Boundary Detection Using F-Measure-, Filter- and Feature- (F^3) Boost

Iasonas Kokkinos

Department of Applied Mathematics, Ecole Centrale Paris
INRIA-Saclay, GALEN Group

Abstract. In this work we propose a boosting-based approach to boundary detection that advances the current state-of-the-art. To achieve this we introduce the following novel ideas: (a) we use a training criterion that approximates the F-measure of the classifier, instead of the exponential loss that is commonly used in boosting. We optimize this criterion using Anyboost. (b) We deal with the ambiguous information about orientation of the boundary in the annotation by treating it as a hidden variable, and train our classifier using Multiple-Instance Learning. (c) We adapt the Filterboost approach of [1] to leverage information from the whole training set to train our classifier, instead of using a fixed subset of points. (d) We extract discriminative features from appearance descriptors that are computed densely over the image. We demonstrate the performance of our approach on the Berkeley Segmentation Benchmark.

1 Introduction

The abundant biological evidence that our visual system employs sophisticated boundary detection mechanisms, and the legacy of D. Marr [2] has led early computer vision researchers to pursue computational approaches to boundary detection, considering it as the starting point for any subsequent processing. Moreover, the striking ease with which we recognize shape-based classes, e.g. sketches while being bereft of all appearance information also suggests that boundary detection may be the missing piece in the current, appearance-dominated, object recognition research.

A revival of research on boundary detection has been observed during the last years, largely due to the introduction of ground-truth labeled datasets [3,4] which facilitated the treatment of the problem in a machine learning framework, while weeding out many of the heuristics previously used in edge detection. Based on the consistent improvements observed during the last years on these benchmarks [5,6,7,8,9,10], boundary detection is anticipated to become an indispensable part of any computer vision 'toolbox'.

Our work proposes another step in the direction of accurate boundary detection, by pushing further the machine learning approach. In this work we reconsider the observation made in earlier works e.g. [4,9], where it was mentioned that using more elaborate machine learning techniques does not significantly improve performance. As we demonstrate here, while using the same cues as [8] we obtain better results based on a combination of techniques developed around boosting.

For this we build on the Anyboost framework [11] that views Boosting as gradient descent in function space. Based on this more general point of view, we first develop a

K. Daniilidis, P. Maragos, N. Paragios (Eds.): ECCV 2010, Part II, LNCS 6312, pp. 650–663, 2010.

new variant of boosting that optimizes an approximation to the F-measure of our classifier during training, instead of the exponential loss which is commonly minimized by Boosting. As boundary detectors are evaluated based on their F-measure, it is natural to expect that training also with the F-measure as a cost function will improve performance. We note that this contribution can be of broader interest, as the F-measure is employed in several other problems, such as retrieval, to deal with the case where the negative class largely outnumbers the positive one.

Second, we deal with ambiguity in the labelling of points by treating the orientation of the boundary as a hidden variable, and train our classifier using Multiple-Instance Learning [12].

Third, we leverage information from the whole dataset during training, instead of using a small set of points, as is commonly the case in other works. As the whole set of feature-label pairs cannot fit in memory, we use a stochastic gradient descent method, inspired from the recent Filterboost work [1]. At each round of boosting a subset of 'interesting points' is chosen and used to construct the weak learner for that round. This is done in a proper way in the setting of Anyboost, by forming a stochastic approximation to the functional gradient of the training cost with respect to the classifer. We can thus train complex classifiers without fear of overfitting, thanks to the huge number of available training samples ($\simeq$ 200 Images x 150000 Pixels).

A further improvement in performance is provided by discriminative information extracted from appearance descriptors. As in recent works [13] we compute descriptors densely on the image, thereby capturing the context of any given point, and provide this as an input to a classification algorithm. This provides additional information, that complements the local features used in the Berkeley edge detector.

Our contributions are experimentally evaluated on the Berkeley Segmentation Benchmark, demonstrating systematic improvements over the current state-of-the-art. As we intend to provide the source code for our work, we omit several implementation details; we focus on the major new ideas, leaving a more detailed presentation of the low-level processing for a longer version of this work.

