Face Detection Using a 3D Model on Face Keypoints

Adrian Barbu, Gary Gramajo

Abstract—The Support Vector Machine is a powerful learning technique that is currently lacking an efficient feature selection method that scales well to the size of the computer vision data. In this paper we bring two contributions. First, we apply a recent feature selection algorithm to optimize a differentiable version of the SVM loss with sparsity constraints. The iterative algorithm alternates parameter updates with tightening the sparsity constraints by gradually removing variables based on the coefficient magnitudes and a schedule. We use nonlinear univariate response functions to obtain a nonlinear decision boundary with feature selection and show how to mine hard negatives with feature selection. Second, we propose an approach to face detection using a 3D model on a number of detected face keypoints. The 3D model can be viewed as a simplex that fully connects the keypoints, making optimization difficult. We also propose an optimization method by that generates a set of 3D pose candidates directly by regression and verifies them with the model’s energy. Experiments on detecting the face keypoints and on face detection using the proposed 3D model show that the feature selection and nonlinear response functions dramatically improve performance and obtain state of the art face detection results on three standard datasets.

1 INTRODUCTION

Most computer vision problems require learning classifiers from large amounts of data, with billions of observations and thousands of features. In these cases variable selection is important for speed and to obtain compact models that generalize well.

The Support Vector Machine [31] is a powerful machine learning technique with many applications in computer vision, medical imaging and beyond. There have been quite a few feature selection methods for SVM, some of the more recent being [4], [15], [24], [25], [26], [36] and will be discussed in Section 2.6. However, we are not aware of any computer vision application of these works, possibly due to the fact that they might be too computationally expensive for large datasets.

In this paper we propose a novel variable selection algorithm for SVM that directly minimizes the differentiable approximation [9] of the SVM primal objective function subject to sparsity constraints. The algorithm starts with the unconstrained loss function and alternates model update steps with variable selection steps that tighten the constraints according to an annealing schedule. After a large number of iterations, the model parameters converge to the minimum value of the loss function on the selected variables. At each iteration, the optimization algorithm performs one gradient update step of the model parameters and a selection step that removes variables based on the coefficient magnitudes. Because many variables are completely removed at each step according to a predefined schedule, each subsequent step is much faster than the previous step, resulting in a very fast training algorithm.

Contributions. This paper brings the following contributions to computer vision and
machine learning:
– It presents an efficient feature selection algorithm for SVM that alternates parameter updates with variable elimination based on coefficient magnitudes and an elimination schedule. The algorithm can obtain simultaneous feature selection and a nonlinear decision boundary through nonlinear response functions on the selected variables. The hard negative mining can be adapted to work with feature selection, which is crucial for computer vision problems with thousands of positives and billions of negatives. When learning without feature selection, the negative mining procedure is proved to obtain the same parameter values as if learning from the entire training set.
– It proposes a face detection approach that uses a 3D model on the configuration of certain face keypoints such as eyes, mouth corners, ears, nose etc. A number of 3D pose candidates are generated using image based regression and are evaluated based on the detected keypoint scores and distances from the model.

Face keypoint detection experiments show that feature selection combined with the nonlinear response functions bring considerable improvements in detection performance for the SVM classifier. Face detection experiments show that the proposed 3D model based approach obtains state of the art face detection results on three standard datasets.

2 Feature Selection for Regression and SVM

Let $D = \{(x_i, y_i) \in \mathbb{R}^M \times Y, i = 1, \ldots, N\}$ be a set of training examples. The space of outcomes $Y$ could be $Y = \{-1, 1\}$ for binary classification and $Y = \mathbb{R}^d$ for multivariate regression.

2.1 Feature Selection with Annealing Overview

Let $L_D(\beta)$ be a differentiable loss function defined using the training examples $D = \{(x_i, y_i) \in \mathbb{R}^M \times Y, i = 1, \ldots, N\}$. The Feature Selection with Annealing (FSA) algorithm is aimed at the constrained optimization of the loss function

$$\beta = \arg\min_{\{i, \beta_i \neq 0\} \leq k} L_D(\beta)$$

where the number $k$ of relevant features is a given parameter that could be obtained either by cross-validation or using an AIC/BIC criterion.

The FSA algorithm starts with an initial value of the parameters $\beta$, usually $\beta = 0$, and alternates two basic steps:
- one step of parameter updates by gradient descent towards minimizing the loss
  $$\beta \leftarrow \beta - \eta \frac{\partial L_D(\beta)}{\partial \beta}$$
- one variable selection step that removes some variables according to their magnitudes $|\beta_j|$. Usually many variables are removed at each iteration, keeping only a number $M_e$ of variables that have the largest values of $|\beta_j|$. The number $M_e$ of variables that are kept after each iteration $e = 1, N_{\text{iter}}$ is similar to an annealing schedule.

Through this schedule, the sparsity constraint after iteration $e$ is $|\{i, \beta_i \neq 0\}| \leq M_e$, thus the constraint is gradually tightened and after a large number of iterations the constraint $|\{i, \beta_i \neq 0\}| \leq k$ is reached.

Fig. 1. The number of features $M_e$ vs iteration $e$ for three annealing schedules, where $M = 10,000, k = 10$.

The annealing schedule $M_e, e = 1, N_{\text{iter}}$ used in this work is an inverse schedule with a parameter $v$ or $\mu = \frac{M}{k}$. 
\[ M_e = \max(k, \frac{M}{1 + e^\mu} - \frac{1}{N^{\text{iter}}}) \]  

Because the \( M_e \) quickly decreases after the first iteration, the FSA algorithm is very fast. In Figure 1 are shown the value of \( M_e \) vs the iteration number \( \epsilon \) for three schedules \( v = 50, 100 \) and 200, where \( M = 10,000 \) and \( k = 10 \). The FSA algorithm is summarized in Algorithm 1.

