Learning SURF Cascade for Fast and Accurate Object Detection

Jianguo Li, Yimin Zhang
Intel Labs China

Abstract

This paper presents a novel learning framework for training boosting cascade based object detector from large scale dataset. The framework is derived from the well-known Viola-Jones (VJ) framework but distinguished by three key differences. First, the proposed framework adopts multi-dimensional SURF features instead of single dimensional Haar features to describe local patches. In this way, the number of used local patches can be reduced from hundreds of thousands to several hundreds. Second, it adopts logistic regression as weak classifier for each local patch instead of decision trees in the VJ framework. Third, we adopt AUC as a single criterion for the convergence test during cascade training rather than the two trade-off criteria (false-positive-rate and hit-rate) in the VJ framework. The benefit is that the false-positive-rate can be adaptive among different cascade stages, and thus yields much faster convergence speed of SURF cascade.

Combining these points together, the proposed approach has three good properties. First, the boosting cascade can be trained very efficiently. Experiments show that the proposed approach can train object detectors from billions of negative samples within one hour even on personal computers. Second, the built detector is comparable to the state-of-the-art algorithm not only on the accuracy but also on the processing speed. Third, the built detector is small in model-size due to short cascade stages.

1. Introduction

Object detection is one hot research topic in computer vision due to wide applications. Great advances have been made in the passed decade, especially since the milestone work by Viola and Jones [36]. To realize good generalization performance, more training data are required in the learning procedure. Nowadays, it is much cheaper to collect many training samples from Internet with small incremental human labeling efforts like Mechanical Turk [32]. However, big data make the learning a critical bottleneck. This is especially true for training object detectors [37, 23, 41]. As is known, the training is usually required to reach very low false positive rate per scan-window (FPPW) such as 10^{-6} [37], which means that hundreds of millions or even billions of negative samples should be processed during training procedure. Therefore, training object detector is a very time-consuming task. In early works, it may take months to train a high quality detector due to the limitation of computing resources. Even with the great increase of computing power today, existing learning frameworks are still not efficient to handle such a large scale training problem. It is still required several days or even weeks to obtain a high-quality detectors [42, 28, 41]. As some fine-tuning of parameters are usually required based on training experiments, the long training time yields very painful experiences for researches in the field of object detection.

Someone may argue that we just care the detection speed since the training only need running once. However, diverse data appear on Internet everyday, and some may not be well covered by existing detectors, thus yields possible hit-miss. This is true for most object categories. One way is to retrain and refresh detectors on new coming data to alleviate possible hit-miss. This is similar to what Google does on refreshing its pagerank and indexing. Hence, the research on training is still very useful since training is a critical infrastructure for visual recognition engine.

Besides the big data problem, another important factor is the convergence speed of the cascade training. To the best of our knowledge, almost all existing cascade detection frameworks are trained based on two conflicted criteria (false-positive-rate and hit-rate) for the detection-error tradeoff. Although some researches introduced intermediate or post tuning of cascade parameters with some optimization methods [40, 3, 31, 39], they did not touch the convergence speed of cascade training. As a result, the final cascade usually has very long stages.

This paper proposes a new cascade learning framework for object detection, with an emphasis on training efficiency. First, the detection window is represented by local SURF patches, which are spatial regions within the window and described by the multi-dimensional SURF descriptor [2]. The number of local SURF patches is limited to several hundreds in the detection window. Thereof, the feature pool size is greatly reduced, which makes feature selection much

easier. Moreover, SURF is much faster in feature extraction than existing local features like HoG [5]. Second, we adopt logistic regression as weak classifier on each local SURF patch. Third, we adopt AUC (Area under ROC curve) as a single criterion for convergence test during cascade training instead of the two conflicted criteria (false-positive-rate and hit-rate). We named the new training framework SURF cascade. The training of SURF cascade converges much faster and generates much shorter cascade stages. Experiments show that the proposed approach can build highly accurate detectors by processing billions of samples within one hour even on personal computers. In summary, this paper has three major contributions:

(1) We introduce variants of SURF descriptors for fast and accurate object detection.

(2) We propose AUC as a single criterion for cascade training, which makes the training converge faster and yields cascade model with much shorter stages.

(3) We show a system that can train cascade object detectors from billions of samples within one hour even on PCs.

In the rest of the paper, we will first revisit related works on object detection in Section 2, and present the details of the proposed framework in Section 3. Experiments are shown in Section 4. Conclusions are drawn in Section 5.

2. Viola-Jones Framework Revisited

The boosting cascade framework by Viola and Jones is a milestone work in object detection [36]. It inspired many following works. It is necessary to revisit the VJ framework before diving into more details. Basically, there are three key ideas that make it able to build real-time object detectors: the integral image trick for efficient Haar feature extraction, the boosting algorithm for ensemble weak classifiers, and the attentional cascade structure for fast negative rejection. The VJ framework has several limitations. First, the feature pool size of Haar-like features is very high, which is usually in hundreds of thousands level for a typical 20×20 detection template. This leads to extremely large feature search space for weak classifier learning. Some heuristic feature selection or filtering strategies have been proposed to speedup the training [24, 4, 27]. However, these kind of techniques are still unsatisfied, due to that the search space after filtering is still large, and there is possible accuracy loss from un-optimal search.