2 Previous Work

After decades of edge detection research driven by insight and guesswork, a quantum jump has been the introduction of ground-truth labeled datasets [3,4] and the phrasing of edge detection as a pattern recognition task. The 'Berkeley edge detector' [4] was shown to outperform most edge detection approaches developed in the previous decades, by replacing intuitively developed measures, such as the strength of directional derivatives [14,15], with statistical measures of texture, color and intensity discontinuity, and leaving their combination to machine learning.

This approach has led to improved detectors based on Boosting [6], topological properties of the image [7], spectral gradients [8], multiscale processing [9] and sparse dictionaries [10], among others. The most powerful boundary detectors currently in use [8,16,17] rely on combining different cues for boundary detection, such as texture gradients, brightness/color gradients, or information extracted from spectral clustering. Each of these cues is indicative of the presence of an edge, and the task of learning their

optimal linear combination is typically accomplished with logistic regression. A point mentioned repeatedly in several works, e.g. [9,4] is that using more intricate machine learning tools does not substantially improve performance. However both [6] and our work indicate that substantial improvements can be obtained by using more sophisticated learning approaches, as we will now present.

3 Learning Boundary Detection

We start with a presentation of boosting, where we introduce notation and the Anyboost technique [11] which is central to the rest of the paper.

Our training data come as sets of input-output pairs, $(X_i, y_i), X_i \in \mathcal{X}, y_i \in \mathcal{Y}, i = 1, \ldots, N$, where N is the size of our training set, typically $\mathcal{X} = R^d$ and for classification $Y = \{-1, 1\}$. Boosting algorithms learn a mapping from the input to the output space using a linear combination of simpler functions ('weak learners'):

$$f_T(X) = \sum_{t=1}^{T} a_t h_t(X). \tag{1}$$

The weak learners h_t are members of a family of simple functions, $\mathcal{H}$ but their combination in f_T can result in a complex classifier ('strong learner').

Boosting algorithms construct f in a sequential manner, by introducing at each iteration t a new component $h_t \in \mathcal{H}$ and a corresponding coefficient a_t that will most quickly improve the performance of the classifier. Performance is quantified by a cost $C(f)$ for the discrepancy between the classifier's predictions and the labels of the training set. This is typically a sum of individual costs over the training set, i.e.:

$$C(f) = \sum_{i=1}^{N} c(f(X_i), y_i) \tag{2}$$

For instance $c(f(X_i), y_i) = \exp(-y_i f(X_i))$ gives the exponential loss used in Adaboost training [18], while logistic regression scores have been considered in [19]. Moreover, different algorithms have been proposed to perform the selection steps for the weak learner and stepsize including Discrete-Adaboost [18], Gentleboost [20], or Confidence-Rated Boosting among others.

A unifying theme for these algorithms has been provided by the Anyboost algorithm [11], that views boosting as gradient descent in function space; namely each round of boosting can be seen as moving the function f in the direction that most rapidly decreases $C(f)$. In specific, consider that the outputs of the classifier f_t at iteration t on the training set are combined in a vector $\mathbf{f}$, s.t. $\mathbf{f}_i = f_t(X_i), i = 1, \ldots, N$. The negative gradient of the cost function with respect to the classifier's responses:

$$\mathbf{g}_i = -\frac{\partial C}{\partial \mathbf{f}_i}, \tag{3}$$

provides the update direction for f on the training set that will most rapidly decrease the cost being optimized. As we can only change our classifier by adding a member

of the family $\mathcal{H}$, boosting resorts to finding that function $h^* \in \mathcal{H}$ that is closest to the direction pointed by $\mathbf{g}$, i.e. has maximum inner product with $\mathbf{g}$:

$$h^* = \text{argmax}_h < \mathbf{g}, h >= \text{argmin}_h l(h), \quad l(h) = \sum_i \frac{\partial C}{\partial \mathbf{f}_i} h(X_i). \tag{4}$$

At each round we thus train a classifier using a reweighted version of the training set; each sample has a weight $|g_i|$, while the sign of g_i determines whether the weak learner should have a positive response. At each round a weak learner $h_t \in \mathcal{H}$ is chosen so as to minimize Eq. 4.