**Algorithm 1 Feature Selection with Annealing (FSA)**

| Input: | Training examples \( \{(x_i, y_i)\}_{i=1}^N \) with \( x_i \in \mathbb{R}^M \). |
|---------------------|-------------------------------------------------|
| Output: | Trained classifier parameters \( \beta \). |
| 1: Initialize \( \beta = 0 \). |
| 2: for \( e=1 \) to \( N^{\text{iter}} \) do |
| 3: Update \( \beta \leftarrow \beta - \eta \frac{\partial L_D(\beta)}{\partial \beta} \). |
| 4: Keep the \( M_e \) variables with highest \( |\beta_j| \) and renumber them 1, ..., \( M_e \). |
| 5: Set \( M = M_e \). |
| 6: end for |

The FSA algorithm can be used for the optimization of any differentiable loss function subject with a sparsity constraint of the form described in eq. (1). In this paper the FSA algorithm is used for optimizing loss functions of the form

\[ L_D(\beta) = \sum_{i=1}^N L(f_\beta(x_i), y_i) + \sum_{j=1}^M \rho(\beta_j) \]  

where \( f_\beta(x_i) \) is the predicted value of \( y_i \), for example \( f_\beta(x_i) = \beta^T x_i \), and \( L(u, y) \) is the loss function, such as \( L(u, y) = \|u - y\|^2 \) for regression. The prior \( \rho(\beta_j) \) on the parameters \( \beta_j \) could be for example the shrinkage \( \rho(\beta_j) = s\|\beta_j\|^2 \).

The specific forms of the loss functions for regression and SVM will be given in sections 2.3 and 2.4 respectively.

### 2.2 Nonlinearity and Feature Selection

A nonlinear prediction compatible with feature selection can be obtained as a sum of univariate functions on the selected variables

\[ f_\beta(x) = \sum_{j=1}^M f_{\beta_j}(x_j), \]  

where \( \beta_j \) is a parameter vector characterizing the response function on variable \( j \), with \( \beta_j = 0 \) indicating that feature \( j \) was not selected.

We will use piecewise linear univariate response functions \( f_{\beta_j}(x_j) \) that can be written as

\[ f_{\beta_j}(x_j) = u_j^T(x_j)\beta_j \]  

where \( u_j(x_j) \) is the basis response vector and \( \beta_j \in \mathbb{R}^{2+1} \) is the coefficient vector of variable \( j \).

We obtain

\[ f_\beta(x) = \sum_{j=1}^M u_j^T(x_j)\beta_j. \]  

Aside from the shrinkage prior \( \rho(\beta_j) = s\|\beta_j\|^2 \) we will use the second order prior

\[ \rho(\beta_j) = s\|\beta_j\|^2 + c \sum_{i=2}^2 (\beta_{j,i+1} + \beta_{j,i-1} - 2\beta_{ji})^2 \]  

that favors “smooth” feature response functions \( f_{\beta_j}(x_j) \) as shown in Figure 2. Since the coefficients \( \beta_j \) for variable \( j \) form a vector now, the variable selection criterion in the FSA changes from \( |\beta_j| \) to the group criterion \( \|\beta_j\| \) based on the entire parameter vector \( \beta_j \) related to the variable \( j \).

### 2.3 Feature Selection with Annealing for Regression

Given training examples \( D = \{(x_i, y_i) \in \mathbb{R}^M \times \mathbb{R}^d, i = 1, N\} \), the FSA for multivariate regression optimizes the square loss

\[ L_D(\beta) = \sum_{i=1}^N \|y_i - f_\beta(x_i)\|^2 + \sum_{j=1}^M \rho(\beta_j) \]  

\[ = \sum_{i=1}^N \sum_{k=1}^d (y_{ik} - f_{\beta_k}(x_i))^2 + \sum_{j=1}^M \rho(\beta_j) \]  

where \( \beta_k \) is a parameter vector characterizing the response function on variable \( k \), with \( \beta_k = 0 \) indicating that feature \( k \) was not selected.

We will use piecewise linear univariate response functions \( f_{\beta_k}(x_k) \) that can be written as

\[ f_{\beta_k}(x_k) = u_k^T(x_k)\beta_k \]  

where \( u_k(x_k) \) is the basis response vector and \( \beta_k \in \mathbb{R}^{2+1} \) is the coefficient vector of variable \( k \).

We obtain

\[ f_\beta(x) = \sum_{k=1}^d u_k^T(x_k)\beta_k. \]  

Aside from the shrinkage prior \( \rho(\beta_k) = s\|\beta_k\|^2 \) we will use the second order prior

\[ \rho(\beta_k) = s\|\beta_k\|^2 + c \sum_{i=2}^2 (\beta_{k,i+1} + \beta_{k,i-1} - 2\beta_{ki})^2 \]  

that favors “smooth” feature response functions \( f_{\beta_k}(x_k) \) as shown in Figure 2. Since the coefficients \( \beta_k \) for variable \( k \) form a vector now, the variable selection criterion in the FSA changes from \( |\beta_k| \) to the group criterion \( \|\beta_k\| \) based on the entire parameter vector \( \beta_k \) related to the variable \( k \).
where $y_i = (y_i^1, ..., y_i^d) \in \mathbb{R}^d$ and $f_\beta(x_i) = (f_\beta^1(x_i), ..., f_\beta^d(x_i)) \in \mathbb{R}^d$.

### 2.4 Feature Selection with Annealing for SVM

Given training examples $D = \{(x_i, y_i) \in \mathbb{R}^M \times \{-1, 1\}, i = 1, N\}$, the FSA for SVM optimizes the differentiable approximation of the primal SVM objective function from $\cite{9}$

$$L_D(\beta) = \sum_{i=1}^N L_h(y_i f_\beta(x_i)) + \sum_{j=1}^M \rho(\beta_j) \quad (9)$$

where $L_h : \mathbb{R} \rightarrow \mathbb{R}$ is the Huber-style differentiable approximation of the hinge loss $\cite{9}$:

$$L_h(x) = \begin{cases} 
0 & \text{if } x > 1 + h \\
\frac{(1 + h - x)^2}{4h} & \text{if } |1 - x| \leq h \\
1 - x & \text{if } x < 1 - h 
\end{cases} \quad (10)$$

### 2.5 Mining Hard Negatives

Usually in computer vision the training set $D = \{(x_i, y_i) \in \mathbb{R}^M \times \{-1, 1\}, i = 1, N\}$ contains billions of instances, which cannot be stored in the computer memory for training. However, because the Huberized SVM loss $\cite{10}$ is zero when $y f_\beta(x) > 1 + h$, most of the training examples will not contribute to the loss function at its minimum, and could in principle be ignored.

