Subset Selection for Efficient SVM Tracking

Shai Avidan [*]

MobilEye Vision Technologies
10 Hartom street
Jerusalem, Israel

E-mail: avidan@mobileye.com

Abstract

We update the **SVM** *score of an object through a video sequence with a small and variable subset of support vectors. In the first frame we use all the support vectors to compute the* **SVM** *score of the object but in subsequent frames we use only a small and variable subset of support vectors to update the* **SVM** *score. In each frame we calculate the dot-products of the support vectors in the subset with the pattern of the object being tracked. The difference in the dot-products, between past and current frames, is used to update the* **SVM** *score. This is done at a fraction of the computational cost required to re-evaluate the* **SVM** *score from scratch in every frame. The two methods we develop are "Cyclic subset selection", in which we break the set of all support vectors into subsets of equal size and use them cyclically, and "Maximum variance subset selection", in which we choose the support vectors whose dot-product with the test pattern varied the most in previous frames. We combine these techniques together for the problem of maintaining the* **SVM** *score of objects through a video sequence. Results on real video sequences are shown.*

1 Introduction

Object detection in video sequence consists of object detection and confirmation. First, the object must be detected in the image and then it must be tracked, and confirmed, in subsequent frames. To confirm that we are still tracking it, we need to re-classify the object in each frame. This might be computationally expansive , especially if we maintain several hypothesis for each object, or if we have several objects in the scene (say, several vehicles on the road, or several faces in an office scene).

Object detection algorithms exhaustively search the current frame in various positions, scales and orientations for a desired object. At each position the candidate image patch is passed to a classifier that determines if the object appears in that particular image patch. These algorithms use various classifiers such as Support Vector Machine [10], neural networks [12, 16] or maximum likelihood on products of histograms [15]. These methods are too slow to run in real-time and one way to accelerate them is to use a rejection-based scheme. In this scheme candidates that are deemed unlikely to be the object are rejected early on so that the classifier can focus on the "interesting" regions of the image. Such approaches include maximal rejection of Elad *et al* [6], sequential **SVM** of Romdhani *et al* [11], the cascade of AdaBoost classifiers of Viola and Jones [18] or the FloatBoost method of Li *et al.* [8].

However, all these methods focus on the problem of detecting an object in a given frame rather than the problem of re-classifying the object in subsequent frames. Clearly, we can use the detected regions from previous frames to guide the object detection algorithm in the current frame. But bear in mind that since these are "interesting" regions in the image the rejection scheme will not help and the full force of the classifier will have to be used for every candidate. Our goal is to accelerate the confirmation stage, given that the object was correctly detected in a previous frame.

In this work we focus on Support Vector Machines because they were shown to perform well on face or vehicle detection [10, 11, 2]. Recall that **SVM** classifies a test pattern by summing the result of a non-linear function (called *kernel*) on the dot-product between the test pattern and a set of support vectors. Thus, evaluating a test pattern is linear in the number of support vectors. Face detection applications, for example, might have several hundreds of support vectors [11, 10]. If there are several objects in the image then this amounts to several thousands of dot-products, just for re-classification.

One approach to reducing the run-time complexity of **SVM** classification is the reduced set method [4, 13] that computes a small number of synthetic support vectors that approximate the original set of support vectors. This process takes the form of a non-linear optimization and is done

[*]The author is currently with the faculty of the Interdisciplinary Center, Herzelia, Israel

off-line. Still, maintaining good classification results requires several hundreds of support vectors in the reduced set. Romdhani *et al.* [11] applied **SVM** sequentially. After each dot-product of the test pattern with a support vector a decision is made if to proceed with the evaluation. In most cases a small number of support vectors is enough to reject a pattern. Unfortunately, this approach will not accelerate object re-classification since these regions in the image are the most probable places for the object to be and hence most, if not all, of the support vectors will be used.

Instead, we propose a different approach. Given that the object was detected in a previous frame, we look for a small and variable number of support vectors that will suffice for a correct update of the **SVM** score for the current frame. Our goal is to find this small subset of support vectors. This is complementary to filtering techniques such as Kalman Filtering or Condensation [7, 9] that are concerned with correct integration of information over time. We, on the other hand, are concerned with reducing the number of features (i.e., support vectors) we need to use. To this end we evaluate two different techniques.

The first technique, termed "Cyclic subset selection" (**CSS**) breaks the summation of the dot-products across several, say k, frames. At each frame we perform only part of the dot-products, keep their results and then use a tracker to track the object to the next frame, where we continue to perform the dot-products. At each frame we sum the intermediate scores of the last k frames, reducing the total number of dot-products performed per-frame by a factor of k. If the test pattern does not change much between successive frames, then this method is approximating a low pass filter over the **SVM** score, had all the support vectors been used in each frame.

The second technique, termed "Maximum Variance Subset Selection" (**MVSS**), chooses a subset of support vectors based on the variance of their dot-product with the test pattern over time (i.e. in previous frames). This way we choose the support vectors that are most likely to affect the **SVM** score in the current frame. The difference between the current dot-products and the previous dot-products is used to update the **SVM** score. Thus, we can update the **SVM** score with only a fraction of the support vectors being used. The subset can be selected deterministically to be the n support vectors with the largest variance in their dot-product, or it can be selected stochastically by a weighted sampling of the support vectors, where the weight is based on the variance.