Once the h_t is chosen, its coefficient a_t is determined with line search to minimize $C(f_{t-1} + a_t h_t)$. The Anyboost algorithm is thus summarized as follows:

$f_T = \text{ANYBOOST}(C, \{X_i, Y_i\}, i = 1 \dots N, T)$
Set $\mathbf{f}_i = 0, \forall i$
for $t = 1$ to T **do**
(a) Compute negative Gradient of C at $\mathbf{f}$: $\mathbf{g}_i = -\frac{\partial C}{\partial \mathbf{f}_i}$.
(b) Find the weak learner h_t which minimizes: $l(h) = \sum_i -\mathbf{g}_i h(X_i)$
(c) Choose the step size a_t that minimizes $C(f_{t-1} + a_t h_t)$ using line search.
(d) Set $\mathbf{f}_i = f_t(X_i)$.
end for
Output $f_T(x) = \sum_{t=1}^{T} a_t h_t(x)$.

We can now proceed with the presentation of our contributions in using Boosting for boundary detection. These are in the following directions:

- Using a cost C that properly measures the performance of our boundary detector system, by approximating its F-measure, Sec. 3.1.
- Dealing with ambiguity in labeling using Multiple-Instance Learning in conjunction with Anyboost, Sec. 3.2.
- Exploiting the whole Berkeley training set by forming a stochastic approximation to the weak-learner training criterion, l, Sec. 3.3 and the cost C used during line-search, Sec. 3.3.

3.1 F-Measure Boosting

Most training criteria in Boosting are defined as summations of a sample-based cost function over the whole training set, as in Eq. 2. This is the case for instance in the exponential loss or the log-likelihood score. However, such criteria can lead to poor classifiers when the training sets are imbalanced, which is the case for boundary detection: there are two orders of magnitude less boundary points than non-boundary points, so a classifier that errs in favor of non-boundary decisions can have a lower score than a more balanced one, when using a summation-based cost.

This is reflected in the F-measure that is used to score boundary detectors, defined as the geometric mean of the classifier's *precision*, p and *recall*, r:

$$F = \frac{2pr}{p+r}, \quad \text{where} \quad p = \frac{TP}{TP+FA}, \quad r = \frac{TP}{TP+MS} \tag{5}$$

In Eq. 5 TP is the number of the true positives, MS the number of misses, and FA the number of false alarms. Precision gives us the proportion of correct detector responses, while recall indicates the proportion of the true boundaries that have been detected. Note that the false negatives do not appear anywhere; therefore the classifier does not get credit for rejecting negatives, but only pays for false alarms. This allows to deal with a large negative class during evaluation (and training).

Even though the F-measure is broadly used as an evaluation measure, it is not commonly used in training as it is harder to optimize. However, algorithms for optimizing it have been developed in the context of SVMs [21] and logistic regression [22], while the authors in [8] mention optimizing the F-measure to training their logistic regression-based classifier. Here we use the Anyboost framework to apply the ideas developed in [22] to classifiers trained with boosting.

Following [22], we express the TP, MS, FA terms as sums over the training set:

$$TP=\sum_{i=1}^{N}[\hat{y}_i=1][y_i=1],\ FA=\sum_{i=1}^{N}[\hat{y}_i=1][y_i=-1],\ MS=\sum_{i=1}^{N}[\hat{y}_i=-1][y_i=1] \quad (6)$$

where $\hat{y}_i$ is the label estimated by the classifier (e.g. by thresholding $f(X_i)$ at 0), y_i is the correct label and $[\cdot]$ is indicating the truth of $\cdot$.

We then replace the quantities in Eq. 6 with probabilistic approximations, by replacing $[\hat{y}_i = 1], [\hat{y}_i = 0]$ with the soft measures $P(y = 1|f(X_i)), P(y = -1|f(X_i))$ respectively. For instance, we approximate the number of false alarms by $FA \simeq \hat{FA} = \sum_{i=1}^{N} P(y_i = 1|f(X_i))[y_i = -1]$. We combine the estimates of precision $\hat{p} = \frac{\hat{TP}}{\hat{TP}+\hat{FA}}$ and recall $\hat{r} = \frac{\hat{TP}}{\hat{TP}+\hat{MS}}$ in an approximation to the classifier's F-measure:

$$\hat{F} = \frac{2\hat{p}\hat{r}}{\hat{p}+\hat{r}} = \frac{\hat{TP}}{\hat{TP} + (\hat{FA} + \hat{MS})/2} \quad (7)$$

We thereby replace the terms showing up in the original F-measure by differentiable quantities, that smoothly vary as we change the classifier's output. This allows us to perform gradient descent so as to maximize the approximate F-measure. We now present how to optimize this measure with a classifier trained with Anyboost, using as cost $C(f) = 1 - \hat{F}$.