This idea has been used in $\cite{14}$ for mining “hard examples”, which were defined for the hinge loss as

$$H(\beta, D) = \{(x, y) \in D, y f_\beta(x) < 1\}$$

For the hinge loss without feature selection (i.e. no constraints on the number of nonzero coefficients), the authors give a procedure in $\cite{14}$ that finds all the hard examples in a finite number of steps and prove that the optimum obtained from minimizing the loss on the hard examples is the same as the optimum obtained from the entire training set $D$. The same conclusions can easily be carried over to the Huberized SVM loss $\cite{10}$ where the hard examples are

$$H_h(\beta, D) = \{(x, y) \in D, y f_\beta(x) < 1 + h\}.$$ 

But after imposing the sparsity constraints for feature selection, the Huberized SVM loss is no longer strictly convex and no such guarantees can be obtained. In this case we adopt a standard approach that starts with a number $N$ of negatives obtained randomly, and adds to the training set $N$ false positives at each iteration until convergence or a maximum number of iterations. Thus at each iteration the set of negatives increases with harder and harder negatives. The procedure is described in detail in Algorithm $\cite{2}$ and was found to work very well in practice as it will be seen in experiments.

#### Algorithm 2 Mining Hard Negatives

**Input:** Training examples $D$.

**Output:** Trained classifier parameters $\beta$.

1: Initialize set of negatives $C_1 \subset D$ with $|C_1| = N$
2: for $t=1$ to $T$ do
3: Train FSA-SVM with negative set $C_t$ obtaining parameters $\beta = \beta^{(t)}$.
4: Generate false positive set $F_t \subset D$ with $|F_t| \leq N$ using classifier $f_\beta(x)$.
5: Set $C_{t+1} = C_t \cup F_t$.
6: if $C_{t+1} = C_t$ then
7: Stop
8: end if
9: end for

Observe that when optimizing the loss $\cite{10}$ without sparsity constraints for feature selection, Algorithm $\cite{2}$ is equivalent to the mining procedure from $\cite{14}$ without step 3 (shrinking the cache). The proofs of Theorems 1 and 2 from $\cite{14}$ can easily be carried over to this case, which means that the algorithm will terminate in a finite number of steps and the minimum over the whole training set $D$ is the same as the minimum over $C_T$ when $T$ is large enough. Thus without feature selection, Algorithm $\cite{2}$ will find the optimum from the entire set $D$ after a finite number of steps. However, these guarantees cannot be offered under the feature selection sparsity constraints.

### 2.6 Related Work

There current literature addresses the feature selection problem for SVM in different ways.
The Recursive Feature Elimination [15] (RFE) procedure is a wrapper method that alternates the training of an SVM classifier on the current feature set with the removal of a percentage of the features based on the magnitude of the coefficients. In contrast, our approach removes variables long before the parameters $\beta$ have converged, thus it can faster than the RFE approach that fully trains the coefficients at each iteration.

Another type of feature selection methods for SVM impose sparsity constraints on the SVM weights, being limited to linear SVM [4], [25], [26], [36] or polynomial kernels [32]. The $L_1$-norm SVM [36] optimizes the hinge loss with a $L_1$ penalty, shows that the regularization path is piecewise linear and offers an algorithm to compute it. The combined SVM method [26] optimizes the hinge loss with a combination of $L_0$ and $L_2$ penalties using the DC (difference of convex functions) optimization. Our work optimizes a differentiable approximation of the hinge loss with $L_2$ regularization and $L_0$ constraints but uses the FSA optimization method [1], which is very fast and scales well to large datasets.

The KP-SVM [24] optimizes the Lagrange dual of the SVM problem with a kernel and a sparsity inducing penalty term on the variable weights. The KP-SVM obtains very good results on standard UCI datasets, but it is not clear how well it scales to the large data sizes from computer vision.

2.7 Application: Face Keypoint Detection

The FSA-SVM approach is used to detect nine face keypoints: the eye centers, mouth corners, ears, nose sides and chin. The face keypoints are detected using a sliding window classifier trained with the FSA-SVM and negative mining. Instead of just using a predefined set of features (such as HOG features), we can start with a larger feature pool and select a small set of good features to improve detection quality while keeping a small classifier that is fast and generalizes well.

**Image Pyramid.** The face keypoints are detected on a Gaussian pyramid with 4 scales per octave (i.e. resized by powers of $2^{1/4}$) down to a minimum size of $24 \times 24$ pixels. Thus the face keypoints are represented as $(x, y, s)$, where $(x, y)$ is the pixel location and $s$ is the index in the pyramid of the image containing the point.

**Feature pool.** The feature pool for training the classifiers consists of $288 \times 3 = 864$ Histograms of Oriented Gradients [11](HOG) features and 61000 Haar features. All features are extracted from the RGB channels in a $24 \times 24$ pixel window centered at the point of interest $(x, y)$ in the appropriate image of the Gaussian pyramid.

Details about training the keypoint detectors are given in section 4.1.

3 FACE DETECTION USING A 3D MODEL

The nine face keypoints detected by the FSA-SVM classifiers are used for face detection using a rigid 3D model on the face keypoint configurations.

3.1 Energy Model

Given an image, the goal is to find the faces and their 3D pose. The face 3D pose is represented as a projected rigid transformation $T_\theta : \mathbb{R}^3 \to \mathbb{R}^2$ with parameters $\theta = (u, s, R)$ consisting of 2D translation $u \in \mathbb{R}^2$, scale $s$ and 3D rotation matrix $R$, and defined as

$$T_\theta(x) = u + s\pi(Rx),$$

where $\pi : \mathbb{R}^3 \to \mathbb{R}^2, \pi(x, y, z) = (x, y)$ is the projection on the $(x, y)$ plane.

