IDIAP RESEARCH REPORT

FAST BOUNDING BOX ESTIMATION BASED FACE DETECTION

Venkatesh Bala Subburaman Sébastien Marcel

Idiap-RR-38-2010

NOVEMBER 2010

Centre du Parc, Rue Marconi 19, P.O. Box 592, CH - 1920 Martigny
T +41 27 721 77 11 F +41 27 721 77 12 info@idiap.ch www.idiap.ch

Fast Bounding Box Estimation based Face Detection

Bala Subburaman Venkatesh and Sébastien Marcel
Idiap Research Institute
P.O. Box 592
CH-1920 Martigny
Switzerland
venkatesh.bala@idiap.ch
marcel@idiap.ch

March 2010

Summary

The sliding window approach is the most widely used technique to detect an object from an image. In the past few years, classifiers have been improved in many ways to increase the scanning speed. Apart from the classifier design (such as cascade), the scanning speed also depends on number of different factors (such as grid spacing, and scale at which the image is searched). When the scanning grid spacing is larger than the tolerance of the trained classifier it suffers from low detections. In this paper we present a technique to reduce the number of miss detections while increasing the grid spacing when using the sliding window approach for object detection. This is achieved by using a small patch to predict the bounding box of an object within a local search area. To achieve speed it is necessary that the bounding box prediction is comparable or better than the time it takes in average for the object classifier to reject a subwindow. We use simple features and a decision tree as it proved to be efficient for our application. We analyze the effect of patch size on bounding box estimation and also evaluate our approach on benchmark face database. Since perturbing the training data can have an affect on the final performance, we evaluate our approach for classifiers trained with and without perturbations and also compare with OpenCV. Experimental evaluation shows better detection rate and speed with our proposed approach for larger grid spacing when compared to standard scanning technique.

1 Introduction

The sliding window approach is the most common technique used for object detection [3, 15, 17]. A classifier is evaluated at every location, and an object is detected when the classifier response is above a preset threshold. Many systems need face processing tasks (detection, tracking, recognition), and needing them to run in real-time with out loosing much of individual performance has become a challenging task. Cascades introduced by Viola et al. [17] speed up the detection by rejecting the background quickly and spending more time on object like regions. Although cascades were introduced, scanning with fine grid spacing is still computationally expensive. To increase the scanning speed one approach is to train a classifier with perturbed training data to handle small shifts in the object location. Another simple approach is to increase the grid spacing (decreases the number of subwindows being evaluated). Unfortunately, as the grid spacing is increased the number of detection decreases rapidly.

Recently efficient subwindow search (ESS) proposed by Lampert et al. [8] for object detection finds bounding box using branch and bound method in sublinear time. This method requires histogram of features to estimate the upper and lower bound of a classification function. Though the ESS provides optimal solution in finding the best bounding box, Lehmann et al. [9] pointed out that calculating integral histogram for large number of bins requires large memory, which can be a constraint in running on certain hardware.

The other technique which has become popular for object detection is the generalized Hough transform. One of the model proposed by Leibe et al. [10], Implicit Shape model (ISM), consists of class specific codebook of local appearance from the object category and spatial probability distribution of where the codebook entry may be found on object. During recognition this information is used to perform a generalized Hough transform in a probabilistic framework. However as pointed out by Jürgen et al. [6], codebook-based Hough transform comes at a significant computational price, and the authors have suggested using random forest to directly learn a mapping between the appearance of an image patch and its Hough vote, more precisely a probabilistic vote about the position of an object centroid.

In this paper we focus on increasing the detection rate and speed of the sliding window approach by building a bounding box estimator with high performance (speed and accuracy). To reach that objective we propose a method to predict the bounding box of an object, using a simple yet effective binary test and a decision tree. We show that this method reduces the miss detections while increasing the scanning grid spacing. In this work we don't intend to increase the performance of the main face classifier, but rather try to improve the detection rate for larger grid spacing which also has an effect on scanning speed.

This paper is organized as follows. In next Section we describe the related work, and in Section 3, we describe our approach on how we increase the detection rate and speed by using bounding box estimation. In Section 4, we show our experiment results and finally, conclusion and future work are given in Section 5.