We combine the above mentioned techniques and show results on the problem of updating **SVM** score of vehicles in a video sequence.

2 Support Vector Machine

For the paper to be self contained we give a brief description of **SVM**. The interested reader is referred to [17, 3] for a more detailed description.

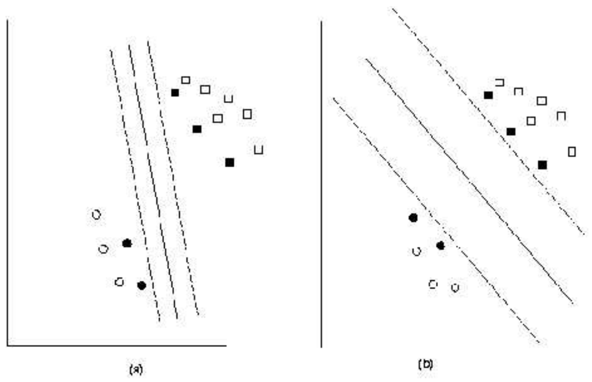


Figure 1. Support Vector Machine: (a) A separating hyperplane with small margin. (b) A separating hyperplane with a large margin. A better generalization capability is expected from (b). The filled squares and circles are termed "support vectors".

2.1 Support Vector Machine

Consider the simple case of two linearly separable classes. Given a data set $\{\mathbf{x_i}, y_i\}_{i=1}^{l}$ of l examples $\mathbf{x_i}$ with labels $y_i \in \{-1, +1\}$, we wish to find a separating hyperplane between the two classes. Formally, we consider the family of decision functions

$$f(\mathbf{x}) = sgn(\mathbf{w}^T\mathbf{x} + b) \tag{1}$$

and wish to find $\mathbf{w}, b$ such that $sgn(\mathbf{w}^T\mathbf{x_i} + b) = sgn(y_i)$. This problem is in general ill-posed because there might be an infinite number of separating hyperplanes. The question is which one has a low generalization error (i.e. which one will do a good job in classifying new examples). It was shown by Vapnik [17] that choosing the hyperplane with the minimal norm of $\mathbf{w}$ minimizes the "Structural Risk" which is an upper bound on the generalization error. Intuitively, this $\mathbf{w}$ is the one to maximize the margin between the two classes (See Figure 1). Practically, this amounts to solving the following quadratic optimization problem (QP)

$$\begin{array}{ll} \min_{\mathbf{w}} \frac{1}{2}\mathbf{w}^T\mathbf{w} & \\ & \\ subject \;\; to & y_i(\mathbf{w}^T\mathbf{w} + b) \geq 1, \\ & i = 1...l \end{array} \tag{2}$$

that can be solved quite efficiently. The example vectors closest to the separating hyperplane are called "support vectors". The classification itself is performed by measuring the signed distance of the test image from the separating hyperplane.

But how can the **SVM** be extended to handle decision functions that are not linear in the data? The answer is to use a nonlinear mapping Φ of the input data and map it to some high-dimensional (possibly even with infinite dimensions) *feature space* $\mathcal{F}$. The linear **SVM** is then performed in $\mathcal{F}$ and will therefor be nonlinear in the original input data. Formally, let

$$\Phi : \mathcal{R}^n \rightarrow \mathcal{F} \tag{3}$$

be a nonlinear mapping from input space to feature space and the decision functions we deal with becomes

$$f(\mathbf{x}) = sgn(\sum_{j=1}^{l} y_j \alpha_j \Phi(\mathbf{x})^T \Phi(\mathbf{x_j}) + b). \qquad (4)$$

where α_j is a set of parameters computed by solving the QP problem. However, working in feature space can be prohibitively expansive to compute. Therefor we use Mercer kernels on the input data to avoid computing the dot products in feature space. Mercer kernels $k(\mathbf{x}, \mathbf{x_j})$ satisfy that $k(\mathbf{x}, \mathbf{x_j}) = \Phi(\mathbf{x})^T \Phi(\mathbf{x_j})$. This way, instead of performing a non-linear mapping first and then do a dot-product in feature space, the order is reversed, first we perform the dot-product and then apply the non-linear mapping. Thus, in kernel-**SVM** we use the following decision functions

$$\begin{aligned} f(\mathbf{x}) &= sgn(\textstyle\sum_{j=1}^{l} y_j \alpha_j \Phi(\mathbf{x})^T \Phi(\mathbf{x_j}) + b) \\ &= sgn(\textstyle\sum_{j=1}^{l} y_j \alpha_j k(\mathbf{x}, \mathbf{x_j}) + b) \end{aligned} \qquad (5)$$

and the quadratic programming problem becomes:

$$\begin{aligned} & maximize \\ & \quad W(\alpha) = \textstyle\sum_{i=1}^{l} \alpha_j - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j k(\mathbf{x_i}, \mathbf{x_j}) \\ & subject \;\; to \\ & \qquad\qquad \alpha_i \geq 0, \quad i = 1...l, \\ & \qquad and \quad \textstyle\sum_{i=1}^{l} \alpha_i y_i = 0. \end{aligned} \qquad (6)$$

It turns out that α_j are equal to 1 for examples on the border between the two classes and 0 otherwise. In typical applications about 10% of the examples have α_j equal to 1 and these examples are called *support vectors*. The rest of the examples are not relevant because they do not help separate between the two classes. The only difference between kernel and linear **SVM** is that the dot product of linear **SVM** is replaced with a kernel function.

