as th ability of the associated weak classifier to explain the data diminishes.

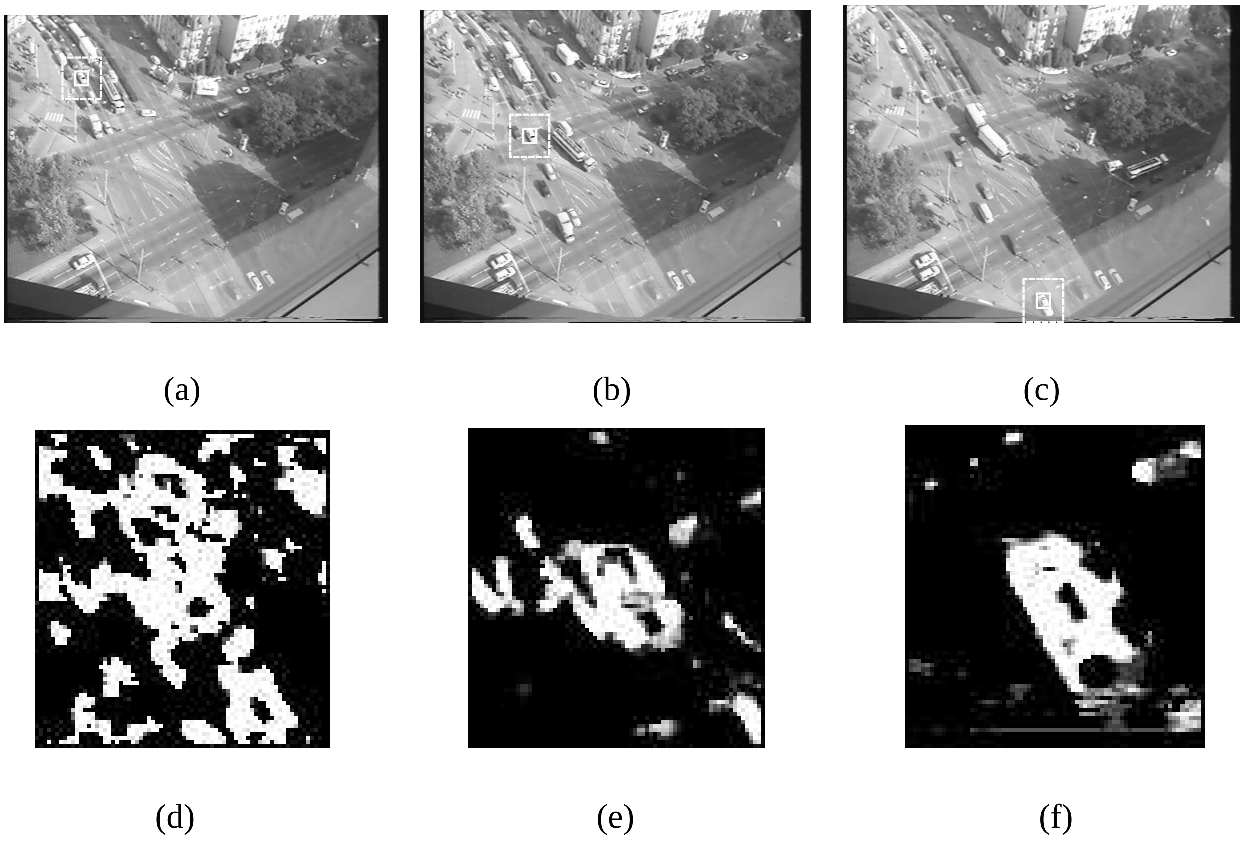


Fig. 10. Ensemble Tracking of a gray scale sequence. (a-e) Frames 0,100 and 225 from a 225-frame long sequence. We track a car from the upper left part of the image to the middle bottom of the image. (d-f) The confidence map for each frame. The confidence map corresponds to the dashed rectangle. Note how the confidence map picks the car's shape over time.