The face has $L$ keypoints that form a rigid 3D configuration that can be written as a $3 \times L$ matrix $F = (F_1, ..., F_L), F_i \in \mathbb{R}^3$. The 2D configuration of the face keypoints in an image is obtained as $\{T_\theta(F_i) + \epsilon_i, i = 1, L\}$ where $T_\theta$ is the 3D face pose defined above and $\epsilon_i \in \mathbb{R}^2$ are independent deformations for each keypoint, illustrated in Fig. 5.

For any $\theta = (u, s, R)$ let $B_\theta$ be the bounding box of the set of transformed keypoints
The face keypoints are fully connected by a simplex in our 3D model.

\{T_\theta(F_i), i = 1, L\}. Thus detecting a face means finding its 3D pose. The best configuration of faces is obtained by energy minimization in a Bayesian framework:

\[(\theta_1, ..., \theta_n) = \underset{n, \theta_1, ..., \theta_n}{\arg \min} E(n, \theta_1, ..., \theta_n)\]

\[E(n, \theta_1, ..., \theta_n) = E_{data}(\theta_1, ..., \theta_n) + E_{prior}(n, \theta_1, ..., \theta_n)\]

The data term depends directly on the detections of the \(L\) face keypoints in the image. The face keypoints are detected on a Gaussian pyramid as described in Section 2.7. The detected keypoints are rescaled to the original image scale, obtaining for each keypoint \(i \in \{1, ..., L\}\) a set \(K_i\) of detections.

The data term is based on the face scores \(S(\theta_j)\) in the image

\[E_{data}(\theta_1, ..., \theta_n) = \sum_{j=1}^{n} (\tau - S(\theta_j))\]

and the parameter \(\tau\) that controls the minimum score for a valid detection. The score function \(S(\theta)\) is defined in more detail in Section 3.5.

The prior \(E_{prior}(n, \theta_1, ..., \theta_n)\) enforces the constraints that the bounding boxes \(B_{\theta_j}, j = 1, n\) have small overlap with each other.

3.2 Related Work

Many works [14], [37] use computationally tractable tree-based models to represent the interactions between the locations of the object parts. In this work we explore a model where the unknown face part locations are fully connected with each other into a simplex parameterized by the projected similarity parameters \(\theta = (u, s, R)\), as shown in Figure 3.

Even though the proposed inference algorithm is not globally optimal, the model more than compensates this disadvantage, as shown in experiments.

In [16] a 2D part configuration is detected using version of the deformable part model [14] and then a 3D pose and shape is inferred from the 2D configuration. In contrast, our work directly uses the 3D pose to represent the relative positions of the parts. This is possible because the individual parts can be detected very reliably by the FSA-SVM approach.

There have been 3D approaches to object detection [23], [27], [29] that use different types of features that are extracted at certain positions depending on the object pose. However, none of these works is aimed at face detection. Moreover, our approach is based on the 3D configuration of a small number of face keypoints (parts) trained in a supervised way and we use a different inference algorithm.

Pose candidates have been previously proposed by image based regression in the shape regression machine [35] and for face alignment [7]. However, they are not based on a 3D model, are not geared for face detection and don’t predict the candidates from the keypoint locations.

3.3 Inference Algorithm

The inference algorithm is illustrated in Fig. 4 and contains:

1) A bottom-up candidate generation step that produces a number of pose candidates \(\theta_1, ..., \theta_n\).

2) A top-down scoring step that computes the scores \(S(\theta_j), j = 1, n\) from eq. (11) and removes low scoring candidates.

3) A non-maximal suppression step that outputs a set of high score candidates that satisfy the overlap constraints.

These steps are described in the next three sections.

3.4 Generating 3D Pose Candidates

Since the keypoints are detected for faces in a range of scales, the pose candidates are also
obtained for faces in the same range.

The 3D pose candidates are generated by image-based regression from the detected keypoint locations. The 3D pose \( \theta = (u, s, R) \) has six parameters \((u, s, \varphi) = (w^x, w^y, s, \varphi^x, \varphi^y, \varphi^z)\). The pose is predicted from a point \((p^x, p^y)\) by predicting the relative vector \(y = (w^x - p^x, w^y - p^y, s, \varphi^x, \varphi^y, \varphi^z)\). A specific 6D pose regressor is trained for each keypoint, using the same Haar+HOG feature pool as the keypoint detectors. The regressors are trained using FSA for the square loss \(8\) and piecewise linear weak learners \(6\) in each dimension. For the 3D pose regression the ground truth vectors \(y_i = (u^x - p_i^x, u^y - p_i^y, s, \varphi^x, \varphi^y, \varphi^z)\) are obtained from the 3D pose \((u, s, \varphi) = (u^x, u^y, s, \varphi^x, \varphi^y, \varphi^z)\) of the face containing the keypoint and the position \((p_i^x, p_i^y)\) of the keypoint in the image.

Other regression methods can be employed to generate the 3D pose candidates, such as regression forests \(10\) or boosted regression \(35\).

The ground truth 3D pose of each face is found as described next.

### 3.4.1 Fitting a Rigid Projection Transformation

Given a matrix \(3 \times L\) matrix \(F\) and a set of 2D points \(P = (p_1, \ldots, p_L)\) in the form of a \(2 \times L\) matrix, the goal is to find a rigid transformation \(\theta = (u, s, R)\) to minimize:

\[
E(u, s, R) = \|u1 + s\pi(RF) - P\|^2
\]

where \(\pi((x, y, z)^T) = (x, y)^T\) and 1 is the row vector of appropriate dimension with all entries 1.

The algorithm uses hidden variables for the \(z\) coordinates of the points \(p_i\) and iterates fitting the rigid transformation with updating the \(z\)-values.