Typical kernels used in the **SVM** literature include $k(\mathbf{x}, \mathbf{x_j}) = exp(- \mathbf{x} - \mathbf{x_j}||)^2$ which leads to a Gaussian RBF, $k(\mathbf{x}, \mathbf{x_j}) = (\mathbf{x}^T \mathbf{x_j} + 1)^d$ which represent polynomial of degree d and $k(\mathbf{x}, \mathbf{x_j}) = tanh(\mathbf{x}^T \mathbf{x_j} - \Theta)$ which leads to multi-layer perceptron. Extension to non-separable classification problem exist [5], where the idea is that a penalty term is used to govern the price we are willing to pay for misclassified examples.

2.2 Reduced Set Methods

Reduced set methods aim at reducing the run-time complexity of **SVM**, during classification, by using a reduced set of support vectors. There are two methods for computing the reduced set. The first involves *selecting* the most important support vectors from the given set of support vectors. This implies changing the weights α_i of the remaining support vectors to compenstate for the support vectors we omit. The second method involves *constructing* a set of newly synthesized support vectors, and their weights, that will approximate the original set. A comprehansive discussion on the topic of reduced set methods can be found in [4, 14].

Both methods approximate the separating hyperplane given by the vector $\Psi \in \mathcal{F}$ with a vector $\Psi' \in \mathcal{F}$. But while Ψ is expressed using N_x images of input patterns, Ψ' is expressed using only N_z images of input patterns, and $N_z < N_x$. Let

$$\Psi = \Sigma_{i=1}^{N_x} \alpha_i \Psi(\mathbf{x_i})$$

where $\alpha_i \in R$ and $\mathbf{x_i} \in R^N$ are images of input patterns and let

$$\Psi' = \Sigma_{i=1}^{N_z} \beta_i \Psi(\mathbf{z_i})$$

with $N_z < N_x, \beta_i \in R, \mathbf{z_i} \in R^N$. Then this leads to the following minimization in $\beta_j, \mathbf{z_j}$:

$$\begin{aligned} ||\Psi - \Psi'||^2 &= \Sigma_{i,j=1}^{N_x} \alpha_i \alpha_j k(\mathbf{x_i}, \mathbf{x_j}) \\ &+ \Sigma_{i,j=1}^{N_z} \beta_i \beta_j k(\mathbf{z_i}, \mathbf{z_j}) \\ &- 2\Sigma_{i=1}^{N_x} \Sigma_{j=1}^{N_z} \alpha_i \beta_j k(\mathbf{x_i}, \mathbf{z_j}). \end{aligned}$$

This minimization is possible even though Ψ is not given explicitly, because we use it implicitly, through the kernel. In the *selection* reduced set method we take the $\mathbf{z_j}$ to be some subset of the original support vectors $\mathbf{x_j}$ and only estimate the β_j. In the *construction* reduced set method we generate a *synthetic* set of support vectors and then try to adjust the weights β_j to improve the apporimxation. A common approach in both cases is the greedy approach which increases the number of support vectors one by one, adjusting their weights after every support vector is added to the reduced set. Thus, the support vectors in the reduced set can be sorted in order of importance from the most important reduced-set vector to the least important reduced-set vector.

3 Subset Selection

Assume that an object is detected in the first frame of a video sequence and is to be re-evaluated in subsequent frames. Further assume that all the support vectors were used in the detection stage, but we wish to use only a small subset of them in subsequent frames for confirmation. Let $\mathbf{x_i}$ denote the test pattern to be evaluated in the $i - th$ frame in the video sequence. We will also assume that an SSD tracker was used to track the test pattern from one frame to the next. In ideal conditions where the object does not deform, the view point does not change, the illumination remains fixed and the sensor does not fluctuate we would expect the **SVM** score to be identical for every test pattern $\mathbf{x}_i \quad 1 \leq i \leq n$. Clearly this does not happen. However, we assume that running the full **SVM** classifier on every pattern $\mathbf{x}_i$ will correctly classify it.

Our goal is to approximate the **SVM** score obtained using all the support vectors with only a small subset of them. This marks a departure from known reduced set methods in two aspects. First, we perform an online reduced set selection that is tuned for the *particular* pattern that is currently tracked, whereas traditional reduced set methods try to find a reduced set that will apporximate the full **SVM** for *every* pattern. Second, finding a good subset must be done considerably faster then evaluating the full **SVM** for the proposed method to be usefull.