Second, Haar features have very limited representation capacity. It has difficulty to handle variations due to pose and illumination. On the one hand, some researches extended Haar features to rotated Haar-like features [21], joint Haar features [26], sparse features [14], polygon features [28], etc. These features can improve detection accuracy, but make feature space even larger, and thus the feature selection problem becomes even critical. On the other hand, some complex features are proposed and used in object detection, which includes histogram of oriented gradient (HoG) features [5, 42], region-covariance features [34], etc. They are much slower in computing than Haar features. This paper introduces some variant of SURF features [2] to describe local patches. It is closely related to HoG features in the cascade HoG framework [42]. However, SURF is much easier and faster in computing than HoG.

Third, the attentional cascade is trained based on two conflicted criteria: false-positive rate (FPR) f_i and hit-rate (or detection-rate) d_i. The overall FPR of a T-stage cascade is $F = \prod_{i=1}^{T} f_i$, while the overall hit-rate is $D = \prod_{i=1}^{T} d_i$. Due to detection error trade-off, the VJ framework suggests setting maximum FPR as $f_i = 0.5$ and minimum hit-rate such as $d_i = 0.995$ during training procedure. If wanting to reach overall FPR = 10^{-6}, it requires at least 20 stages ($0.5^{20} \approx 10^{-6}$) by the given global setting. This is inefficient since some stages may reach the goal without convergence. It is better that f_i can be adaptive among different stages such that we could easily reach overall training goal. Some methods design automatic schemes to tune the intermediate threshold of each stage [40, 3, 31, 4]. These methods may alleviate the painful manual tuning efforts, but have nothing to do with the convergence speed. Inspired by [22], we introduced AUC [8] as a single criterion for cascade convergence testing. This will realize adaptive FPR among different stages, and yield fast convergence speed and cascade model with much shorter stages.

The proposed approach has some relationship with part-based model [10]. In the early stage, part-based model trained part detectors separately and integrated them together [25] for detection. Later works trained mixtures of deformable part models even under the cascade framework [10, 11]. The proposed approach can be viewed as a simple case of the part-based model, when viewing each picked local patch as part-based model. However, the proposed approach just combines local parts in a discriminative way for each stage without any pictorial structures among parts. Although it may not have the same power to handle deformable objects, it is still valuable in object detection field due to its simplicity and efficiency.

3. SURF Cascade for Object Detection

The proposed approach contains four ingredients: SURF features for local patch description, logistic regression based weak classifier for each patch, boosting ensemble of weak classifiers for each stage, and AUC-based cascade learning algorithm. We describe them below separately.

3.1. Feature Description

Dense local patches: Given a detection window, we define rectangular local patches within it using similar scheme in [42]. For instance, given a 40×40 detection template,

we define patch with 4 spatial cells, and allow the patch size ranging from 12×12 pixels to 40×40 pixels. We slide the patch over the detection template with 4 pixels forward to ensure enough feature-level difference. We further allow different aspect ratio for each patch (the ratio of patch width and patch height). Different from [19], the 4 spatial cells can be configured not only 2×2, but also 4×1 and 1×4. [1]. In this way, 450 patches are generated within the 40×40 detection template. Each patch is represented by a 32-dimensional SURF descriptor.

SURF descriptor: This paper adopts SURF feature to describe local patches due to its balance of computing efficiency and representation capacity. SURF is a scale and rotation invariant detector and descriptor [2]. It has been successfully applied in applications like object recognition, image matching/registration, structure-from-motion, etc. This paper derives some variant of SURF descriptor for object detection, and does not use the keypoint detector part at all.

The SURF descriptor is defined over the gradient space. The gradient may have different granularity. In the simplest case, we define d_x as the horizontal gradient image obtained with the filter kernel $[-1, 0, 1]$, and d_y as the vertical gradient image by the filter kernel $[-1, 0, 1]^T$. The 8-element SURF (or 8-bin in simple) computes the sums of d_x and $|d_x|$ separately for $d_y < 0$ and $d_y \geq 0$, and the sums of d_y and $|d_y|$ for $d_x < 0$ and $d_x \geq 0$ for each spatial cell.

The computing of 8-bin SURF requires branches operation, which is bad for speed optimization. This paper tries a variant of SURF descriptor, namely T2 descriptor [38]. The 4-bin T2 descriptor is defined as $\sum(|d_x| - d_x, |d_x| + d_x, |d_y| - d_y, |d_y| + d_y)$ [2]. And we extended 4-bin T2 descriptor to 8-bin by concatenating an additional 4 bins from gradients of diagonal and anti-diagonal directions. Given 2D filter kernels $diag(-1, 0, 1)$ and $antidiag(1, 0, -1)$, we obtain leading diagonal gradient image d_u and anti-diagonal gradient image d_v, respectively. Therefore, the additional 4-bin is defined as $\sum(|d_u| - d_u, |d_u| + d_u, |d_v| - d_v, |d_v| + d_v)$.