For now, we consider turning the output of a boosting-based classifier into a soft estimate by setting $P(y = l|f(X)) = \sigma_l(f(X)) = \frac{1}{1+\exp(-lf(X))}$, where l indicates the label of the training point. In Sec. 3.2 we will present a more elaborate expression, that can be directly incorporated in what we now present.

To apply the Anyboost algorithm, we need to measure how changing the response of the classifier at point i will affect the classifier's F-measure. This is given by [22]:

$$\mathbf{g}_i = \frac{\partial C}{\partial \mathbf{f}_i^t} = \left[H[y_i = 1] - \frac{H^2}{2}\hat{TP}\right]\sigma'_{y_i}(\mathbf{f}_i^t),\ \ H = \left(\sum_{i=1}^{N}[y_i = 1] + \frac{1}{2}(\hat{TP} + \hat{FA})\right)^{-1}, \quad (8)$$

while for a sigmoidal σ we have $\sigma'_{y_i}(\mathbf{f}_i) = \frac{d\sigma_{y_i}(\mathbf{f}_i)}{\mathbf{f}_i} = \sigma_{y_i}(\mathbf{f}_i)(1 - \sigma_{y_i}(\mathbf{f}_i))$.

The expression in Eq. 8 determines the weighting of point i for round t, using the classifier f_{t-1} from the previous round: The vector $-\mathbf{g}$ indicates how the classifier's outputs should change so as to most rapidly increase the F-measure; and as already mentioned, with Anyboost we choose the $h_t \in \mathcal{H}$ that is closest to this direction, by maximizing $\sum_i -\mathbf{g}_i h_i(X_i)$.

We note that the cost function is *not* defined as a sum of individual costs, but rather is combining nonlinearly two global measures, the classifier's precision and recall. But at each round we compute the partial derivative of the cost w.r.t. the classifier's output on the individual points, which is forming a local linear approximation to the cost; this is then used to drive the fitting of the weak learner. Of course, the optimization cost is no longer convex so we may end up in local minima of the cost; nevertheless in our results we observed that the performance of the classifier trained with this criterion is better than that of the one trained with the convex criterion of standard Adaboost.

3.2 Multiple Instance Learning with Noisy-OR

So far we have been considering that we are provided with feature-label pairs. But our classifiers use orientation-dependent features, and classify each point based on an assumed orientation, say j; a point is labeled as positive if the classifier fires along *any* orientation. Using manual annotations to determine the orientation is tricky, since different users may suggest different orientations for the same image location depending on the granularity of their segmentation (e.g. on texture boundaries). Moreover, for points such as corners, or junctions, orientation is not properly defined.

To deal with this we use the adaptation of Multiple Instance Learning to Boosting by [12], and train a classifier in a way that copes with the missing orientation information. In specific, we extract features X for our classifier at all N orientations (we use $N = 8$), obtaining a 'bag' of features $\mathcal{X}_i = \{X_{i,1}, \ldots, X_{i,N}\}$ at each point i. For each orientation our classifier provides us with a probability estimate $P(y_i = 1|X_{i,j}) = 1/(1 + \exp(-f(X_{i,j})))$. The final decision is taken by a Noisy-OR combination:

$$p_i = P(y_i = 1|\mathcal{X}_i) = 1 - \prod_{j=1}^{N} (1 - P(y_i = 1|X_{i,j})). \tag{9}$$

This has a similar behavior with the a maximum-based combination - the left hand side is large when any of $P(y_i = 1|\phi_j)$ is large, and small only when all of them are small - but is differentiable. We use p_i as a shorthand for the result of the noisy-or combination rule.

This allows us to train this classifier using gradient descent and in specific, with Anyboost. We now refine our earlier presentation: the probabilistic estimates used in the approximation of the F-measure in Sec. 3 correspond to the left-hand side of Eq. 9, namely the combined decision about the point after considering all orientations. However, the classifier that we train shows up in the the right hand side, and gives the probability of point i being an edge using the features $X_{i,j}$ computed for orientation j: $P(y = 1|X_{i,j}) = (1 + \exp(-f(X_{i,j})))^{-1}$.