For clarity let us make the following definitions:

Definition: The *response* of a support vector $\mathbf{s_j}$ to a test pattern $\mathbf{x_i}$ is

$$\alpha_j y_j k(\mathbf{x_i}, \mathbf{s_j}).$$

Definition: The *response variance* of support vector $\mathbf{s_j}$ is

$$Var(\alpha_j y_j k(\mathbf{x_i}, \mathbf{s_j})) \quad i = 1...n$$

where n is the number of frames the test pattern $\mathbf{x_i}$ appears in.

3.1 Cyclic Subset Selection

A straightforward approach to choosing a subset of the support vectors is to break the set of all support vectors into k subsets of equal size and use a different subset in every frame. This way, every support vector will be used once every k frames.

To measure the error incurred by this method consider the following toy problem. The video consists of only two frames with patterns $(\mathbf{x}_1, \mathbf{x}_2)$ and there are only two support vectors $(\mathbf{s}_1, \mathbf{s}_2)$. We assume that:

$$\sum_{j=1}^{2} y_j \alpha_j k(\mathbf{x_i}, \mathbf{s_j}) > 0 \quad \forall i = 1, 2$$

and consider the difference between the full **SVM** evaluation and the Cyclic subset selection (**CSS**) version for the test pattern $\mathbf{x_2}$. The **SVM** score of $\mathbf{x_2}$, using full **SVM** evaluation is given by

$$(y_1 \alpha_1 k(\mathbf{x_2}, \mathbf{s_1}) + y_2 \alpha_2 k(\mathbf{x_2}, \mathbf{s_2}))$$

whereas its **SVM** score using **CSS** (with two subsets) is given by

$$(y_1 \alpha_1 k(\mathbf{x_1}, \mathbf{s_1}) + y_2 \alpha_2 k(\mathbf{x_2}, \mathbf{s_2}))$$

The difference between the two expressions is given in the following derivation.

$$\begin{array}{ll} (y_1 \alpha_1 k(\mathbf{x_2}, \mathbf{s_1}) + y_2 \alpha_2 k(\mathbf{x_2}, \mathbf{s_2})) - & \\ (y_1 \alpha_1 k(\mathbf{x_1}, \mathbf{s_1}) + y_2 \alpha_2 k(\mathbf{x_2}, \mathbf{s_2})) & = \\ y_1 \alpha_1 (k(\mathbf{x_2}, \mathbf{s_1}) - k(\mathbf{x_1}, \mathbf{s_1})) & = \\ y_1 \alpha_1 (\Phi(\mathbf{x_2})\Phi(\mathbf{s_1}) - \Phi(\mathbf{x_1})\Phi(\mathbf{s_1})) & = \\ y_1 \alpha_1 (\Phi(\mathbf{x_2} - \mathbf{x_1})\Phi(\mathbf{s_1})) & = \\ y_1 \alpha_1 k(\mathbf{x_2} - \mathbf{x_1}, \mathbf{s_1}) & \end{array} \tag{7}$$

and the dot-product $< \mathbf{x_2} - \mathbf{x_1}, \mathbf{s_1} >$ should be small since the SSD tracker minimized the difference between $\mathbf{x}_1$ and $\mathbf{x}_2$. This suggests that the error incurred by splitting dot-product computations across several frames is bounded by the kernel of the dot-product of the temporal derivative of the test pattern and the support vectors. This is why we use an SSD tracker, it minimizes the sum-of-squared-differences in gray values. And in doing so it minimizes the error introduced by **CSS**.

Cyclic subset selection can be summarized as follows. Given n support vectors, break them into k groups of equal size. At each frame compute the dot-product of the current pattern with the current n/k support vectors and add the result of the rest of the $n - n/k$ dot-products computed in the previous $k - 1$ frames to give the **SVM** score for the pattern in the current frame.

Ideally, **CSS** will approximate a moving average of the full **SVM** score. This is because it takes every support vector to be an approximation of its average response value over the past k frames. Unfortunately, this may not be good enough in practice. As we show in the experimental section, the rigid structure of **CSS** does not allow it to change the subset of support vectors selected and there might be cases in which a subset of support vectors that hardly change their response to the test pattern is chosen, while support vectors that have very large response variance are not chosen, leading to errors in the updated **SVM** score.

3.2 Maximum Variance Subset Selection

Instead of using fixed subsets that are cyclically used we look for support vectors that we suspect might change their response to the test pattern and use them.