Feature normalization: The local patch is divided into 4 spatial cells. The SURF descriptor is extracted in each cell. Concatenating features in 4 cells together yields a 32-dimensional (8×4) feature vector. To make the descriptor invariant to patch size, the feature vector needs normalization. Good normalization methods may also alleviate the impact from illumination/contrast variations. We study different normalization methods proposed in [5], and find that L_2 normalization followed by clipping and renormalization (L_2Hys) works the best. For more detail on how to apply L_2Hys normalization, please refer to supplementary.

[1] See supplementary for some configuration examples. Patches of 2×2 cells allow 1:1, 1:2, 2:1, 2:3, 3:2 aspect ratio. Patches of 4×1 cells only allow 4:1 aspect ratio. Patches of 1×4 cells only allow 1:4 aspect ratio.

[2] The absolute |.| operation can be implemented without branch operation, and accelerated with SIMD optimization. On the contrary, original SURF can't be optimized by SIMD.

Multi-channel integral images: We adopt the integral image tricks to speedup feature extraction [37]. Given 8-bin SURF for describing local patches, if building one integral image per bin separately, it needs 4×8 = 32 address access ops to obtain SURF feature from one cell. Here, we introduce the array-of-structure trick which packs data of 8 bins together with a structure, and generate multi-channel integral images. We only need 4 access ops for each cell with the multi-channel integral image. Furthermore, this array-of-structure trick allows using instruction-level parallelization (i.e., SIMD), not only on the address access operation, but also on the summation and subtraction operations for computing SURF feature in each cell. In summary, this trick can greatly speed up feature extraction .

Feature comparison: This paper adopts 8-bin T2 descriptor as it has similar representation capacity to original SURF and even HoG, while dominates others on the feature extraction speed. Practice shows that 8-bin T2 descriptor has almost the same accuracy as the original SURF descriptor in object detection, while it is about 1.5X faster in feature extraction. Meanwhile, 8-bin T2 descriptor is slightly worse than HoG in terms of overall cascade accuracy, while it is more than 2X faster than SIMD optimized HoG descriptor in feature extraction. In addition, 4-bin T2 descriptor is also applicable in object detection. The differences between 4-bin and 8-bin are two folds. First, 8-bin T2 descriptor has more representation capacity than that of 4-bin. As an example, to build detectors with the same accuracy, 8-bin T2 descriptor requires 8 stages, while 4-bin T2 descriptor requires 12 stages. Second, the memory requirement of 8-bin is twice as that of 4-bin. Hence, 4-bin T2 descriptor may be applied in some memory limited applications.

3.2. Weak Classifier

We built weak classifier over each local patch, and picked optimal patches in each boosting iteration from the patch pool. In the VJ framework, the weak classifier is decision tree. In some following works, people introduced other linear classifiers such as Fisher linear discriminant [18], linear asymmetric classifier [39, 4] and linear SVM [5]. This paper chooses logistic regression as weak classifier due to that it is a linear classifier with probability output.

Given SURF feature $\mathbf{x}$ over local patch, logistic regression defines a probability model (a.k.a logit model)

$$P(y = \pm 1|\mathbf{x}, \mathbf{w}) = \frac{1}{1 + \exp(-y(\mathbf{w}^T\mathbf{x} + b))}, \qquad (1)$$

where y = 1 for face samples, y = -1 for non-face samples, $\mathbf{w}$ is a weight vector for the model, and b is a bias term. Given training samples $\{\mathbf{x}_i, y_i\}_{i=1}^N$, the parameters can be found via minimizing the following objective

$$l(\mathbf{w}) = \sum_{i=1}^{N} \log(1 + \exp(-y_i(\mathbf{w}^T\mathbf{x}_i + b)) + \lambda\|\mathbf{w}\|_k^k,$$

while λ is a tunable parameter for the regularization term, and $\|\mathbf{w}\|_k^k$ means L_k norm (k=1,2) of the weight vector. This problem can be solved by algorithms in [7].

3.3. Learning Boosting Classifier

We trained the boosting cascade on local patches from large scale dataset. Suppose there are N training samples, and K possible local patches represented by d-dimensional (= 32) SURF feature $\mathbf{x}$, each stage is a boosting learning procedure with logistic regression as weak classifier. There are different variants of boosting algorithms to ensemble weak classifiers. We find that Gentle AdaBoost gives the best results over other variants [12].