### Algorithm 3 Fit Rigid Projection

**Input:** \(3 \times L\) matrix \(F\) and \(2 \times L\) matrix \(P\).

**Output:** Scalar \(s\), \(3 \times 3\) rotation matrix \(R\) and 2D vector \(u\) to minimize \(\|u1 + s\pi(RF) - P\|^2\)

1. Initialize \(L \times 3\) matrix \(B = (PT, 0)\).
2. for \(i = 1\) to \(N_{iter}\) do
3. Call Algorithm 4 to find \(u, s, R\) to minimize \(\|1^T u^T + sF^T R - B\|^2\)
4. Extract third column \(c_3 = (C_{13})_i\) of \(C = sF^T R\)
5. Update \(B = (PT, c_3)\)
6. end for
7. Change \(R\) to \(R^T\) and discard the \(z\)-component of \(u\).

The algorithm to fit a rigid transformation between two sets of points of the same dimension \(d\) is due to Schonemann \(28\) and is presented next.

### 3.4.2 Fitting a 3D Transformation

This algorithm is due to Schonemann \(28\).

Given two sets of points \(A, B\) of the same dimension \(d\), it finds a rigid transformation \((u, s, R)\) represented by a translation vector \(u\), scaling \(s\), and rotation matrix \(R\) and to minimize \(\|1^T u^T + sAR - B\|^2\).
Algorithm 4 Fit Rigid Transformation

**Input:** Matrices $A, B$ of size $p \times d$.

**Output:** Scalar $s \times d$ rotation matrix $R$ and $d \times 1$ vector $u$ to minimize $\|1^T u^T + s A R - B \|^2$

1. Compute the column means $\tilde{\alpha} = 1 A / p, \tilde{\beta} = 1 B / p$ and column centered matrices $A^* = A - 1^T \tilde{\alpha}$ and $B^* = B - 1^T \tilde{\beta}$.
2. Decompose $A^T B^* = U D V^T$ by SVD, where $U, V$ are rotation matrices and $D$ is a diagonal matrix.
3. Obtain $R = U V^T, u = \tilde{\beta} - s \tilde{\alpha} R$ and $s = tr[R^T A^T B^*] / tr(A^T A^*)$.

3.5 The 3D Face Pose Score $S(\theta)$

For a face with 3D pose $\theta = (u, s, R)$ let $(\varphi^x, \varphi^y, \varphi^z)$ be the roll-pitch-yaw decomposition of the rotation matrix $R$. For each keypoint type $i = 1, ..., L$ let $d_i$ be the distance between the transformed face keypoint $T_B(F_i)$ and the closest detected keypoint of type $i$ and let $p_i$ be the SVM score of the closest detected keypoint of type $i$.

The score $S(\theta) = S(u, s, R)$ of the 3D pose $\theta = (u, s, R)$ is based on the distances $d = (d_1, ..., d_L)$ and scores $p = (p_1, ..., p_L)$:

$$S(\theta) = S(\theta, d, p) = \sum_{i=1}^{L} (u_i(\varphi^y)d_i + u_i(\varphi^y)p_i)$$

$$= u(\varphi^y)^T d + u(\varphi^y)^T p.$$  \hspace{1cm} (11)

The coefficients $u(\varphi^y), u(\varphi^y)$ depend parametrically on the yaw angle $\varphi^y$ of the rotation $R$. The yaw angle ranges between $-\pi$ and $\pi$, being 0 for frontal faces and $\pm \pi/2$ for profile faces. For this application, it is discretized into 32 bins, so there are parameter vectors $\omega_k, u_k, k = 1, 32$, one pair for each yaw angle bin. These parameters are collected in the matrices $W = (\omega_1, ..., \omega_{32})$ and $U = (u_1, ..., u_{32})$.

Training the score function $S(\theta)$. The scoring function is trained to predict the overlap $o(\theta)$ between the bounding box $B_\theta$ of the transformed keypoints of the pose $\theta$ and the bounding box of an annotated face with largest overlap with $B_\theta$. Given a training set of poses $\theta_{j}, j = 1, N$ with yaw angle bins $b_j \in \{1, ..., 32\}$, distance and probability vectors $(d_j, p_j)$, and overlaps $o(\theta_j)$ with the annotated faces, learning the parameters $W, U$ is obtained by minimizing the following energy:

$$E(W, U) = \sum_{j=1}^{32} (S(\theta_j, d_j, p_j) - o(\theta_j))^2 + \sum_{k=1}^{32} (\rho(w_k) + \rho(u_k))^2$$

$$= \sum_{j=1}^{32} (w_j^T d_j + u_j^T p_j - o(\theta_j))^2 + \sum_{k=1}^{32} (\rho(w_k) + \rho(u_k))^2$$  \hspace{1cm} (12)

where the smoothness prior $\rho(w)$ is given in eq. [7]. This energy is quadratic in $(W, U)$ so it can be minimized analytically.

3.6 Non-Maximal Suppression

The non-maximal suppression repeats the following steps until convergence:

1. Select the pose candidate with the largest support above a threshold and finds the bounding box $B$ of its projected points.
2. Remove the candidates that have at least 50% overlap with $B$.

4 Experiments

First we present experiments evaluating the impact of feature selection and nonlinear response functions for the face keypoint detectors. Then we present experiments evaluating the face detector and compare it with some popular face detection methods.

The following datasets will be used in this work:

**LFPW.** The training part of the dataset contains 1132 image links with one face annotated per image. The face annotation consists of 35 keypoints, with a binary visibility label for...
Fig. 5. Precision-recall curves on the LPFW test set for detecting the left mouth corner (left), right eye center (mid-left), right ear canal (mid-right) and chin (right).

TABLE 1

| Method      | Number of SV | Average Precision |
|-------------|--------------|-------------------|
|             | Mouth | Eye | Ear | Chin | Mouth | Eye | Ear | Chin |
| LinSVM HOG  | 15k   | 11k | 118k | 25k   | 0.35  | 0.61 | 0.01 | 0.22 |
| SVM-PL HOG  | 8120  | 7637 | 4539 | 7515  | 0.55  | 0.76 | 0.23 | 0.51 |
| LinFSA-SVM  | 10k   | 7499 | 5197 | 9879  | 0.46  | 0.72 | 0.02 | 0.26 |
| FSA-SVM     | 7307  | 4167 | 3184 | 6522  | 0.79  | 0.91 | 0.56 | 0.66 |

each keypoint. Out of the 1132 training image links, only 762 links were still valid at the time of download, of which 215 contained two or more faces per image with only one face annotated, so they were discarded. Thus the LFPW training set used in this work contains the remaining 551 images with one face per image. The LFPW dataset also contains a test set of 300 links, out of which only 216 were valid. Out of these, 198 pictures included one face per image and were used as the LFPW test set in this paper.