C. Handling occlusions

So far, we have only considered partial occlusions, or very brief occlusions that the tracker managed to overcome automatically. However, in some cases the object undergoes a long period of occlusion and the tracker can no longer handle it automatically. To handle these cases we use a very simple particle filter approach that works as follows. As long as the classification rate is high, the tracking goes unchanged. When the classification level drops, we stop updating the ensemble and switch to prediction mode. The ensemble update is resumed once we find a region that is classified with a classification rate higher than a predefined threshold. We take the classification rate to be the fraction of the number of pixels that were correctly classified, in each frame. In case all pixels are correctly classified, then the classification rate is 1. When all pixels are wrongly classified, then the classification value drops to 0. In practice we found the classification score to be above 0.9 and drop to about 0.5 in case of occlusion (This is because

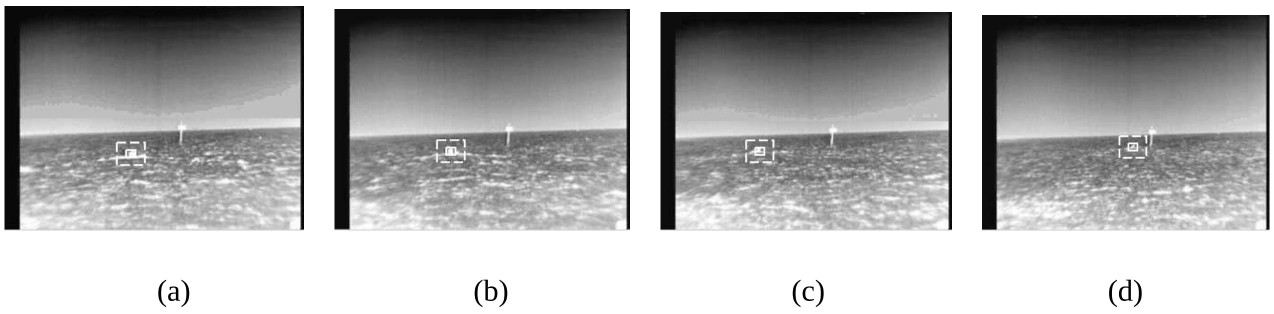


Fig. 11. Ensemble Tracking of an IR sequence. (a-d) Frames 1,200, 400 and 671 from a 671-frame long sequence. We track a zodiac boat in this IR sequence.

during occlusions we can still label the background pixels correctly). Once occlusion is detected we start sampling, according to the particle filter, possible locations that the object might appear. In each such location we compute the confidence map, using the existing ensemble, and run mean-shift to find the peak. We then compute the classification score at this point and if it is above a threshold (which in our case was set to 0.7) then tracking resumes.

The particle filtering assumes zero-motion with different uncertainty in the x and y directions. This is because, in our case, the objects mainly move horizontally. We do not assume constant velocity because both the object and the camera are assumed to be moving and hence a constant motion assumption, in the image plane, does not hold. Clearly one can use image stabilization to stabilize the entire image first and then use a filtering method with a constant velocity or constant acceleration for prediction. In the following examples we sample 5 locations in each frame. Increasing the number of locations sampled per frame will result in earlier re-detection at the cost of higher processing cost per frame.

Figure 12 show several frames from an 80-frame color video sequence where the size of each frame is 240×320. We use an $11D$ feature space, an ensemble of 3 classifiers and just a single level of the pyramid. The classification score proved to be a reliable enough to detect occlusions (observe how it drops sharply when the car is occluded). The car was re-detected two frames after it re-appears and tracking is resumed.

Another example is shown in figure 13. This is a 348-frame long color sequence, taken by a hand-held