Let $f(\mathbf{x_i})$ be the **SVM** score of pattern $\mathbf{x_i}$ (either using full SVM computation or **CSS**). Then the **SVM** score of pattern $\mathbf{x_{i+1}}$ can be written as:

$$f(\mathbf{x_{i+1}}) = f(\mathbf{x_i}) - \sum_{j=1}^{N_S} y_j \alpha_j k(\mathbf{x_i}, \mathbf{s_j}) + \sum_{j=1}^{N_S} y_j \alpha_j k(\mathbf{x_{i+1}}, \mathbf{s_j}). \tag{8}$$

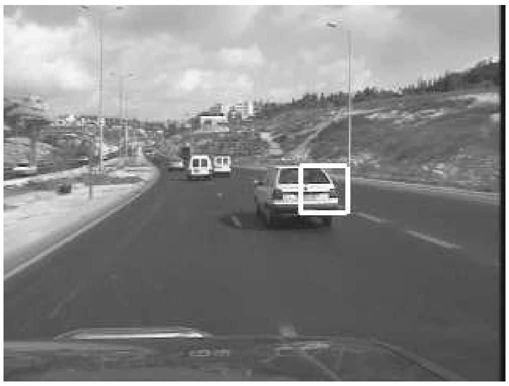
Where N_S is the number of support vectors in the subset of frame $i + 1$. Note that as N_S approach the total number of support vectors we obtain a better approximation of the correct **SVM** score. Intuitively, we would like to choose the support vectors whose response, with respect to this particular test pattern, varies the most. This is because there is no need to re-compute the dot-product with a support vector with low response variance. The difference between the old and new support vector responses is used to update the **SVM** score. We use deterministic and stochastic approaches to select support vectors with maximum variance.

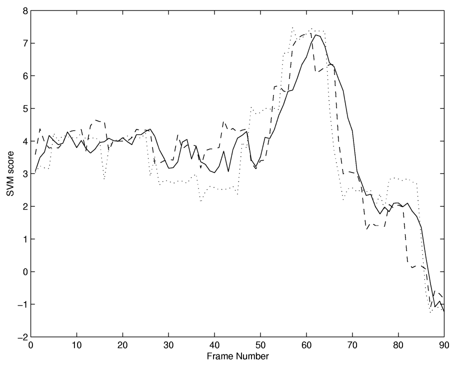
3.2.1 Top Maximum Variance Subset Selection

Top Maximum Variance Subset Selection (Top-**MVSS**) takes the n support vectors with the largest response variance. To do this we keep track of the variance of the response of each support vector to the test pattern in the past. In every frame We choose the n support vectors with the largest response variance, for some fixed number n. Alternatively, we can select all support vectors with response variance greater than a predefined threshold, which can allow us to bound the error on the updated **SVM** score.

(a) Frame 0

Frame 50


(c) Frame 100



Cyclic subset selection evaluation

Figure 2. Evaluation of the Cyclic subset selection (CSS) method. (a-c) show the first, middle and last frames from a sequence of 100 frames. (d) compares the SVM score of CSS to full SVM evaluation with 50 support vectors. The solid line is the SVM score using all 50 support vectors. The dashed line shows CSS with 5 subsets (i.e. 10 support vectors per frame). The dotted line shows CSS with 10 subsets (i.e. 5 support vectors per frame).

3.2.2 Stochastic Maximum Variance Subset Selection

Stochastic Maximum Variance Subset Selection (Stochastic-**MVSS**) performs a weighted sampling of the support vectors. we perform a weighted sampling of m support vectors, where the weight of every support vector is proportional to its response variance, and m is some predefined fixed number. In this process there is a higher probability that we will choose support vectors with high response variance, that will affect the updating of the **SVM** score, than support vectors with low response variance.

It is important to emphasize that subset selection is not equivalent to computing a reduced set of support vectors as was done, for example, in [11]. As we will show in the experimental section, support vectors with relatively low importance might have a larger variance with respect to the test pattern and hence might affect the updated **SVM** score more than the leading support vectors (that might have low response variance).

Each of the above mentioned techniques serves a different purpose. The **CSS** technique guarantees that all the support vectors will be used every k frames, for some fixed number k. However, it does not take the response variance into account and therefor might fluctuate with respect to the correct **SVM** score. To battle this phenomena we have introduced the Maximum Variance Subset Selection (**MVSS**) that specifically selects support vectors that have large influence on updating the **SVM** score. The Top-**MVSS** method is a deterministic algorithm and as such it might get stuck with a single subset. Stochastic-**MVSS** can prevent that from happening. Taken together the three methods cover a wide range of conditions and allow for an accurate update of the **SVM** score, using only a small number of support vectors.

A couple of comments are in order. First, there is no need to determine the number of selected support vectors beforehand. In fact it would be wiser to use the **SSD** tracking error to determine this number online. Second, the estimated variance is biased, because not all the support vectors are evaluated in every frame.

4 Experiments

In the experiments that follow we assume that the detection of the object in the first frame was completed successfully and proceed from there. Furthermore, we have used a tracker to track the object from one frame to the next, and used our tech-

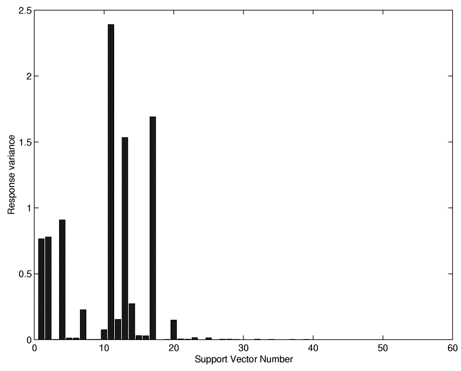


Figure 3. The response variance of support vectors to a test pattern tracked over 100 frames. There are 50 support vectors that are ordered from left to right, in a decreasing order of importance, yet the largest variance is recorded for support vector 12. The test pattern is the one shown in Figure 4. See text for further details.