2 Related work

Object detection has been approached in many different ways in the literature. Either parts of the object or the whole object have to be classified in some way. The main idea of detecting parts rather than whole image object is to reduce the variability in appearance of the object. Bounding box estimation can also be posed as an object part identification problem. This has been done using different features but the most popular ones are scale-invariant feature transform (SIFT) [11] and Ferns [12].

SIFT descriptors have been popular as it has proved to be robust to illumination changes, but the computation of SIFT descriptors turns out to be costly. Feature matching is done with Nearest Neighbor approach, but to reduce the complexity, an approximate algorithm called the Best Bin First (BBF)

algorithm proposed by Beis et al. [1] is used. In [12] an alternative feature called Ferns was introduced which showed comparable or better performance than SIFT features for patch identification. Ferns consists a set of binary features, and the binary feature is obtained by comparing the intensity value of two pixels. A Semi-Naive Bayesian classifier is used to identify a patch, where the class posterior probabilities are modeled with hundreds of binary features.

In [13] the bounding box and pose are estimated before giving the hypothesized window to a pose specific Support Vector Machine (SVM) classifier. Histogram of SIFT like features, and a Naive Bayesian approach is used to learn different poses and bounding box. The bounding box estimation is carried out in two steps. First a fixed sized window is used to infer the aspect ratio, and then the area is estimated. This approach maximizes the overlap between the estimated location of the box and the ground truth. In [2], a component based face detector is described. Their system consists of a two-level hierarchy of SVM classifiers. On the first level, components of face are independently detected and the second level, the geometrical configuration of the detected components are checked with face model. While their technique might perform better, but since many component classifiers are evaluated the speed could be an issue.

Our work is inspired by [6] in the way that the object centers are estimated by using Hough forest. Each leaf node in the tree gives a probabilistic votes of the object centroid. The hypothesis which is tested at each node is a simple comparison of values (can be intensity or gradient in x and y direction) at two locations (in some sense similar to [12]). In [6], a forest with 15 trees is used which takes considerable amount of time if a dense scan is performed. Therefore we decided to use only a single decision tree for estimating the offset (bounding box) of the face. The advantage of using a decision tree is that the number of tests that needs to be performed grows only logarithmically thus saving computational time at runtime.

3 Proposed approach

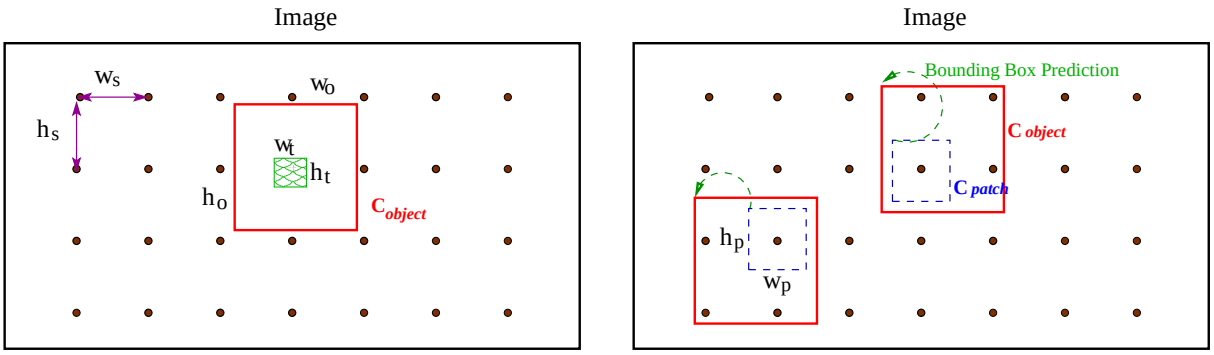
In this section we first analyze the standard sliding window approach, and then describe how to reduce the miss detections with larger grid spacing by using a bounding box estimator. We then describe how a decision tree is learnt for this task.