Gentle AdaBoost thinks that the base learner fits a regression function over training data, and outputs a real value. Given weak classifiers $h_t(\mathbf{x})$,the strong classifier is

$$H^T(\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} h_t(\mathbf{x}).$$

In the t-th boosting round, we built K logistic regression models $\{h_k(\mathbf{x})\}_{k=1}^K$ for each local patch in parallel from the boosting sampled subset. We tested each model $h_k(\mathbf{x})$ in combination with model of previous t-1 rounds. That is to say, we tested $H^{t-1}(\mathbf{x}) + h_k(\mathbf{x})$ on all the N training samples. Each tested model will produce an AUC score $J(H^{t-1}(\mathbf{x}) + h_k(\mathbf{x}))$. We picked the one which produces the highest AUC score, i.e.,

$$H^t(\mathbf{x}) = \arg\max_{H^k,\ k=1:K} J(H^k = H^{t-1}(\mathbf{x}) + h_k(\mathbf{x})). \quad (2)$$

This procedure is repeated until the AUC score is converged, or the designed number of iterations is reached. Weak classifier h_t is trained on balanced positive and negative samples. To avoid overfitting, we restricted the number of used samples during training as in [40]. In practice, we sampled an active subset from the whole training set according to the boosting weight. It is generally good to use about $30\times d$ (if d=32, $30\times d$=960) positive samples and the same number of negative samples as the active training set.

The whole procedure is forward selection and inclusion of weak classifier over possible local patches. It is simple to extend the procedure with capability to backward removing redundant weak classifiers. This is not only able to shrink the number of weak classifiers in each stage, but also able to improve the generalization capability of the strong classifier. For details on including backward removing or even floating searching capability into boosting framework, please refer to [20]. In our implementation, for training speed consideration, we only add the backward removing capability to Gentle AdaBoost. Details of boosting learning algorithm is given in Table 1.

3.4. Cascade Training

Within one stage, we did not need to give threshold for intermediate weak classifiers. We just need to determine the decision threshold θ of the strong classifier $H^t(\mathbf{x})$. In the VJ framework [36, 42], the threshold is manual tuned on a validation set based on the two conflicted criteria, FPR and hit-rate. In our case, the threshold is much easier to be determined since ROC curve is consistently available. With ROC curve, FPR is easily determined when given minimal hit-rate d_{min}. We decreased d_i from 1.0 on the ROC curve, until reaching the transit point $d_i = d_{min}$. The corresponding threshold at that point is the desired θ.

After one stage is converged, we continued to train another stage with false-positive samples coming from scanning non-target images with partial trained cascade. We repeated this procedure until the overall FPR reach the goal. The cascade training algorithm is given in Table 2.

In the proposed approach, FPR is adaptive among different stages, and is usually much smaller than 0.5. For instance, we got an 8-stage cascade for face detection with f_i at each stage forming a vector like (0.305, 0.226, 0.147, 0.117, 0.045, 0.095, 0.219, 0.268). In comparison, the VJ framework requires 20 stages to reach the same goal (FPR=$10^{-6}\approx 0.5^{20}$). This means that AUC based cascade training can converge much faster. As a byproduct, this will not only make model-size very smaller (for instance, model-size of 8-stage cascade is only 50KB), but also increase the detection speed quite a lot.

Table 1. Learning Boosting Classifier for SURF cascade

1. Given training set: $\{(\mathbf{x}_i^k, y_i)\}_{i=1}^N$, $k = 1 : K$, where N is the number of samples and K is the number of local patches. $\mathbf{x}^k$ is d-dimensional SURF descriptor of the k-th local patch.
2. Initialize weight for positive samples and negative samples.
   (1) $w_{1,i}^+ = \frac{1}{N_+}$ for those y_i=1;
   (2) $w_{1,i}^- = \frac{1}{N_-}$ for those y_i= -1.
3. for $t = 1 : T$ boosting round
   3.1 Sampled $30\times d$ (d=32 in 2×2 SURF patch) positive samples and $30\times d$ negative samples from training set according to the weight as the active training subset;
   3.2 Parallel for each patch $\{\mathbf{x}_i^k, y_i\}$, train a logistic regression model $h_k(\mathbf{x}, \mathbf{w})$;
   3.3 For each h_k, combine it with existing model H^{t-1}; and evaluate on the whole training set to obtain AUC score $J(H^{t-1} + h_k)$;
   3.4 Choose the best model $H^t(\mathbf{x})$ according to Eq 2 to include weak classifier h_t;
   3.5 Update weight $w_{t,i} = w_{t,i}\exp[-y_i h^t(\mathbf{x}_i)]$;
   3.6 Normalize the weight so that $\sum_i w_{t,i}^+$ =1 and $\sum_i w_{t,i}^-$ =1;
   3.7 Compute $H^t(\mathbf{x})$ on the whole training set to obtain the AUC score J_t;
   3.8 If AUC value J_t is converged, break the loop.
4. While true // backward removing
   4.1 Find $h^- = \arg\max_{h_k} J(H^T \backslash h_k)$, where $H^T \backslash h_k$ means that H^T excludes weak learner h_k;
   4.2 If $J(H^T \backslash h^-) > J(H^T)$ on the whole training set, let $J(H^T) = J(H^T \backslash h^-)$; otherwise, break the loop.
5. Output final strong model $H^T(\mathbf{x})$ for this stage.