Using Anyboost we consider the classifier responses $f_{i,j}$ on all possible orientations, and stack the derivative of the cost with respect to them in a vector $\mathbf{n}$. With a slight abuse of notation we use two indexes for the elements of this vector and denote its elements by $\mathbf{n}_{i,j} = \frac{\partial C}{\partial f_{i,j}}$. The partial derivatives can be computed using the chain law:

$$\frac{\partial C}{\partial \mathbf{f}_{i,j}} = \frac{\partial C}{\partial \mathbf{p}_i} \frac{\partial \mathbf{p}_i}{\partial \mathbf{g}_{i,j}} = \left[H[y_i = 1] - \frac{H^2}{2} \hat{TP} \right] (1 - p_i) p_{i,j} y_i. \tag{10}$$

The bracket on the left comes from Eq. 8 while the right hand side can be derived using the property of the sigmoidal $\sigma' = \sigma(1 - \sigma)$.

The weak learner is thus trained by maximizing $\sum_{i,j} n_{i,j} h(X_{i,j})$. An inspection of Eq. 10 reveals that this assigns higher weights to the orientations which give higher responses, and can thus drive more quickly the change in the cost function.

3.3 Filtering via Stochastic Gradient Descent

Up to now we have considered that at each round a weak learner is trained by optimizing a quantity obtained by summing over the whole training set as dictated by the Anyboost algorithm; e.g. for noisy-or we considered minimizing $\sum_{i,j} n_{i,j} h(X_{i,j})$ where i ranges over all pixels in all images and j ranges over 8 possible orientations.

In practice this can be infeasible, due to both time and memory constraints. It is therefore common practice to pick at random a subset of the training set initially and then use it throughout training. This can lead however to overfitting, in particular if a small training set is given and a complex classifier is trained, while an unfortunate choice of a subset for training can also result in poor performance. We would like instead to maintain the whole training set throughout training, and use a proper portion of it at each round.

For this we propose a solution inspired from the recent Filterboost work [1], that adapts Boosting to the filtering problem; the filtering problem amounts to iteratively training a classifier with a subset of the training set at a time, while guaranteeing its good performance over the whole training set.

The adaptation of this idea is straightforward, once the Anyboost interpretation of Boosting is developed: we replace the criterion $l(h) = \sum_i \mathbf{g}_i h(X_i)$ used in Adaboost with a stochastic approximation, $\hat{l}(h)$ obtained by using a subset of the training data. In specific, we first normalize $\mathbf{g}$ so that $\sum_i |\mathbf{g}_i| = 1$. This does not affect the choice of h. We then construct a distribution $p_{\mathbf{g}}(i) = |\mathbf{g}_i|$ on the training set and interpret $l(h)$ as the expectation of $\mathrm{sgn}(\mathbf{g}_i) h(X_i)$ with respect to this distribution: $l(h) = E_{p_{\mathbf{g}}}(\mathrm{sgn}(\mathbf{g}) h(X))$.

We can then form a Monte Carlo approximation to this expectation, by drawing samples from the training set according to $p_{\mathbf{g}}$, and averaging the value of $\mathrm{sgn}(\mathbf{g}_i) H(X_i)$ on those samples:

$$l(h) = E_{p_{\mathbf{g}}}(\mathrm{sgn}(\mathbf{g}) h(X)) \simeq \frac{1}{K} \sum_{k=1}^{K} \mathrm{sgn}(\mathbf{g}_k) h(X_k) \equiv \hat{l}(h) \tag{11}$$

where $X_k, k = 1 \dots K$ are samples drawn from p_g. We thus replace the original problem of optimizing $l(h)$ by the optimization of its approximation $\hat{l}(h)$, formed using K instead of N samples. In practice, while our training set contains $N \simeq 3 \cdot 10^7$ points, we use $K = 5 \cdot 10^5$ samples at each iteration.

Using this scheme the points are chosen adaptively at each iteration: they are drawn from p_g, which is quantifying their usefulness for decreasing the cost *at the current round*. This allows our training algorithm to make the best use of the training data, and focus on the harder ones from the whole training set. Contrary, if one works with a fixed subset of the training set throughout, boosting fine tunes the performance of the classifier over a small set of points which leads to diminishing returns, or even overfitting for larger rounds of boosting.