3.1 Analysis of standard sliding window technique

The standard sliding window technique with regular grid scan is shown in Figure 1(a), where a classifier C_{object} is placed on the scanning grid and checks if it is an object or not. We start by formulating the chance of hit $\mathcal{H}_c$, as the chances for the target object to be within the classifier detection range, with respect to the scanning grid interval (w_s, h_s), and to the translation tolerance (w_t, h_t) of the classifier C_{object}, (see Figure 1(a)).

$$\mathcal{H}_c \approx \frac{w_t h_t}{w_s h_s} \tag{1}$$

As an example, lets assume that the object present in the image is of the same size as the classifier is trained with, if $w_t = h_t = 3$ and $w_s = h_s = 6$ then the chance of getting a hit $\mathcal{H}_c$ is 0.25, which is very low. For $\mathcal{H}_c$ greater than 1, means that the classifier has more chances to detect the object. As we decrease w_s and h_s (a finer search), $\mathcal{H}_c$ increases, while scanning speed decreases (slower). Our goal is to increase $\mathcal{H}_c$ without decreasing too much of the scanning speed (thus making it faster), which is described below.



(a) Standard scanning technique (b) Our proposed scanning framework

Figure 1: Standard scanning technique vs our proposed scanning framework. The dots represent the scanning grid with interval (w_s, h_s), target object size (w_o, h_o), translation tolerance (w_t, h_t) of target object classifier C_{object}, target patch size (w_p, h_p), and target patch classifier C_{patch}. The classifier C_{patch} predicts the bounding box for C_{object} in our approach.

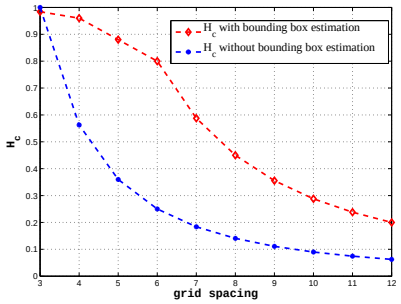


Figure 2: Estimated chance of hit $\mathcal{H}_c$ with and without bounding box estimation with respect to scanning grid spacing.

3.2 Chance of hit with our approach

In this subsection we explain how our method increases the probability of hit. Figure 1(b), shows the proposed scanning framework. The classifier C_{patch} is evaluated on a regular grid, while the main classifier C_{object} is placed on location predicted by C_{patch}. Assuming that we have a classifier C_{patch} that predicts the patch location correctly within the translation tolerance (w_t, h_t) of the classifier C_{object}, with prediction rate d_p, then the chance of hit can be approximately given by:

$$\mathcal{H}_c \approx d_p \mathcal{H}_p \tag{2}$$

$$\mathcal{H}_p = \frac{(w_o - w_p + 1)(h_o - h_p + 1)}{w_s h_s} \tag{3}$$

where $\mathcal{H}_p$ is the chance of hit for the patch, (w_p, h_p) is the patch width and height, and (w_o, h_o) is the object width and height, with constraints $w_p < w_o$ and $h_p < h_o$ (see Figure 1(b)). For example if, $w_p = h_p = 14$, $w_o = h_o = 19$, $w_s = h_s = 6$, and $d_p = 0.8$ (this value is taken from our experiment

results), we get $\mathcal{H}_c = 0.8$, which is 55% greater than standard scanning approach. The smaller the patch size is, the more the spacing between the grid can be, for a increase in scanning speed. Unfortunately at the same time estimating the bounding box becomes complex as individual patch will contain less and less information for distinguishing one from another. Figure 2 shows the chance of hit with and without bounding box estimation. To generate the plot we have used the same values as given in the example.

3.3 Bounding box estimation

The key idea for our algorithm lies in estimating the bounding box with high performance (speed and accuracy). We intend to use decision tree as it proved to be simple and efficient for our task. The decision tree is trained in a supervised manner. The training data, binary test and tree construction, and data stored in leaf node is described below.

Training data. For an object of size $w_o \times h_o$, and patch size of $w_p \times h_p$, we can have $\frac{(w_o - w_p + 1)(h_o - h_p + 1)}{2}$ number of overlapping patches (see Figure 3(a)). We represent a set of patches by $\{\mathcal{P}_i = (\mathcal{I}_i, \boldsymbol{d}_i)\}$, where $\mathcal{I}_i$ is the appearance of the patch and $\boldsymbol{d}_i$ is the offset of the patch. The offset vector $\boldsymbol{d}_i$ is a 2D vector representing (x, y) shifts from the object center or from a fixed point in the object.