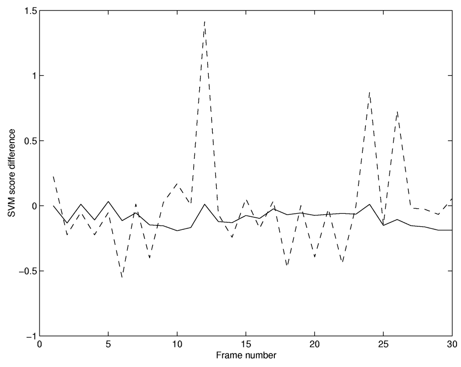


Figure 5. Cyclic subset selection (CSS) Vs. Maximum variance subset selection (MVSS). The graph shows the difference between the SVM score calculated by each of the methods and the correct SVM score, computed using all the support vectors. The dashed line is the difference between CSS (using 18 support vectors per-frame) and the correct SVM score (using all 50 support vectors), the solid line is the difference between MVSS (using 9 support vectors due to stochastic-MVSS and 9 support vectors due to Top-MVSS) and the correct SVM (using all 50 support vectors). The CSS technique does not take into account support vectors that exhibit large response variance and therefor fluctuates much more than the MVSS technique.

nique to update the **SVM** score computed in the first frame.

The classification engine was trained on a set of approximately 10000 images of vehicles and non-vehicles. Vehicles include cars, SUVs and trucks in different colors and sizes. The images were digitized from a progressive scan video at a resolution of 320×240 pixels and at 30 frames per second. Typical vehicle size is about 50×50 pixels. The vehicles and non-vehicles were manually selected and reduced to the size of 20×20 pixels. Their mean intensity value was shifted to the value 0.5 (in the range $[0..1]$) to help reduce the effect of variations in vehicle color. In all the experiments we used an homogeneous quadratic polynomial kernel given by $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}')^2$ to perform the learning phase. The classification rate was about 92% for the learning set, with about 2000 support vectors. A similar classification rate was obtained for the testing set that contained approximately 10000 images as well. To speed up the classification phase we used the Reduced Set Method [4] to reduce the number of support vectors from 2000 to 400. The Reduced Set Method shows that for homogeneous quadratic polynomial kernel the number of support vectors does not have to exceed the dimensionality of the input space. The number of support vectors can be reduced, through Principal Component Analysis on the support vectors in feature space, to a number bounded by the dimensionality of the input space, which is 400 in our case. In practice we found that the 50 support vectors with the largest eigenvalues are sufficient for classification.

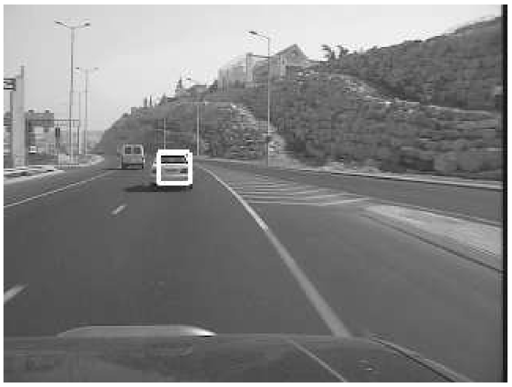
In all the experiments we used an optic-flow [1] based tracker to track the rectangle from frame to frame. In each frame we have used the proposed techniques in the following order. First, we used **CSS**, then we used stochastic-**MVSS** to sample support vectors with large variance and finally we used Top-**MVSS** to select additional support vectors. The support vectors selected by any one of the techniques were added to the subset and evaluated in the current frame. In addition, the variance response of each support vector, in the subset, is updated. The experiments were conducted on a mix of MATLAB and C++ programs and hence exact timing information is not available. We did observe that decreasing the size of the subset increased the speedup in run time.

We found that the method works better on reduced-set vectors than on the full set of support vectors. The reason is that if we have to select a subset from, say, 2000 support vectors, then we need to choose a large portion of the support vectors, or else the response variance of the support vectors outside the subset will be large enough to offset the results. Also, there is the chance that many support vectors will be correlated with each other and therefor choosing one of them will not suffice.

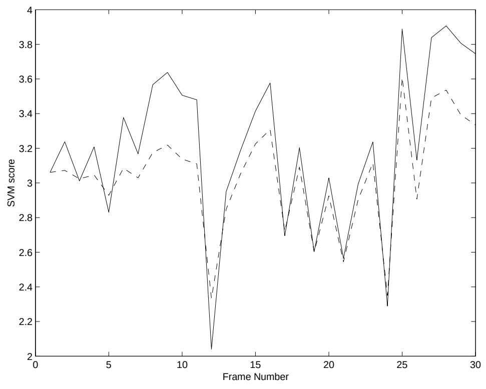
In the first experiment we compared **CSS** with full **SVM** evaluation. Figure 2 show several images from a 100-frame long sequence. We used the reduced homogenous quadratic polynomial with 50 support vectors and plotted the **SVM** score when using all the support vectors in every frame and compared that to **CSS** using 5 or 10 subsets.