Table 2. Training SURF cascade based on ROC analysis

- Input: over all FPR F_t; minimum hit-rate per stage d_{min}; positive sample set X^+; negative sample set X^-.
- Initialize F_i = 1.0, D_i = 1.0, i=0.
- While $F_i > F_t$
  1. $i = i + 1$;
  2. Train one stage classifier $H^i(\mathbf{x})$ using samples of X^+ and X^- via the algorithm in Table 1;
  3. Evaluate the model $H^i(\mathbf{x})$ on the whole training set to obtain ROC curve;
  4. Determine the threshold θ_i by searching on the ROC curve to find the point (d_i, f_i) such that $d_i = d_{min}$;
  5. $F_{i+1} = F_i \times f_i$ and $D_{i+1} = D_i \times d_i$;
  6. Empty the set X^-;
  7. If $F_{i+1} > F_t$, adopt current cascaded detector to scan non-target images with sliding window and put false-positive samples into X^- until the size $|X^-| = |X^+|$.
- Output the boosting cascade detector $\{H^i(\mathbf{x}) > \theta_i\}$ and overall training accuracy F and D.

4. Experiments

We applied the proposed approach to face detection and car detection. This section will show the implementation details, evaluation results on public datasets and the detection speed of the proposed approach.

Note that the proposed approach is not limited to face detection and car detection. We also applied it to several real-world applications such as real-time hand gesture detection. Specially, we have made a high quality face detection SDK free available to public [3].

4.1. Implementation

We implemented all the training and detection modules in C/C++ on x86 platform. For the feature extraction part, we adopted the integral image trick to speedup the computation as we described in Section 3.1. For the cascade training part, we parallelized the time-consuming weak classifier training step with OpenMP in task level. Furthermore, we did SIMD optimization on some hotspots, which include the L_2Hys vector normalization of SURF features, the dot-product operation in logistic model (i.e., $\mathbf{w^T x}$), and the feature extraction from the packed 8-channel integral images. The detection module shares the same SIMD optimization with the training part [4].

The training and detection experiments were done on a personal workstation with 3.2GHz Core-i7 CPU (4 cores 8 threads SandyBridge) and 12GB RAM.

[3]Face detection and tracking based on SURF cascade are integrated in Intel perceptual computing SDK, which is available at http://www.intel.com/software/perceptual.

[4]For more details, please refer to the implementation FAQ at the project page in https://sites.google.com/site/leeplus/.

4.2. Face Detection

4.2.1 Frontal Face Detector

We collected training samples from Internet. Positive samples of frontal faces are mainly from the GENKI dataset [35], the facetracer dataset [17], the FERET dataset [29], etc. We discarded faces less than 32×32 pixels, and collected 15,000 frontal faces from these datasets. We further derived 15,000 faces with mirror transform, and 15,000 samples by random perspective transforming face image within [-10, 10] degree. All cropped and derived frontal faces are resized to 40×40. Finally, we had 45,000 positive training samples. The negative images are mainly from Caltech 101 dataset [9], Corel 5k image set, etc. We made some refinement on this negative set to remove images containing faces. Totally, we collected about 18,000 images without faces [5].

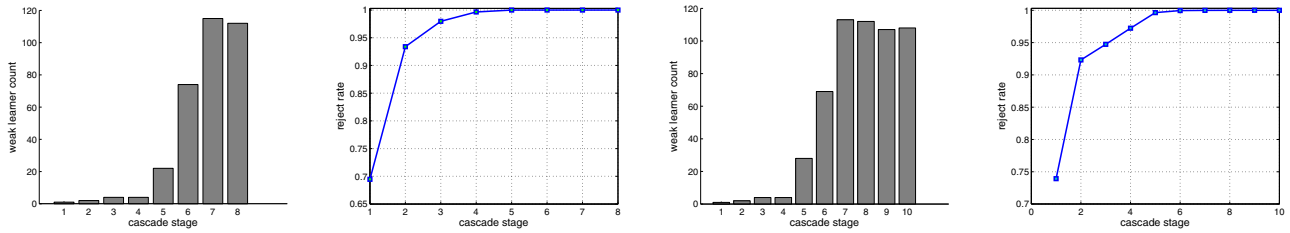
We placed 450 local patches on the 40×40 detection template as described in section 3.1. We set the maximum number of weak classifiers in each stage to 128. To obtain fast detector, we restricted that the first 3 stages have at most 4 weak classifiers. We set the minimum hit-rate to 0.995 for each stage and the overall FPR per window to 10^{-6}. Given this configuration, the training procedure is fully automatic. It took about 47 minutes (2,841 seconds) to converge at the 8th stage. During training, we scanned both the original and the mirror of each negative image with a scalable sliding window to get false-positive samples. In the end, more than 13.6 billions negative samples were processed.

In comparison, we tried to train face detector using the OpenCV Haar training modules on the same dataset. However, we can't finish the training within 2 days on the same computer with parallel processing tuned on. This means that SURF cascade is at least 60X faster than Haar cascade in training. Besides, we tried to replace AUC based criterion with VJ's criteria to control SURF cascade training, which requires more than 5 hours to converge. This experiment indicates that AUC based criterion brings about 6X (5 hours vs 47 min) speedup in training, while SURF representation brings the other 10X speedup.