Binary test. In a decision tree $\mathcal{T}$, a test has to be performed at a node. We first consider a simple binary test introduced in [6, 12], which is given by:

$$t_f(\mathcal{I}) = \begin{cases} 1 & \text{if } \mathcal{I}(x,y) \leq \mathcal{I}(x\prime, y\prime) \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where (x, y) and $(x\prime, y\prime)$ are two locations in the patch $\mathcal{I}$. We also propose a new test which is given by:

$$t_{\mu f}(\mathcal{I}) = \begin{cases} 1 & \text{if } \mathcal{I}(x,y) \leq avg(\mathcal{I}) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $avg(\mathcal{I})$ is the average of the pixel values in the patch $\mathcal{I}$. Our test requires only half the number of pixel access compared to the previous test, but requires an integral image to quickly calculate the average value.

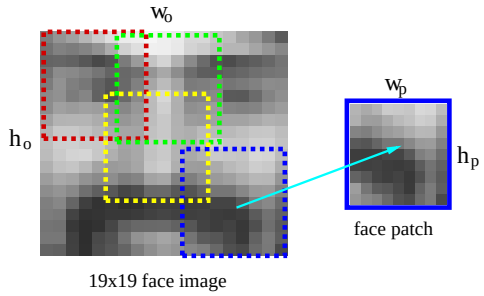
Tree construction During training, each non-leaf node picks the binary test that splits the training samples in an optimal way. We use the offset uncertainty as in [6] which is defined as:

$$U(A) = \sum_{i \in A} (\boldsymbol{d}_i - \boldsymbol{d}_A)^2 \tag{6}$$

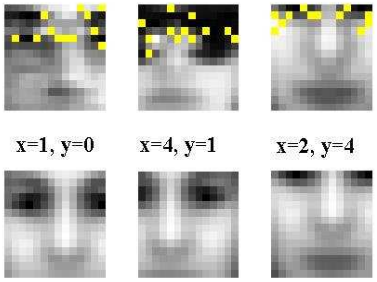
where d_A is the mean offset vector over all object patches in the set $A = \{\mathcal{P}_i = (\mathcal{I}_i, \boldsymbol{d}_i)\}$. A binary test $t^\star$ is chosen to minimizes the following expression:

$$t^\star = \arg \min_{t=1,\ldots,T} (U(A_L) + U(A_R)) \tag{7}$$

where T is the number of possible binary tests, and A_L and A_R are the subset of training samples reaching the left node and the right node respectively. Each leaf node l in the constructed tree stores a



(a)



(b)

Figure 3: (a)Examples of some overlapping face patches, All patches lie within the face region. (b) In this figure each column corresponds to a leaf node. The first row shows the average of the test images that arrives at a leaf node. The pixel locations that are evaluated ($t_{\mu f}$) at the nodes of a tree are also indicated. The second row provides the estimated offset value of the patches reaching a leaf node. The third row shows the average image of all the test patches having the offset value corresponding to the leaf node. The patch size of 14x14 is shown here.

single offset vector (x_l, y_l). If A_l is the subset of training examples that arrives at the leaf node l, then (x_l, y_l) is given by:

$$x_l = \frac{1}{|A_l|} \sum_{k \in A_l} x_k \qquad y_l = \frac{1}{|A_l|} \sum_{k \in A_l} y_k \tag{8}$$

Similarly to [6], we use two stopping criteria for the construction of the tree: the maximum depth of the tree and the minimum number of samples at a node. If a node has this minimum number of samples, we add an additional constraint which checks the variance of the offset vectors with a specified threshold. This way we have a better estimate of the offset at the leaf nodes. At runtime, a patch is given to the tree and the estimated offset value at the leaf node is used to place the main object classifier for subsequent detection. For illustration, we show in Figure 3(b) the pixel locations (x, y) associated to the tests $t_{\mu f}$ at each node in the tree from the root to different leafs. In these examples, we basically observe that the tree learns the shifts near the eye location.

4 Experiments

We first evaluate the performance of bounding box estimation for different patch size. We then compare our proposed scanning framework with standard scanning technique with respect to detection rate, false alarm rate and scanning speed on benchmark face database.