In the second experiment we analyzed which support vectors had the largest variance in their response to a test pattern. We measured the variance of the response of each support vector to a test pattern that was tracked over 100 frames and show the result in Figure 3. The support vectors are ordered left-to-right with the leading support vectors on the left. Yet, the support vector with the highest response variance is support vector number 12. A possible reason for this behvior is that different support vectors respond to different variations in the test pattern (change in illumination, view-point, etc.) and therefor different support vectors will vary their response to reflect changes in the different image formation factors.

Next we tested the maximum variance subset selection



(a)



(b)

Figure 4. Maximum Variance Subset Selection(MVSS). (a) the first image from a 30 frame sequence. (b) Comparing MVSS with full SVM evaluation. The SVM score evaluated using all 50 support vectors is shown in solid line, the dashed line shows the SVM score computed using MVSS using 10 vectors (5 using stochastic-MVSS and 5 using Top-MVSS).

(**MVSS**). In the first frame we used all 50 support vectors for object detection. In subsequent frames we used stochastic-**MVSS** to sample 5 support vectors and another 5 support vectors using Top-**MVSS**, giving a total of 10 support vectors per-frame. Figure 4 shows the results of this experiment. The maximum difference between the **SVM** score computed using all support vectors and the one computed using maximum variance subset selection was 0.2 at most. Because of the stochastic-**MVSS** we repeated the experiment 100 times and reported the average **SVM** scores.

In the following experiment we compared **CSS** with **MVSS**. We compared both methods on a 30 frame sequence and show the results in Figure 5. The graph shows the difference between the **SVM** score, as computed by each of the techniques and the correct **SVM** score. Notice how the **CSS** fluctuates because it does not account for support vectors with large variations in their response to the test pattern. The **MVSS** method, on the other hand gives much better results. Again, we repeated the experiment 100 times and reported the average **SVM** scores.

In the following experiment we compare the **MVSS** method with a combined **CSS** + **MVSS** method. We compared the performance of **MVSS** with 18 support vectors (In each frame we sampled 9 support vectors using stochastic-**MVSS** and 9 support vectors using Top-**MVSS**) and a combined **CSS** + **MVSS** (In each frame we choose 6 support vectors using **CSS**, another 6 support vectors were sampled using stochastic-**MVSS** and the other 6 support vectors were selected using Top-**MVSS**). Figure 6 shows the results. It shows that the **MVSS** captures the high-frequency fluctuations in the **SVM** score but might have a bias. The bias is corrected with the **CSS** technique. The results shown are averaged over 100 trials to avoid rear samplings in the stochastic-**MVSS** part of the algorithm affect the result.

In the last experiment we have used an RBF kernel, instead of the homogenous quadratic polynomial kernel we have used so far. Our goal was to show that instead of taking a reduced set with a smaller number of support vectors, it is better to use a reduced-set with a larger number of support vectors and then use our technique. The classification results of the RBF kernel were about 96% on the test set. Then, we used the method of [4] to create two reduced sets from the original set of 2092 support vectors. One reduced set consisted of 200 support vectors with a classification score of roughly 95%. The second reduced set consisted of only 90 support vectors with classification score of roughly 93%. Then we compared **CSS**+**MVSS** ran on the large reduced set (of 200 support vectors) to the results of the smaller reduced set (of 90 support vectors). The results are shown in Figure 7. As can be seen the combination of **CSS** and **MVSS** gives a better approximation of the reduced set of 200 support vectors, compared to the reduced set of 90 support vectors. Again, we repeated the experiment 100 times and reported the average **SVM** scores.

5 Conclusions

We have used subset selection to maintain the **SVM** score of a test pattern through a video sequence. In each frame we select a small subset of the support vectors and compute their dot-product with the test pattern in the current frame. The difference in dot-products between previous and current frames is used to update the **SVM** score of the test pattern at a fraction of the computational cost required to evaluate the **SVM** score from scratch in every frame.

In particular we have used “Cyclic subset selection” and “Maximum variance subset selection”. The first breaks the set of support vectors into smaller subsets and cyclically goes over them. The second chooses support vectors whose response to the test pattern vary the most. The maximum variance subset selection can be done either deterministically (i.e. choose the k support vectors with maximum variance) or stochastically (i.e. use weighted sampling to choose support vectors with high variance). Taken together this methods can effectively reduce the computational cost associated with object re-evaluation.