We note that the proposed technique may seem similar to the stochastic Boosting method of [23]; however in our case the choice of points is driven by the cost gradient at the current round, instead of being a random sampling of the training set. Moreover, the same approach can be used to optimize any other training cost, and is not constrained to the F-measure used here. We therefore believe it can prove useful for a broader range of problems apart from feature detection.

Step size selection. The scheme described above allows us to find the approximately optimal weak learner h_t at round t; a similar scheme can be used to estimate the optimal step size a_t using a stochastic approximation to the cost function. As is known from Monte Carlo integration, forming a good stochastic approximation of a quantity requires sampling it more densely where it is larger in magnitude. We therefore use a different set of samples to estimate the step-size, by sampling more densely points that contribute to the $\hat{TP}, \hat{FA}, \hat{MS}$ quantities used to estimate the F-measure. For this, we form the function $d_i = y_i + p_i$ that adds the two quantities which indicate whether the response p_i at point i, can affect the F-measure: being a true positive/false negative, in which case $y_i = 1$ or being a false positive, in which case p_i is large. We normalize d_i so that it sums to one, and we see it as a distribution on lthe training set, denoted by p_d.

We then express $\hat{TP}, \hat{FA}, \hat{MS}$ as expectations with respect to this distribution, and form Monte Carlo approximations to these; for instance for $\hat{TP}$ we have:

$$\hat{TP} = \sum_{i=1}^{N} p(y=1|\mathcal{X}_i) = \sum_{i=1}^{N} d_i \frac{p(y=1|\mathcal{X}_i)}{d_{i,j}} = E_{p_d}\left(\frac{p(y=1|\mathcal{X}_i)}{d_{i,j}}\right), \quad (12)$$

so $\hat{TP} \simeq \sum_{k=1}^{K} p(y=1|\mathcal{X}_k)/d_k$ where X_k are samples drawn from p_d. A sample here amounts to the whole bag $\mathcal{X} = \{X_{i,1}, \dots, X_{i,N}\}$ at point i.

Summarizing, the optimal step a_t is found at each round t using line search, based on the stochastic approximation to $C(f_{t-1} + a_t h_t)$. When estimating the value of C for a candidate step a_t we perform the following steps:

- Compute the classifier's response $f(X_{i,j}) = f_{t-1}(X_{i,j}) + a_t h_t(X_{i,j})$ for all samples $i = 1 \dots K$ and all instances $j = 1, \dots 8$, within each bag.
- For each sample i, combine these responses using the noisy-or combination rule, to provide $p_i = p(y=1|\mathcal{X}_i)$.

– Form the Monte-Carlo approximations to $\hat{TP}, \hat{MS}, \hat{FA}$ as in Eq. 12 and combine them to provide an estimate $\hat{f}$.

4 Discriminative Features from Descriptors

In our earlier work [24] we have observed that a substantial improvement in performance can be gained by extracting discriminative information from descriptors. A similar result was preliminarily observed in [25] for the figure-ground assignment task, where geometric blur descriptors were used to leverage mid-level information.

Here we develop this idea further, and demonstrate the gain obtained by integrating descriptors with the other cues used during boosting. Specifically, in [24] we extract SIFT descriptors at multiple scales around candidate edgels to form a high-dimensional feature vector describing the context in which each edgel appears. We extend this idea by *densely* computing descriptors over the image; we can thus use their low-dimensional projections as regular features during both training and testing. Instead, in our earlier work we needed to do boundary detection and non-maximum suppression at the very beginning to extract descriptors around candidate edges. We thus replace our original two-tiered detection with an integrated version. More importantly, we experimentally demonstrate that even when using a highly optimized detector, using descriptor information yields an additional gain in performance.

We use a log-polar sampling scheme as in [13,26], using a sampling grid with 5 scales and 12 angles. Compared to SIFT, the log-polar sampling allows us to re-use the same descriptor for multiple orientations, simply by permuting its indexes, while also taking into account the context from a larger part of the image.

As in the Daisy descriptor [13], at each sampling point we compute derivative-of-Gaussians along eight orientations; the scales of the Gaussians are set proportional to scale of the point, and we compute such descriptors for all three channels of the Lab space. We also use Gabor filters for the L-component to capture texture information. Both Gabor and Gaussian filters are implemented using recursive (IIR) filtering to speedup descriptor computation. In all, we have 4 channels (3 for Lab and 1 for Gabor-texture), with 6 scales, 12 radii, and 6 orientations each, giving us a high-dimensional descriptor of the context around a point.