4.1 Evaluation of bounding box (bbx) estimation

We evaluate the performance of bounding box estimation for different patch sizes (w_p, h_p) and for two different types of binary test t_f and $t_{\mu f}$. We first describe the training and testing face dataset and parameters set for training the decision tree. We obtain approximately 35,000 cropped face images

(19x19) (faces are scaled and cropped with respect to eye location) from standard face database (BANCA, BIOID, Purdue, and XM2VTS). A subset of 15,000 face images are used for training, 10,000 are used for validation and the rest 10,000 are used for testing. The dataset which we used for this evaluation have well defined eye locations and can assume that the patches have good groundtruth offset values.

To build a decision tree we set the maximum depth, minimum number of samples and variance threshold. The depth of the tree is varied from 12 to 15 depending on the patch size, but kept the same for two types of test. Large patch size have fewer offset values to be estimated, therefore we use smaller depth size, while smaller patch size have many offset values which creates more training samples and requires larger depth for better estimation. The variance threshold is set to 0.1 and minimum number of samples to 10 for all our experiments. A smaller variance will force the training samples to split if the offset values are too different. The total number of possible binary tests for t_f is $\frac{(w_p \times h_p)(w_p \times h_p - 1)}{2}$, which is large. Hence a fixed number of tests are evaluated (200 in our case) at every node, and for each test the pixel pairs are picked randomly. The total number of possible binary tests in the case of $t_{\mu f}$ is $w_p \times h_p$, as it compares a pixel value to the average value of the patch. Since the number of test evaluation is small, each node evaluates all the test and selects the best one based on (7).

Training of a decision tree proceeds by giving all the samples at the root node and recursively splitting the training samples using (7) until it reaches the maximum depth or the variance of the samples at a node reaches below a specified threshold value. At test time, a patch is passed through the tree, and the leaf node gives an offset estimate $(\hat{x}, \hat{y})$. Since we want to measure how close the estimated offset is to the true offset we use squared L_2 norm to evaluate the estimation error:

$$\lambda = (\hat{x} - x)^2 + (\hat{y} - y)^2 \tag{9}$$

where (x, y) is the true offset value of the patch.

To inspect the distribution of error λ, we define $g(\lambda)$ as the number of test patches that have estimation

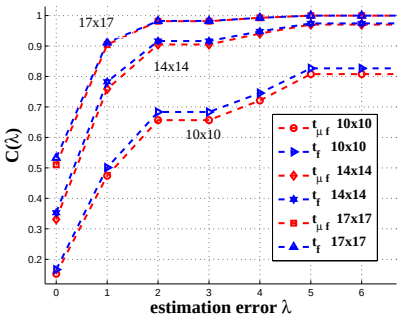
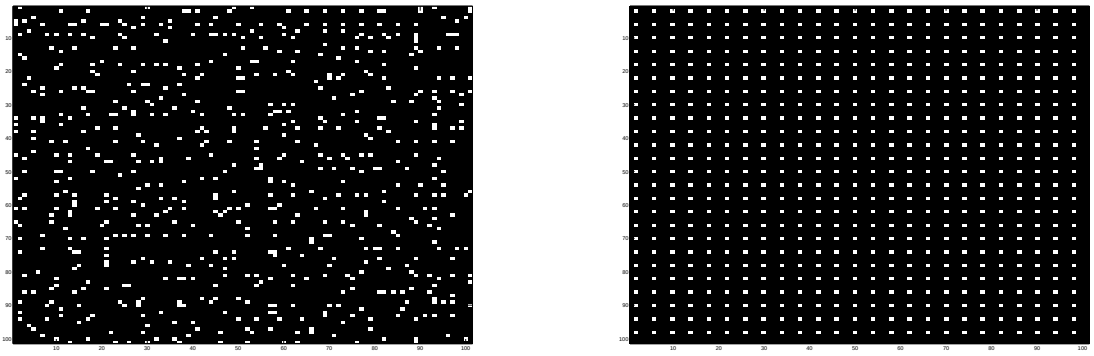


Figure 4: Cumulative distribution of estimation error λ for patch sizes of 10x10, 14x4, and 17x17. The figure shows only the cumulative distribution for first few λ's.

error of λ, and the cumulative distribution of estimation error as $c(\lambda) = \sum_{j=0}^{\lambda} g(j)$. Figure 4 shows the cumulative distribution of estimation error for square patch sizes of 10, 14 and 17. We see that there is only a slight performance difference between the two types of test.