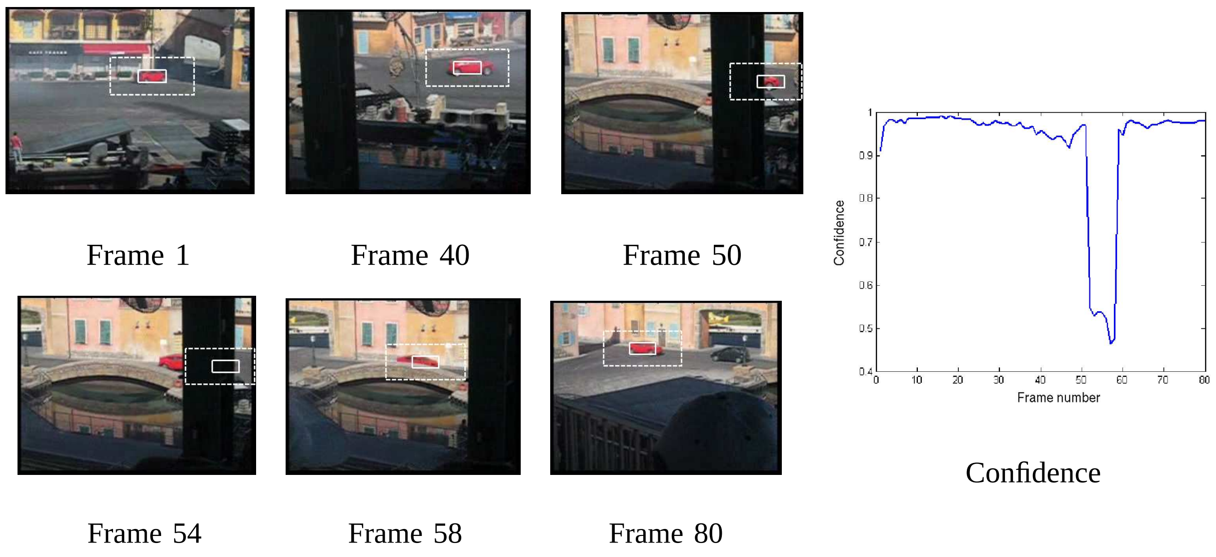


Fig. 12. Handling occlusions. On the left we show 6 frames from an 80-frame long sequence of a red car taken with a hand-held camera. On the right we show the confidence measure that is used during tracking. The x axis is for the frame number and the y axis is the classification score. The classification score is taken to be the number of pixels that are correctly classified by the ensemble, in each frame. As can be seen, most of the time the ensemble correctly classifies more than 95% of the pixels. This number drops to about 0.5 when occlusion occurs. The tracking re-initialization kicks in and picks up the car again and the tracking resumes.

camera, in which we track a woman going behind a large pole. The length of the occlusion is about 100 frames, and still the tracker managed to re-lock on the target. Observe how the method automatically handles partial occlusion (see how the confidence map of frame 57 correctly handles the partially occluding pole). In frame 155 it seems that the tracker is locked on the man, not the woman, however looking at the confidence map, as well as the confidence score which is about 0.6 shows that the tracker "knows" that he should still not re-initiate detection and should keep on looking (as before, our re-detection threshold is set to 0.7). Finally, in frame 167, the lady re-appears and the tracker picks her up with confidence level of about 0.95 and resumes tracking.