**LFW.** The LFW dataset [17], [12] contains 13232 images, with one face annotated with nine keypoints in each image. From them, 468 images were removed from the evaluation since they contained two or more faces and only one was annotated. So the LFW dataset used in this paper contained 12764 images with one face per image.

**Helen.** The Helen dataset [22] contains 2330 high resolution images with one or more faces, of which only one is annotated. One hundred images have been removed because they contained unannotated faces, so the Helen dataset used in this paper contains 2130 images with one face per image.

**AFLW.** The AFLW dataset [19] contains 21123 images containing 24386 faces, annotated with 21 points. Of them, 16207 images were found to contain one face per image and were split into two disjoint sets, one of 999 images for training and 15208 images for testing.

### 4.1 Face Keypoint Detection Evaluation

Nine face keypoints (eyes, ears, mouth, nose) are detected, as shown in Figure 4 left. They are represented as 2D points \((x, y)\) in an image \(I_s\) of the Gaussian pyramid. The keypoint detectors are trained on the LFPW [3] train set described in the previous section.

**Training examples.** The training examples are points on the Gaussian pyramid, with the positives within one pixel from the keypoint annotation on the images of the pyramid where the inter-eye distance is in the \([20, 40]\) pixel range. The negatives are all points at least 4 pixels from the keypoint annotation. In total the LFPW training set contains about 0.5 billion negatives, all of which were used for training the classifiers. A separate classifier was trained for each keypoint.

**Algorithms.** The following algorithms were compared to evaluate the contribution of the nonlinear response functions and feature selection to the performance of the obtained classifier.
Fig. 6. Precision-recall curves for face detection on three datasets. Top-left: LPFW test set (198 images). Top-right: LFW dataset (12764 images). Bottom-left: Helen dataset (2130 images). Bottom-right: AFLW dataset (15208 images).

1) LinSVM HOG - The Linear SVM classifier trained on the HOG features.
2) SVM-PL HOG - SVM with piecewise linear response functions trained on the HOG features.
3) LinFSA-SVM - The FSA-SVM on the loss with linear response $f_\beta(x) = \beta^T x$, trained on the HOG + Haar features to select $k = 1500$ features.
4) FSA-SVM - The FSA-SVM algorithm on the piecewise loss trained on the HOG + Haar features to select $k = 1500$ features.

**Detection criteria.** The following criteria were used for evaluating keypoint detection. The visible face keypoint is considered detected in an image if a detection is found at most 3 pixels away in one of the images of the pyramid. A detected point $p$ in one of the images of the pyramid is a false positive if it is at least 5 pixels away from the true keypoint location (visible or not).

**Results.** In Figure 6 are shown the precision-recall curves obtained on the LPFW test set based on the detection criteria above. The Average Precision and number of support vectors are shown in Table 1.

One could see that the performance of the linear SVM on HOG features is poor and that the feature selection and nonlinear response functions improve performance considerably.

### 4.2 Face Detection

The LinSVM-HOG, SVM-PL HOG, and FSA-SVM keypoint detectors from previous section are also evaluated as part of the 3D model based face detection approach described in Section 3. Also evaluated are the fully independent face detector from Zhu & Ra-
The cascade of convex models [8] and the Microsoft face detector[1] used in [30] and three face detectors available in OpenCV 2.4.6: OpenCV Tree, OpenCV Alt, and OpenCV LBP.

The precision-recall curves on the LFPW, LFW and Helen datasets are shown in Figure 6. The criterion for a correct detection is a 50% overlap (intersection over union) between the GT bounding box and the detection box. The results are consistent with the experiments from section 4.1, in that both the feature selection and the piecewise linear learners help considerably. As tested with the pretrained independent classifiers, the performance of Zhu [37] on the LFW dataset is poor, possibly because the faces are too small (on average 90×100 pixels).

The FSA-SVM obtains state of the art performance, outperforming Zhu on all four datasets. It also outperforms the Microsoft face detector on three of the datasets, including the most challenging dataset AFLW, which contains many profile and semi-profile faces. The FSA-SVM and the Microsoft detector performs similarly on the LFW dataset, containing mostly frontal faces. Examples of detections using the FSA-SVM and Zhu are shown in Figures 7 and 8.

5 CONCLUSION AND FUTURE WORK

This paper presented an approach to face detection using a rigid 3D model on a number of face keypoints that are detected with a sliding window classifier. The classifier is trained by an SVM approach that contains a feature selection step to speed-up detection and control overfitting.

Experiments showed that the proposed feature selection algorithm works well in practice, and the feature selection and nonlinear decision boundary help in improving the keypoint detectors. The experiments also showed that the 3D-model based face detection using the trained face keypoint detectors obtains state of the art results on three standard datasets.

Our conclusion is that in some applications such as face detection, it is not that important to use a computationally tractable model such as the DPM [14] and that a model based on a fully connected simplex together with a good generator of simplex candidates can work as well and sometimes even better.

In the future we plan to find a better energy for the 3D model and extend the FSA-SVM method to challenging object detection problems such as those part of the PASCAL Visual Object Challenge [13].