4.2 Evaluation on benchmark face database

We now compare the detection rate and speed of scanning with and without bbx estimation on CMU+MIT [16] and Fleuret [4] frontal face databases. These databases have a total of 375 images and 1085 faces of var-



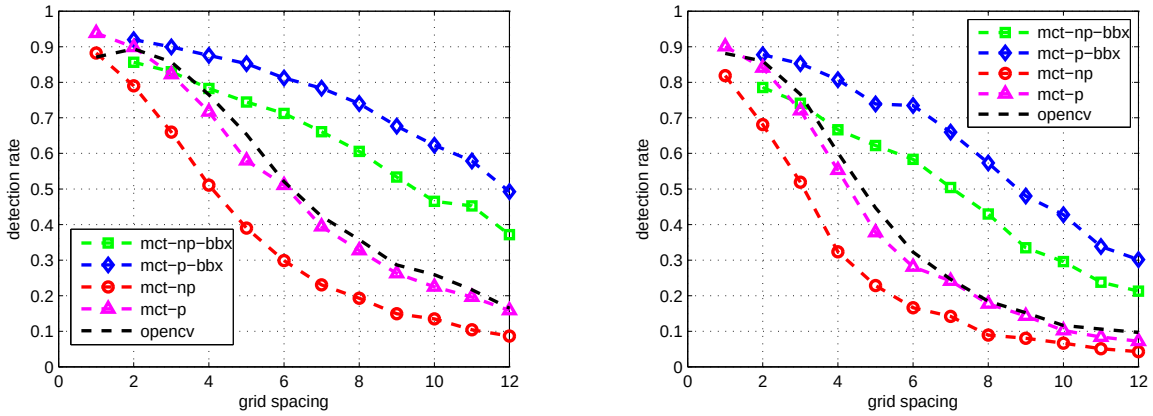
(a) estimated locations (our approach) (b) regular grid spacing

Figure 5: A dot shows the top left corner of the subwindow given to the classifier. (a) estimated locations with our approach, (b) for regular grid spacing.

ious size. A pyramid based scanning approach [15] is used to detect faces at different scales. Multiple detections are merged by averaging the detection within a certain radius which is a function of scale. The estimated eye coordinate of merged detection are compared with ground truth eye coordinates using Jesorsky measure [7], which is set to 0.6 for all our experiments. We describe first the different face classifiers used for our experiments, then show how the location on the grid gets modified with bounding box prediction. We then show how our approach is better than standard scanning technique irrespective of the classifier used.

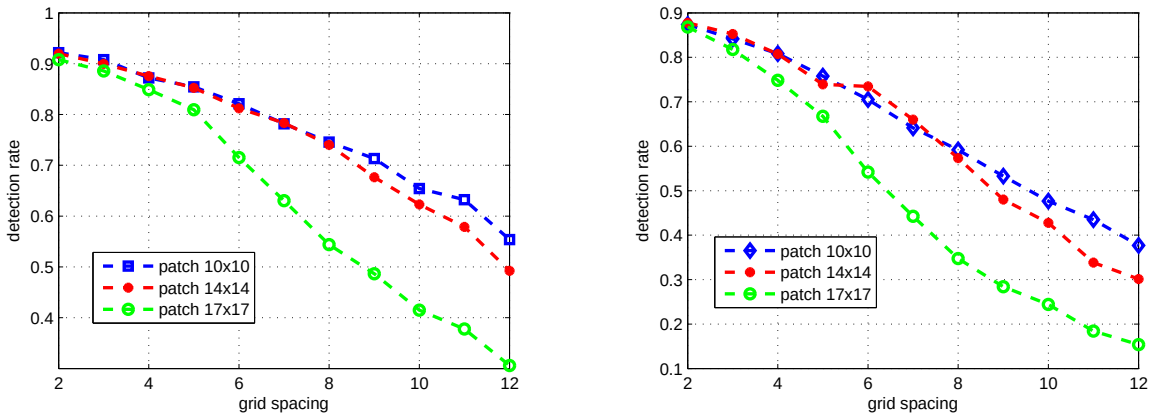
Main classifier. Many different classifiers and features are available for face detection task. We choose Modified Census Transform (MCT) features as it has been shown to be robust to lighting variations and does not require any preprocessing [5]. Boosting is used to train the classifiers and the cascade architecture is used for speeding up detection process. Two different classifiers are trained with different sets of face training data. One classifier is trained without perturbing the face training data (C_{mct-np}) and the other with perturbed face training data (C_{mct-p}). The perturbation consists of shifting the face by one pixel in x and y directions. We also use the OpenCV face classifier (C_{opencv}) [14] to compare the performance (detection rate and false alarms) with our approach for different grid spacing. Figure 5 shows the locations at which the main classifier is placed with and without bbx prediction.