As in [24] we use a pre-processing step that discriminatively compresses descriptors into a low-dimensional space, and then use the coordinates in this space as inputs to our classifier. In specific, we use the Spliced Average Variance Estimation technique of [27] to find such a projection; this provides us with a set of orthogonal projection directions that can be easily computed in test-time.

In Fig. 1 we visualize the first two projection directions for descriptors extracted around a presumably horizontal edge. We show two different projections (vertical direction) for three cues (horizontal direction). Each projection is computed by summing the products of the descriptor values with the corresponding projection elements. Each needle shows the matrix entry for the corresponding location and orientation of the descriptor: red/blue denotes sign while the length indicates magnitude. Even though not as easily interpretable as the projections we would obtain from PCA, we observe that the projection dimensions correspond to geometrically meaningful patterns.

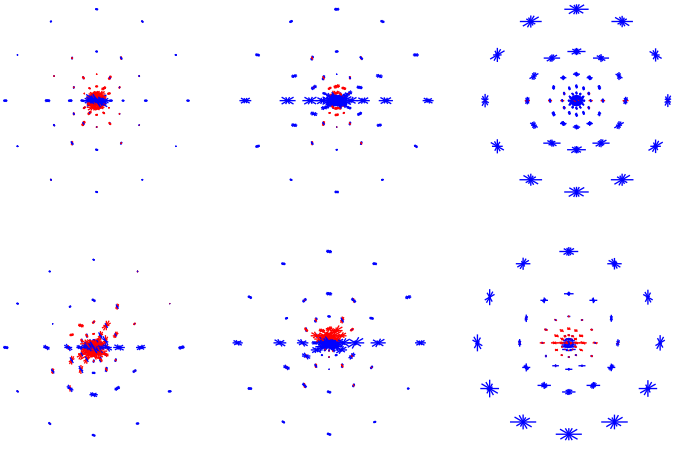

Fig. 1. Discriminative projections computed for (left) intensity (middle) color and (right) texture descriptors. The color of the needles indicates the sign, and their length indicates the magnitude of the projection coefficient for the corresponding descriptor dimension.

5 Application to Boundary Detection

We now focus on the problem of boundary detection. We first describe an adaptation of F-measure boosting that proved beneficial in tuning our detector, and then provide experimental results.

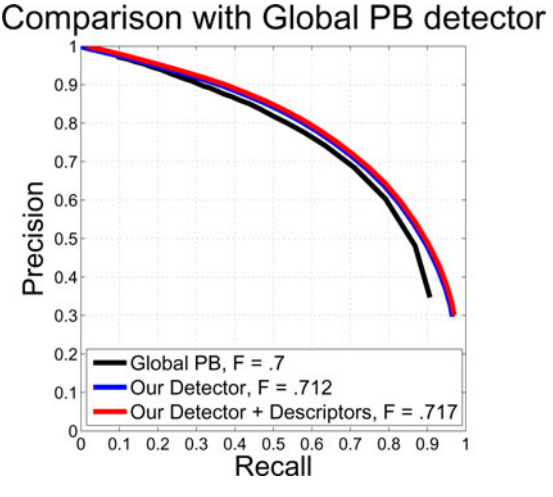

Fig. 2. Benchmarking results. Our method achieves an F-measure of 0.712, while together with descriptors the performance increases to 0.717. This compares favorably to the global-Pb detector, whose reported F-measure is .70. Please see text for details.

Fig. 3. Sample results from the Berkeley benchmark: the ground truth is shown on the middle and on the right we show our detector's estimate for the probability of a boundary

5.1 Calibrating the F-Measure for Boundary Detection

So far we have considered training a classifier with F-measure boosting in a general setting. For the boundary detection task in specific, we have realized that the reported F-measure in the evaluations is affected by two additional factors: First, images in the Berkeley benchmark are labeled by multiple persons, so certain points receive a 'boundary' label multiple times. We therefore take into account the number of times N_i that each training point i was labelled as positive in the expressions for TP and TM:

$$TP = \sum_{i=1}^{N} N_i[\hat{y}_i = 1] \simeq N_i P(y_i = 1|f(X_i)) = \hat{TP} \tag{13}$$

$$MS = \sum_{i=1}^{N} N_i[\hat{y}_i = -1] \simeq \sum_{i=1}^{N} N_i P(y_i = -1|f(X_i)) = \hat{MS} \tag{14}$$

The expressions for TP and MS are the ones that are computed during the evaluation, according to the code of [4], while the expressions for $\hat{TP}$ and $\hat{MS}$ are used for training. These expressions emphasize points that are consistently labeled by more users as boundaries. This improves the F-measure of the detector on both the training and test sets.