REFERENCES

[1] Adrian Barbu and Yiyuan She. Fast simultaneous feature selection and learning. arXiv preprint arXiv:1310.2880, 2013.
[2] A. Bartoli, D. Pizarro, and M. Loog. Stratified generalized procrustes analysis. IJCV, 101(2):227–253, 2013.
[3] P.N. Belhumeur, D.W. Jacobs, D.J. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. In CVPR, pages 545–552, 2011.
[4] P.S. Bradley and O.L. Mangasarian. Feature selection via concave minimization and support vector machines. In ICML, pages 82–90, 1998.
[5] P. Breheny and J. Huang. Coordinate descent algorithms for nonconvex penalized regression, with applications to biological feature selection. Ann. App. Statistics, 5(1):232–253, 2011.
[6] R.L. Burden and J.D. Faires. Numerical analysis. 2001. Brooks/Cole, USA.
[7] Xudong Cao, Yichen Wei, Fang Wen, and Jian Sun. Face alignment by explicit shape regression. In CVPR, pages 2887–2894, 2012.
[8] H. Cevikalp and B. Triggs. Efficient object detection using cascades of nearest convex model classifiers. In CVPR, pages 3138–3145, 2012.
[9] O. Chapelle. Training a support vector machine in the primal. Neural Comp., 19(3):1155–1178, 2007.
[10] Antonio Criminisi, Jamie Shotton, Duncan Robertson, and Ender Konukoglu. Regression forests for efficient anatomy detection and localization in ct studies. In Medical Computer Vision. Recognition Techniques and Applications in Medical Imaging, pages 106–117. Springer, 2011.
[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In CVPR, volume 1, pages 886–893, 2005.
[12] M. Dantone, J. Gall, G. Fanelli, and L. Van Gool. Real-time facial feature detection using conditional regression forests. In CVPR, pages 2578–2585, 2012.
[13] M. Everingham, L. Van Gool, C.K.I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge. IJCV, 88(2):303–338, 2010.
Fig. 7. Detected faces on the LFPW dataset by FSA-SVM (left) and Zhu (right). Also shown (left) are the detected keypoints that were closest to the 3D pose of the detected face.

Fig. 8. Detected faces on the LFW dataset by FSA-SVM (left) and Zhu (right).

[14] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2010.

[15] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Mach. Learn.*, 46(1-3):389–422, 2002.

[16] M. Hejrati and D. Ramanan. Analyzing 3d objects in cluttered images. In *NIPS*, pages 602–610, 2012.

[17] G.B. Huang, M. Mattar, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. 2007.

[18] D. Jiang and J. Huang. Majorization minimization by coordinate descent for concave penalized generalized linear models. Technical Report, 2011.

[19] Martin Koestinger, Paul Wohlhart, Peter M. Roth, and Horst Bischof. Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization. In *First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011.

[20] K. Koh, S.J. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *JMLR*, 8(8):1519–1555, 2007.

[21] J. Langford, L. Li, and T. Zhang. Sparse online
6 Simulation Experiments

In this section we present simulations on synthetic data to evaluate the feature selection and prediction performance of the FSA-SVM in comparison with other popular learning methods.

The data for simulations has correlated predictors sampled from a multivariate normal \( x \sim \mathcal{N}(0, \Sigma) \) where \( \Sigma_{ij} = \delta^{i-j} \) and \( \delta = 0.9 \). The label \( y \) for a data point \( x \in \mathbb{R}^M \) is

\[
y = \begin{cases} 
1 & \text{if } \sum_{i=1}^{k^*} x_{10i} > 0 \\
0 & \text{otherwise}
\end{cases}
\]

Thus the data is linearly separable and only the variables with index \( 10i, i = 1, k \) are relevant. All experiments were performed on a six core Intel Core I7-980 machine at 3.3GHz with 24Gb RAM.

6.1 Stability with Respect to Parameters

In this experiment, we evaluate the stability of the FSA-SVM Algorithm 1 with respect to its tuning parameters: the learning rate \( \eta \), the annealing rate \( \mu \) and the number of iterations \( N_{\mathrm{iter}} \). We experimented on linearly separable data with \( N = 1000 \) observations of dimension \( M = 1000 \) and \( k = k^* = 10 \).

In Figure 9 is shown the dependence of the average area under the ROC curve (AUC) with respect to \( \eta \) (left), \( \mu \) (middle) and \( N_{\mathrm{iter}} \) (right). For the left plot, we had \( \mu = 1, N_{\mathrm{iter}} = 1000 \), for the middle plot \( \eta = 20 \mu, N_{\mathrm{iter}} = 1000 \) and for the right plot \( \mu = 1, \eta = 20 \). The obtained curves are averages from 10 independent runs. One can see that all three parameters have a large range of values that yield very good and stable prediction performance. One can also see that if the learning rate \( \eta \) or the annealing parameter \( \mu \) are too large, the performance drops quickly.

6.2 Feature Selection Comparison

In this experiment we compare the variable selection and the prediction performance of the FSA-SVM algorithm with different regularized loss methods and with the Logitboost algorithm. Logitboost can be used for feature selection if each weak learner depends on only one variable, then one variable will be selected at each boosting iteration.
Fig. 9. Dependence of the AUC vs algorithm parameters for a linear dataset with \( M = N = 1000, k = k^* = 10 \). Left: dependence on \( \eta \). Middle: dependence on \( \mu \) when \( \eta = 20\mu \). Right: dependence on \( N_{\text{iter}} \).

The experiments are performed on the linearly separable data described above. The algorithms being compared are:

1) FSA - The FSA Algorithm [1] for the SVM loss \( v = 200 \) annealing schedule, \( \eta = 20 \).

2) FSA-LB - The FSA Algorithm [1] for the logistic loss with the \( v = 200 \) annealing schedule, \( \eta = 20 \).

3) \( L_1 \) - The interior point method [20] for \( L_1 \)-penalized Logistic Regression using the online implementation from [http://www.stanford.edu/~boyd/l1logreg/](http://www.stanford.edu/~boyd/l1logreg/).

Since we assume that the number of variables \( k \) is given, we will determine the value of the \( L_1 \) penalty \( \lambda \) using the bisection method [6]. The bisection procedure calls the interior point training routine about 9 times until a \( \lambda \) was found that gives exactly \( k = 10 \) nonzero coefficients.

4) MCP - Logistic regression using MCP (Minimax Concave Penalty)[34]. Two implementations were evaluated: the \texttt{ncvreg} R package based on the coordinate descent algorithm [5] and the \texttt{cvplogistic} R package based on the Majorization-Minimization by Coordinate Descent (MMCD) algorithm [18]. The \texttt{cvplogistic} package obtained better results, which are reported in this paper.