Detection rate vs grid spacing. Figure 6 shows the performance of detection rate with respect to grid spacing. We can see that classifier C_{mct-p} performs similar to C_{opencv}. Training without perturbation has lower detection rate as the grid spacing is increased which is obvious. In the figure `mct-p-bbx` and `mct-np-bbx`, represents the performance with bbx prediction. It can seen clearly that our approach has higher detection rate for both the MCT based classifiers. It can also be seen that there is an equal amount of drift when different classifiers are used with and without bbx prediction. This might be due to that our approach is independent of the classifier used. Figure 7 shows the detection rate for different patch sizes when the grid spacing is varied. It can be seen that smaller patches are able to perform better compared to larger patches when the grid spacing is larger which goes along with our hypothesis (Section 3.2).



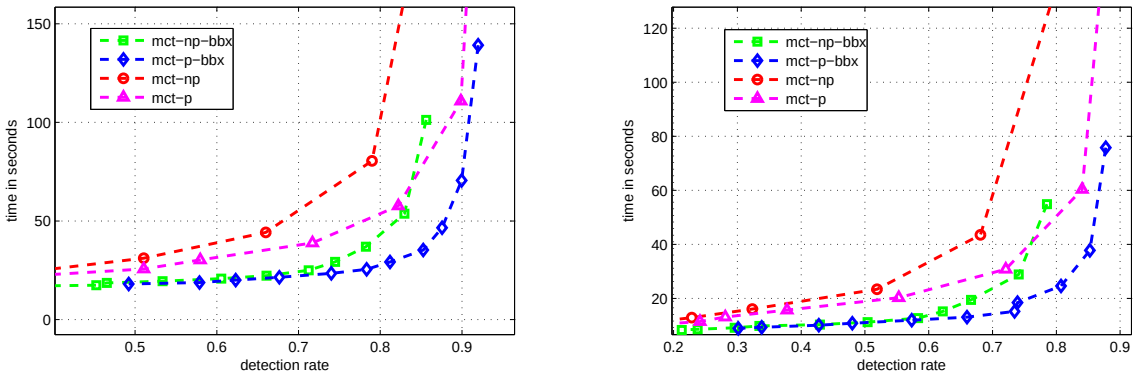
(a) Scaling factor 1.2 (b) Scaling factor 1.5

Figure 6: Detection rate vs grid spacing with and without bbx prediction.



(a) Scaling factor 1.2 (b) Scaling factor 1.5

Figure 7: Detection rate vs grid spacing with bbx prediction for different patch sizes.



(a) Scaling factor 1.2

(b) Scaling factor 1.5

Figure 8: Time in seconds to scan 375 images vs detection rate with and without bbx prediction.

Detection rate vs time In Figure 8, we can see that we can achieve better detection rate for a fixed time, or better time (increase in speed) for a fixed detection rate. We also see that C_{mct-p} has more detection rate than C_{mct-np}, but our approach performs better when using the same classifier.