Figure 1(a) and 1(b) illustrate details of the final cascade, including the number of weak learner in each stage and the accumulated rejection rate over the cascade stages. It shows that the first three stages rejects 98% negative samples with only 7 weak classifiers. The cascade detector contains 334 weak classifiers, and only need to evaluate 1.5 per window. On the contrary, the default Haar-based face detector in OpenCV contains more than 24 stages and 2,912 weak classifiers, and requires to evaluate more than 28 Haar features per window [21].

Figure 2(a) further depicts the top-3 picked local patches. We observed that the best local patch lies in the regions of

[5]Detail info about the training set is also available on the project page.

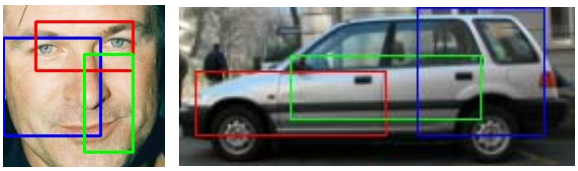


(a) (b) (c) (d)

Figure 1. (a) The number of weak classifiers at each stage of the face detector, (b) the accumulated rejection rate over the cascade stages of the face detector; (c) The number of weak classifiers at each stage of the car detector, (d) the accumulated rejection rate over the cascade stages of the car detector.

two eyes. This is similar to that of Haar-based detector [36].

The final detector contains a common post-processing step, which merges cascade outputs using the disjoint-set algorithm and filters unreliable results using the non-maximum suppression algorithm.



(a) (b)

Figure 2. Top-3 local patches picked by training procedure in the red-green-blue order (a) on the face object (b) on the car object.

4.2.2 Multi-view Face Detector

Besides frontal face detectors, we also trained multi-view face detector using the proposed approach. The multi-view detector is a parallel structure with independent detectors for each view: frontal view, left/right half-profile views (about 40∼60 degree of out-of-plane rotation to the frontal view), and left/right full-profile views (about 70∼90 degree of out-of-plane rotation to the frontal view). The frontal view detector is the same as the previous frontal detector. For half/full profile views, the training data are mainly from the CAS-PEAL database [13], the FERET database [29], and the CMU PIE database [30]. We collected 12,000 faces for half-profile view, and 8,000 faces for full-profile view. The detector training for each view follows the same procedure as the training of frontal view detector.

4.2.3 Face Detection Evaluation

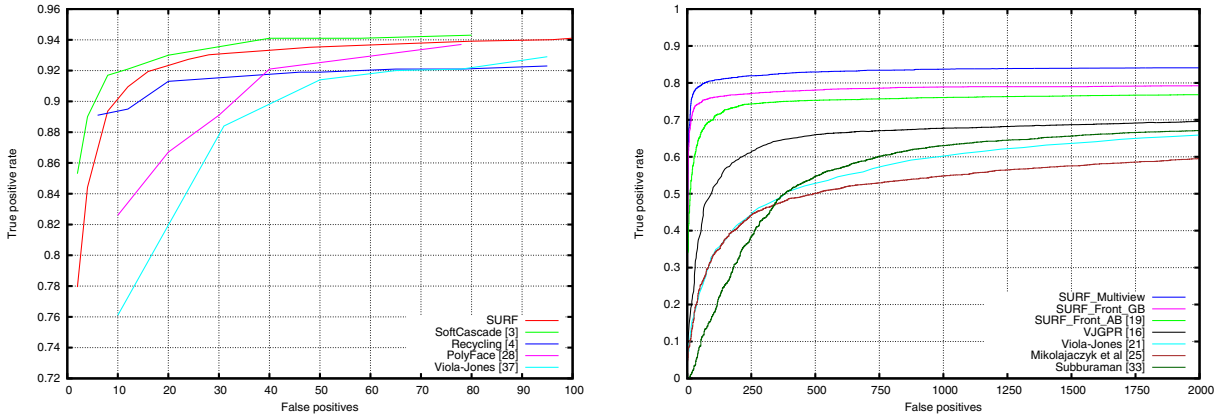
We evaluated SURF cascade detector on two public datasets: one is the CMU+MIT dataset, the other is the UMass FDDB dataset [15].

The standard CMU+MIT dataset consists of 130 gray images with 507 frontal faces. As SURF cascade can directly output probability score (in the range 0∼1) at any stage, it is natural to define score for each detection window w as $s(w) = p(w) + k(w)$, where $k(w)$ is the number of passed stages and $p(w)$ is probability output at the exit stage. With this score, we strictly followed the benchmark protocol suggested in [15], and generated the ROC curve as shown in Figure 3(a). Comparable results are depicted for some recent works in face detection such as the VJ detector [37], polygon-feature detector [28], soft cascade detector [3] and recycling-cascade detector [4]. Figure 3(a) shows that SURF cascade achieves comparable performance to the state-of-the-art method soft-cascade [3], while outperforms all the other methods.