5) SCD - The two R packages \texttt{ncvreg} and \texttt{cvplogistic} also implement the SCAD penalty, with the \texttt{cvplogistic} package obtaining better results, reported in this paper.

6) LB - Logitboost using univariate linear regressors as weak learners. In this version, all \( M \) linear regressors (one for each variable) are trained at each boosting iteration and the best one is added to the classifier.

The variable detection rate measures how many times (in percent) the set of selected variables is exactly the same as the set of true variables i.e. \( \{i, \beta_i \neq 0\} = \{i, \beta_i^* \neq 0\} \).

One important remaining issue is the model selection problem of finding how many variables should be included in the model (or how large should be the penalty for the penalized methods). To address this, different methods such as AIC/BIC or cross-validation could be used with the FSA algorithm and would lead to different results. To remove the influence of this model selection step, we assume that the number of variables \( k \) is given, and we take \( k = k^* \) in these simulations.

In Table 2 are shown the variable detection rate and area under the ROC curve (AUC) on unseen data of same size as the training data, for the methods being evaluated. The averages are obtained from 100 independent runs. The FSA-SVM and FSA-LB detect the true variables more often and obtain better AUC numbers than the penalized methods and Logitboost, with the FSA-LB being just slightly better than FSA-SVM.
TABLE 2
Experiments on synthetic data with $\rho = 0.9$, averaged over 100 runs.

| $N$ | $M$ | $k$ | $\text{FSA}$ | $\text{FSA-LB}$ | $\text{L}_1$ | $\text{MCP}$ | $\text{SCD}$ | $\text{LB}$ | $\text{FSA}$ | $\text{FSA-LB}$ | $\text{L}_1$ | $\text{MCP}$ | $\text{SCD}$ | $\text{LB}$ |
|-----|-----|-----|-------------|---------------|--------|--------|--------|--------|--------|-------------|---------------|--------|--------|--------|--------|
| 300 | 1000 | 10  | 27 | 31 | 0 | 3 | 1 | 0 | 0.990 | 0.991 | 0.991 | 0.956 | 0.934 | 0.950 |
| 300 | 10000 | 10 | 16 | 18 | 0 | 4 | 4 | 1 | 0.983 | 0.986 | 0.881 | 0.938 | 0.921 | 0.946 |
| 1000 | 1000 | 10 | 100 | 100 | 2 | 39 | 25 | 44 | 0.999 | 0.999 | 0.925 | 0.967 | 0.933 | 0.967 |
| 3000 | 1000 | 10 | 100 | 100 | 33 | 65 | 63 | 97 | 0.999 | 0.999 | 0.971 | 0.975 | 0.973 | 0.971 |
| 1000 | 1000 | 30 | 10 | 11 | 0 | 0 | 0 | 0 | 0.993 | 0.994 | 0.993 | 0.937 | 0.937 | 0.956 |
| 5000 | 1000 | 30 | 100 | 100 | 0 | 8 | 14 | 4 | 0.999 | 0.999 | 0.919 | 0.950 | 0.975 | 0.976 |

TABLE 3
Average training times (seconds) on the synthetic data from Table 2.

| $N$ | $M$ | $k$ | $\text{FSA}$ | $\text{FSA-LB}$ | $\text{L}_1$ | $\text{MCP}$ | $\text{SCD}$ | $\text{LB}$ |
|-----|-----|-----|-------------|---------------|--------|--------|--------|--------|
| 300 | 1000 | 10  | 0.02 | 0.03 | 17 | 70 | 68 | 0.13 |
| 300 | 10000 | 10 | 0.09 | 0.09 | 1240 | 749 | 644 | 1.5 |
| 1000 | 1000 | 10 | 0.07 | 0.06 | 469 | 352 | 332 | 0.44 |
| 3000 | 1000 | 10 | 0.15 | 0.23 | 705 | 1122 | 1103 | 1.3 |
| 1000 | 1000 | 30 | 0.08 | 0.13 | 469 | 352 | 332 | 1.2 |
| 3000 | 1000 | 30 | 0.2 | 0.26 | 469 | 352 | 332 | 4.1 |

The training times are shown in Table 3, where we see that FSA-SVM is the fastest, closely followed by FSA-LB. There exist algorithms [21], [33] for $L_1$-penalized logistic regression in the literature that are faster than [20]. Even though the training times might be greatly reduced compared to the interior point method [20], these algorithms optimize the same $L_1$ penalized loss function as [20], so the variable detection rates and AUC’s should be similar to or worse than those in Table 2.

7 LEARNING A 3D FACE MODEL FROM 2D ANNOTATIONS

The face 3D model matrix $F$ is obtained from a number of 2D face images where the key-points have been annotated. In this work we used the AFLW dataset annotations, containing about 26000 faces and their annotations.

Let $P_i = (p_{i1}, ..., p_{iL})$ be the 2D coordinates of the $L$ keypoints for face $i$, $i = 1, n$. Write $P_i = (X_i, Y_i)$, obtaining the row vectors $X_i, Y_i$ as the $x$ and $y$ coordinates of the $L$ keypoints.

The goal is to find the matrix $F$ of size $3 \times L$ and the projected rigid transformations $\theta_i, i = 1, n$ for the annotated faces, to minimize

$$E(\theta, F) = \sum_{i=1}^{n} \| u_i 1 + s_i \pi(R_i F) - P_i \|^2 = \sum_{i=1}^{n} \| T_{xi} F - X_i \|^2 + \sum_{i=1}^{n} \| T_{yi} F - Y_i \|^2,$$

where each $T_{\theta_i}$ is a $2 \times 3$ matrix with rows $T_{xi}^T, T_{yi}^T$.

The minimization starts with a random $F$ and alternates two steps until convergence:

1) Given current $F$, fit the projected rigid transformations $\theta_i$ using Algorithm 3 for each face
2) Given current $\theta_i$, find $F$ by minimizing eq. (14)

This approach is a simplified version of the Stratified Procrustes Analysis [2].