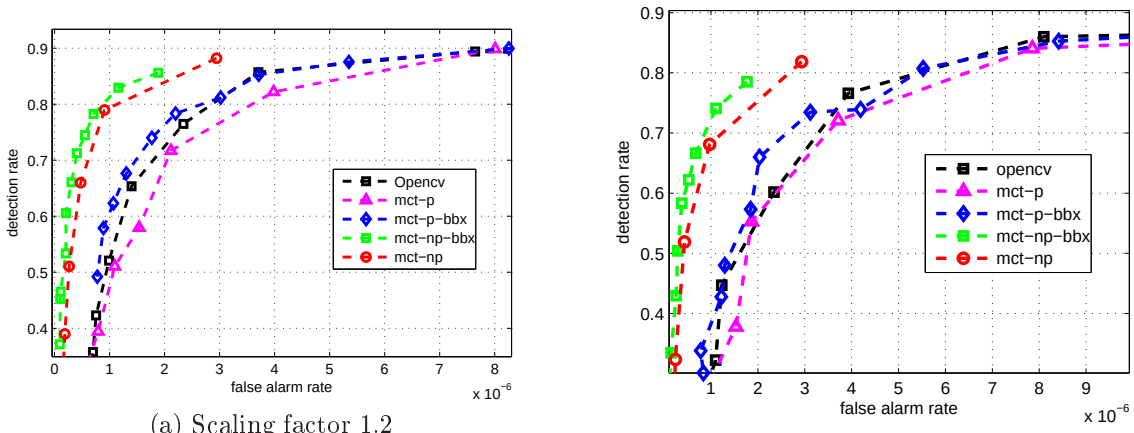
Detection rate vs false alarm rate. In Figure 9 it can be seen that the false alarm rate for C_{opencv} and C_{mct-p} are similar, while C_{mct-np} has lower false alarm rate. This indicates that training without perturbation has lower false alarm at the price of detection rate. We can also see that the false alarm is lower when using bbx prediction for a fixed detection rate.

5 Conclusion and future work

We have presented a method to improve the detection rate and speed while increasing the grid spacing in a sliding window based scanning technique. The key idea is to predict the bounding box of the object with high performance (both in speed and accuracy) which is achieved by using a decision tree with simple binary test at each node. We have used benchmark face databases to validate our approach and demonstrated that there can be an improvement in speed for a fixed detection rate, or better detection rate for a fixed time. We also see that the false alarm rate for a given detection rate is lower with our approach. We plan to further extend the idea of using decision tree to estimate other parameters like scale and pose of an object.

6 Acknowledgment

The authors would like to thank the Swiss National Science Foundation (projects 200020-122062 and 51NF40-111401) and the FP7 European MOBIO project (IST-214324) for their financial support.



(a) Scaling factor 1.2

(b) Scaling factor 1.5

Figure 9: Detection rate vs false alarm rate with and without bbx prediction.

References

[1] J.S. Beis and D.G. Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1000–1006, June 1997.

[2] S. M. Bileschi and B. Heisele. Advances in component based face detection. In *IEEE International Workshop on Analysis and Modeling of Faces and Gestures*, 2003.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[4] F. Fleuret. Fast binary feature selection with conditional mutual information. *Journal of Machine Learning Research*, 5:1531–1555, 2004.

[5] B. Froba and A. Ernst. Face detection with the modified census transform. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 91–96, May 2004.

[6] Jürgen Gall and Victor Lempitsky. Class-specific hough forests for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Miami, USA, 2009.

[7] O. Jesorsky, K. Kirchberg, and R. Frischholz. Robust face detection using the Hausdorff distance. In *International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 90–95, Halmstad, Sweden, 2001.

[8] Christoph H. Lampert, Matthew B. Blaschko, and Thomas Hofmann. Beyond sliding windows: Object localization by efficient subwindow search. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[9] Alain Lehmann, Bastian Leibe, , and Luc Van Gool. Feature-centric efficient subwindow search. In *IEEE Conference on Computer Vision*, October 2009.

[10] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. In *Proceedings of the Workshop on Statistical Learning in Computer Vision*, Prague, Czech Republic, May 2004.

[11] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[12] M. Özuysal, P. Fua, and V. Lepetit. Fast keypoint recognition in ten lines of code. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, Minneapolis, USA, 2007.

[13] M. Özuysal, V. Lepetit, and P. Fua. Pose estimation for category specific multiview object localization. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 778–785, Miami, USA, 2009.

[14] A. Kuranov R. Lienhart and V. Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *25th Pattern Recognition Symposium*, pages 297–304, 2003.

[15] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[16] Henry Schneiderman and Takeo Kanade. A statistical model for 3d object detection applied to faces and cars. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 746–751, June 2000.

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 511–518, 2001.