While our work focused on support vector machines, it can

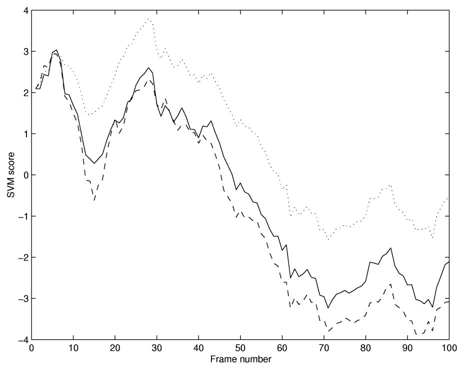


Figure 6. Cyclic subset selection (CSS) and Maximum variance subset selection (MVSS). The graph shows the SVM score of a 100 frame sequence (not shown here), using three different methods. The solid line was computed using all the 50 support vectors and it serves as the ground truth. The dotted line was computed using 18 support vectors (9 stochastic-MVSS, 9 deterministic-MVSS). The dashed line was also computed using 18 support vectors (6 CSS, 6 stochastic-MVSS and 6 Top-MVSS. The combined MVSS and CSS gives a better approximation to the ground truth.

be extended to other classification techniques. For example, in the work of Viola and Jones [18] there is no need to recompute all the features from scratch, only those that exhibit large variance in response to the test pattern. The same can be said about the calculation of products of histograms [15].

Future research directions will focus on sparse classifiers (**SVM** in particular), so that once an object is detected, only a small subset of the support vectors will respond to it, and only this small subset will be used for later re-evaluations.

6 Acknowledgment

I would like to thank the research memebers of MobilEye Vision Technologies for their good advice and for allowing me to use some of their data in my experiments. I would also like to thank the anonymous reviewrs for their useful comments.

References

[1] P. Anandan, J. Bergen, K. Hanna and R. Hingorani. Hierarchical Model-Based Motion Estimation. In M. Sezan and R. Lagendijk, editors, *Motion Analysis and Image Sequence Processing*, chapter 1, Kluwer Academic Press, 1993.

[2] S. Avidan. Support Vector Tracking. In *IEEE Conference on Computer Vision and Pattern Recognition*, Hawaii, 2001.

[3] C.J.C. Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 2(2):121-167, 1998.

[4] C.J.C. Burges. Simplified support vector decision rules. In L. Saitta, editor, *Proceedings, 13th Intl. Conf. on Machine Learning*, pages 71-77, San Mateo, Ca, 1996.

[5] C. Cortes and V. Vapnik. Support Vector Networks. *Machine Learning*, 20:273-297, 1995.

[6] M. Elad, Y. Hel-Or and R. Keshset, Pattern Detection Using a Maximal Rejection Classifier", Pattern Recognition Letters, 23(12) 1459-1471, October 2002.

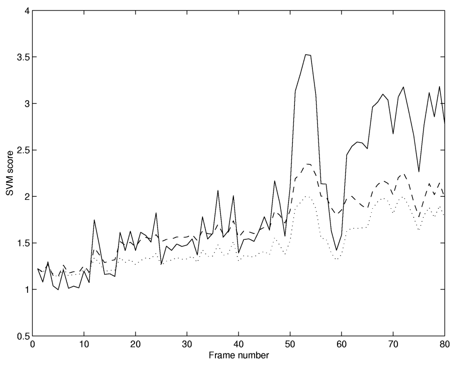


Figure 7. Cyclic subset selection (CSS)+Maximum variance subset selection (MVSS) Vs. Reduced Set. The solid line shows the SVM score for an 80-frame sequence, using 200 reduced set vectors. The dotted line shows the SVM score of 90 reduced set vectors and the dashed line shows our method (30 vectors using CSS, 30 using stochastic-MVSS and 30 using Top-MVSS). Our method performs better than the 90 reduced set vector with the same number of dot-products.

[7] M. Isard and A. Blake. Condensation - conditional density propagation for visual tracking. *Int. J. Computer Vision*, 28, 1, 5-28.

[8] S.Z. Li, L. Zhu, Z.Q. Zhang, A. Blake, H.J. Zhang and H. Shum. Statistical Learning of Multi-View Face Detection. In *Proceedings of the 7th European Conference on Computer Vision*, Copenhagen, Denmark, May 2002.

[9] K. Mikolajczyk, R. Choudhury and C. Schmid. Face detection in video sequence - a temporal approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, Hawaii, 2001.

[10] E. Osuna, R. Freund and F. Girosi. Training Support Vector Machines: An Application to Face Detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, Puerto Rico, pages 130-136, 1997.

[11] S. Romdhani, P. Torr, B. Scholkopf, and A. Blake. In *International Conference on Computer Vision*, Vancouver, 2001.

[12] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. PAMI, 20:23-38, 1998.

[13] B. Scholkopf, S. Mika, C. Burges, P. Knirsch, K.-R. Muller, G. Ratsch, and A. Smola. Input space vs. feature space in kernel-based methods. IEEE *Transactions on Neural Networks*, 10(5):1000-1017,1999.

[14] B. Scholkopf and A. Smola. Learning with Kernels. MIT Press, 1999.

[15] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In CVPR, pages 746-751, 2000.

[16] K-K. Sung and T. Poggio. Example-based learning for view-based human face detection. In *Proceedings from Image Understanding Workshop*, Monterey, CA, November 1994.

[17] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, N.Y., 1995.

[18] P. Viola and M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. In *IEEE Conference on Computer Vision and Pattern Recognition*, Hawaii, 2001.