VI. 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

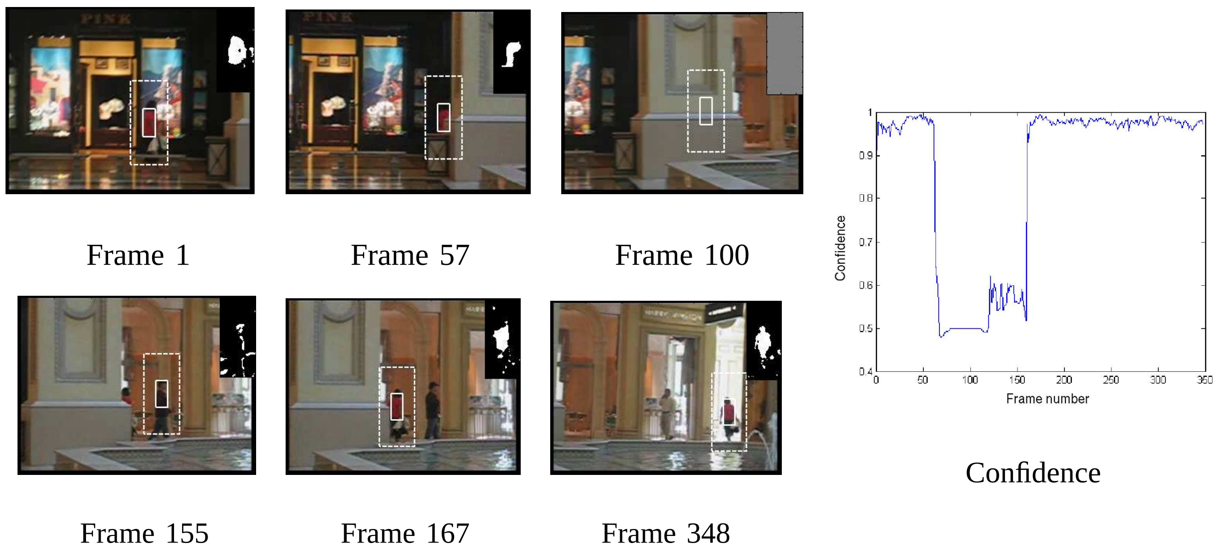


Fig. 13. Handling occlusions. On the left we show 6 frames from a 348-frame long sequence, taken with a hand-held camera, of a woman going behind a pole. In the top-right corner of each image we overlaid its associated confidence map (the region corresponds to the dashed rectangle, where white represent the object and black means the background). On the right we show the confidence measure that is used during tracking. The x axis is for the frame number and the y axis is the classification score. The classification score is taken to be the number of pixels that are correctly classified by the ensemble, in each frame. As can be seen, most of the time the ensemble correctly classifies more than 95% of the pixels. This number drops to about 0.5 when occlusion occurs. After the occlusion ends, and before the woman re-appears (frame 155), the tracker hovers around with a low confidence score of about 0.6 until the woman re-appears (frame 167) and the tracker snaps back to her.

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.

We have shown that the tracker can work in a wide variety of scenarios, including static and dynamic cameras, color, gray-scale and IR imagery and various object size. The tracker can also handle some occlusions, by refusing to learn pixels that belong to the occluding object. The classification score was shown to be a reliable confidence measure that can be used to detect occlusions and particle filtering can be used to overcome the occlusion.

There are several limitations to the proposed system. First, the tracker is not designed to handle full and long term occlusions. This is solved by adding a particle filtering on top of it, still it would enhance the method if particle filtering and ensemble tracking could be fused together. Second, the tension between adaptation and drifting appears here as well. Namely, it is possible to have the tracker adapt more rapidly to legitimate changes in the scene, at the expanse of suffering from drift. A somewhat ad-hoc solution is to prevent the tracker from removing the weak classifier trained on the first frame. Finally, the current feature space selected does not take into account spatial information, thus making the problem more difficult than it should be. Again, an ad-hoc solution might be to break the region to be tracked into several sub-regions and build a different ensemble of weak classifiers for each sub region independently.

Going beyond visual tracking we hope that this method can be used in any case where a discriminant function of time-varying distributions is needed.

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] Chu, F. and Zaniolo, C. Fast and Light Boosting for Adaptive Mining of Data Streams. The Eighth Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2004.

[5] Collins T. R., Liu, Y. and M. Leordeanu Online Selection of Discriminative Tracking Features. *IEEE Trans. on Pattern Analysis and Machine Intelligence* (PAMI), 27:10, pp 1631-1643, 2005.

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

[7] Crowley, J. and Berard, F. Multi-Modal Tracking of Faces for Video Communications. *Proceedings of the Conference on Computer Vision and Pattern Recognition* (CVPR ’97), Puerto Rico 1997.

[8] 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.

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

[10] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *IEEE Conference on Computer Vision and Pattern Recognition* (CVPR), 2005.

[11] W.T. Freeman and M. Roth. Orientation histograms for hand gesture recognition. International Workshop on Automatic Face- and Gesture- Recognition, IEEE Computer Society, Zurich, Switzerland, June, 1995, pp. 296–301.

[12] J. Friedman, T. Hasite and R. Tibshirani. Additive Logistic Regression: a Statistical View of Boosting. Dept. of Statistics, Stanford University Technical Report, 1998.

[13] Freund, Y. and 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.

[14] Georescu, B., Shimshoni, I. and 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.

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

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

[17] Jepson, A.D., Fleet, D.J. and El-Maraghi, T. Robust Online Appearance Models for Vision Tracking. In *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25(10):1296-1311.

[18] Kotler, J. and Maloof, M. Dynamic Weighted Majority: A new Ensemble Method for Tracking Concept Drift. In *Proceedings of the Third International IEEE Conference on Data Mining*, 2003.

[19] Levi, K. and 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.

[20] Lin R-S, Ross, D., Lim, J. and Yang, M-H. Adaptive Discriminative Generative Model and Its Applications. In Conference on Neural Information Processing System (NIPS), Vancouver, 2004.

[21] Nguyen, H. T. and Smeulders, A. Tracking Aspects of the Foreground against the Background. In the European Conference on Computer Vision (ECCV), Prague, 2004.

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

[23] Vapnik, V. N. Estimation of Dependences Based on Empirical Data. Springer-Verlag, 1982.