The CMU+MIT dataset is a little out-of-date as it only contains gray-scale, relative low-resolution images, and the size of the dataset is too small to reflect nowadays data explosion status. Hence, the UMass face detection benchmark (FDDB) is introduced [15]. FDDB contains 2,845 images with a total of 5,771 faces under a wide range of conditions. Besides, it provides a systematic protocol to evaluate performance of face detection system. Figure 3(b) shows the discrete-score ROC curve generated by SURF cascade in comparison to available results on the benchmark [33, 21, 25, 16]. We also compared with our previous algorithm [19] which used Discrete AdaBoost for ensemble logit classifiers. It is obvious that SURF cascade outperforms others significantly, and Gentle AdaBoost is better than Discrete AdaBoost for ensemble logit classifiers. Furthermore, our multi-view detector yields significant improvement over pure frontal face detectors. Supplementary illustrates some examples of face detection results on CMU+MIT and UMass FDDB.

4.2.4 Detection Speed

We ran faces detector on videos to collect performance data. The frontal detector reaches 100 fps (frame-per-second) for a typical VGA video with single face in each frame, while the multi-view detector can process this video in real time. In comparison, the OpenCV default face detector can only achieve 60 fps with parallel processing tuned on. As is known, the OpenCV face detector is tailored optimized for



(a) (b)

Figure 3. (a) ROC curves for different methods on CMU+MIT dataset (b) ROC curves for different methods on UMass FDDB dataset.

processing speed.

This is due to two facts. First, SURF cascade detector has fewer number of stages (8 vs 24), fewer number of weaker classifiers (334 vs 2,912) and fewer average number of evaluated weak classifiers per detection window (1.5 vs 28) than that of OpenCV Haar detector. These advantages can compensate one weakness point that the computing on each weak classifier is higher than that of OpenCV detector. Second, SURF cascade benefits more from optimization than Haar cascade. The 8-stage SURF cascade has better workload load balance among threads in parallelization than that of 24-stage Haar cascade. Besides, SURF cascade is much easier for SIMD optimization (i.e., $\mathbf{w}^{\mathbf{T}}\mathbf{x}$ in logit model, etc) than that of Haar cascade.

4.3. Car Detection

For car detection, we collected 600 side view car samples from PASCAL VOC 2005 dataset [6, 1], containing the UIUC subset and ETHZ subset. We further derived 600 samples with mirrors, 600 samples with resampling (downsampling 10% followed by upsampling 10%), and 1200 samples with random perspective transforming within ± 5 degrees, etc. All cropped and derived car samples are resized to 80×32 pixels. Finally, we had 4,200 positive training samples. The negative images are collected similar to the face detection task. In total, there are about 5,000 negative images without cars for training.

On the target 80×32 detection template, we defined patch size range from 16×16 to 80×32, and allowed the patch aspect ratio like 1:1, 1:2, 2:1, 3:1, 4:1, etc. Totally, we placed 502 local patches on the detection template. And we set the same training configuration as face detection. It took 27 minutes (1,672 seconds) to finish the training with about 3.7 billions negative samples processed.

The final cascade contains 10 stages. Figure 1(c) and 1(d) illustrate the number of weak classifier in each stage and the accumulated rejection rate over the cascade stages. It shows that the first three stages rejects 95% negative samples with 7 weak classifiers. On average, the detector only need to evaluate 2.2 local SURF patches per detection window. We showed the top-3 local patches picked by the training algorithm in Figure 2(b).

The test set is 200 images from the TUGRAZ subset in PASCAL VOC 2005 with near side-view cars [6], which contains cars with different scales and even some viewpoint variations to sideview. We ran the detector on the test set, and found 141 cars and 18 false alarms. In summary, the detection rate is 70%, while the FPPW is 2×10^{-6}, and the false-positive-per-image is 9%. This result looks promising considering that the detector is trained by the poor quality UIUC dataset [1]. The detection speed on this dataset (VGA resolution images) is about 56 fps. Some challenging detection results are shown in supplementary.

5. Conclusions

This paper presents SURF cascade for fast and accurate object detectors. The proposed approach brings three key improvements over the Viola-Jones framework. First, we introduce some variant of SURF features for fast and accurate object detection. Second, we propose AUC as the single criterion for cascade optimization. Third, we show a real example that can train cascade object detector from billions of samples within one hour on personal computers.

We compared SURF cascade detector with existing algorithms on detection accuracy and speed. Experiments show that SURF cascade can achieve results on par with state-of-the-art detectors, while beats tailored optimized OpenCV detector in detection speed.

Future work will consider three points: (1) other possible SURF variants to further improve detection accuracy; (2) applying the approach on other object detection task like human detection; (3) combing SURF cascade with deformable part based models.

References

[1] S. Agarwal, A. Awan, and D. Roth. Learning to detect objects in images via a sparse, part-based representation. *IEEE TPAMI*, 26(11):1475–1490, 2004. 7

[2] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool. Surf: Speeded up robust features. *CVIU*, 110:346–359, 2008. 1, 2, 3

[3] L. Bourdev and J. Brandt. Robust object detection via soft cascade. In *CVPR*, 2005. 1, 2, 6

[4] S. Brubaker, J.Wu, J. Sun, and et al. On the design of cascades of boosted ensembles for face detection. *IJCV*, 2008. 2, 3, 6

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 2, 3

[6] M. Everingham, L. V. Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 7

[7] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 2008. 4

[8] T. Fawcett. Roc graphs: Notes and practical considerations for researchers. *Machine Learning*, 2004. 2

[9] L. Fei-Fei, R. Fergus, and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. In *CVPR workshop*, 2004. 5

[10] P. Felzenszwalb, D.McAllester, and D. Ramanan. A discriminatively trained, multiscale, deformable part model. In *CVPR*, 2008. 2

[11] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *CVPR*, 2010. 2

[12] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of statistics*, 2000. 4

[13] W. Gao, B. Cao, S. Shan, X. Chen, D. Zhou, X. Zhang, and D. Zhao. The cas-peal large-scale chinese face database and baseline evaluations. *IEEE Trans. SMC-A*, 38:149–161, 2008. 6

[14] C. Huang, H. Ai, Y. Li, and S. Lao. Learning sparse features in granular space formulti-view face detection. In *AFGR*, 2006. 2

[15] V. Jain and E. Learned-Miller. Fddb: A benchmark for face detection in unconstrained settings. Technical Report UM-CS-2010-009, UMass, 2010. http://vis-www.cs.umass.edu/fddb/. 6

[16] V. Jain and E. Learned-Miller. Online domain adaptation of a pre-trained cascade of classifiers. In *CVPR*, 2011. 6

[17] N. Kumar, P. N. Belhumeur, and S. K. Nayar. Facetracer: A search engine for large collections of images with faces. In *ECCV*, 2008. 5

[18] I. Laptev. Improvements of object detection using boosted histograms. In *BMVC*, 2005. 3

[19] J. Li, T. Wang, and Y. Zhang. Face detection using surf cascade. In *ICCV workshop*, 2011. 3, 6

[20] S. Li, Z. Zhang, and et al. Floatboost learning for classification. In *NIPS*, 2002. 4

[21] R. Lienhart and J. Maydt. An extended set of haar-like features for rapid object detection. In *ICIP*, 2002. 2, 5, 6

[22] P. Long and R. Servedio. Boosting the area under the roc curve. In *NIPS*, 2007. 2

[23] S. Maji and A. C. Berg. Max-margin additive classifiers for detection. In *ICCV*, 2009. 1

[24] B. McCane and K. Novins. On training cascade face detectors. *Image and Vision Computing*, 2003. 2

[25] K. Mikolajczyk, C. Schmid, and A. Zisserman. Human detection based on a probabilistic assembly of robust part detectors. In *ECCV*, 2004. 2, 6

[26] T. Mita, T. Kaneko, and O. Hori. Joint haar-like features for face detection. In *ICCV*, 2005. 2

[27] M.-T. Pham and T.-J. Cham. Fast training and selection of haar features during statistics in boosting-based face detection. In *ICCV*, 2007. 2

[28] M.-T. Pham, Y. Gao, V. Hoang, and et al. Fast polygonal integration and its application in extending haar-like features to improve object detection. In *CVPR*, 2010. 1, 2, 6

[29] P. J. Phillips, H. Moon, P. J. Rauss, and S. Rizvi. The feret evaluation methodology for face recognition algorithms. *IEEE TPAMI*, 22:1090–1104, 2000. 5, 6

[30] T. Sim and S. Baker. The cmu pose, illumination, and expression (pie) database. In *FG*, 2002. 6

[31] J. Sochman and J. Matas. Waldboost - learning for time constrained sequential detection. In *CVPR*, 2005. 1, 2

[32] A. Sorokin and D. A. Forsyth. Utility data annotation with amazon mechanical turk. In *CVPR Workshop*, 2008. 1

[33] V. Subburaman and S. Marcel. Fast bounding box estimation based face detection. In *ECCV workshop*, 2010. 6

[34] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *ECCV*, 2006. 2

[35] http://mplab.ucsd.edu. The MPLab GENKI Database. 5

[36] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *CVPR*, 2001. 1, 2, 4, 6

[37] P. Viola and M. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004. 1, 3, 6

[38] S. Winder and M. Brown. Learning local image descriptors. In *CVPR*, 2007. 3

[39] J. Wu and et al. Fast asymmetric learning for cascade face detection. *IEEE PAMI*, 2008. 1, 3

[40] R. Xiao, H. Zhu, H. Sun, and X. Tang. Dynamic cascades for face detection. In *ICCV*, 2007. 1, 2, 4

[41] C. Zhang and Z. Zhang. A survey of recent advances in face detection. Technical Report MSR-TR-2010-66, Microsoft Research, 2010. 1

[42] Q. Zhu, S. Avidan, M.-C. Yeh, and K.-T. Cheng. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, 2006. 1, 2, 4