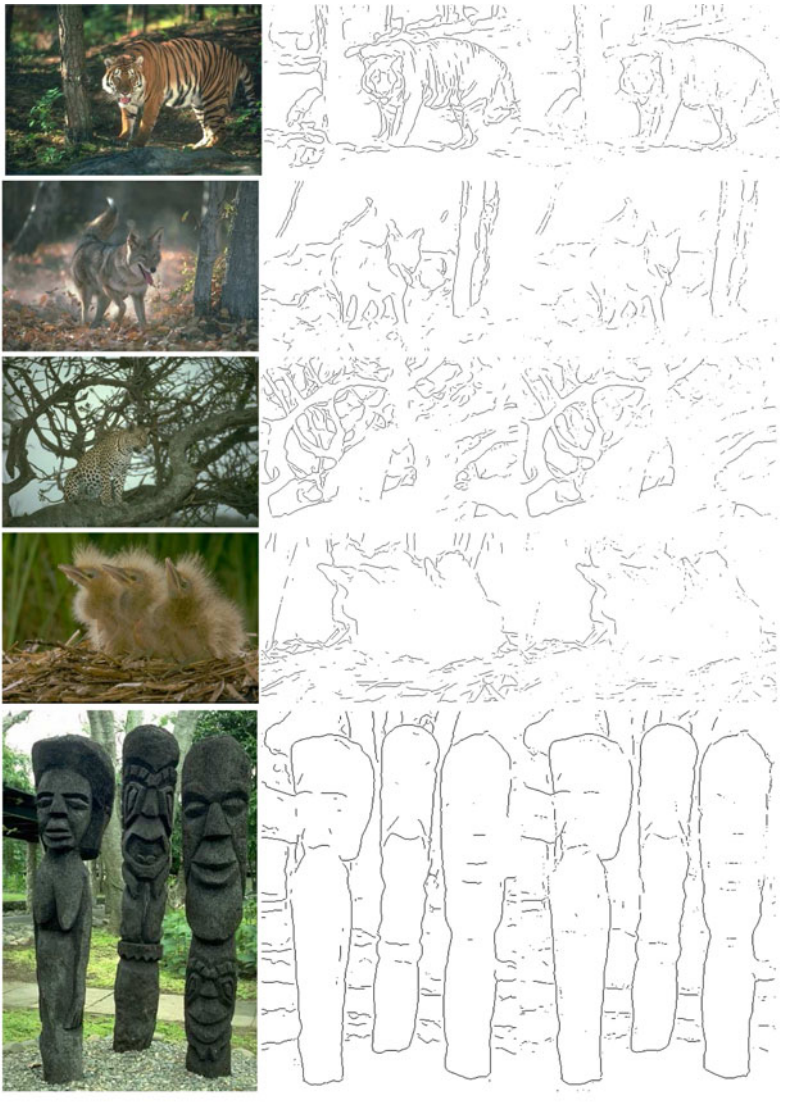

Fig. 4. Comparisons of the Global PB detector (left) with our results (right). Both detectors are thresholded at the value giving the best global F-measure. Overall, we observer that our detector responds less to textured, or cluttered image areas.

Second, the boundaries used for evaluating the detector are obtained after non-maximum suppression. Only a fraction of false positives will thus survive suppression, and give a false alarm. Scaling the estimate of $\hat{FA}$ in Eq. 14 by a fraction of $1/10$ therefore yields an estimate of false alarms that is much closer to the one reported by the evaluation software.

5.2 Experimental Results

In order to systematically evaluate our approach we have conducted systematic experiments on the Berkeley Benchmark.

As mentioned in the introduction, our contributions are in both the learning, and the feature extraction direction. To validate our contribution in learning, we first train a classifier using exactly the same features as in [8], namely multi-scale color and texture gradients, as well as 'spectral gradients', obtained from the directional derivatives of the eigenvectors found from normalized cuts. The difference is in the learning algorithm (Adaboost) and the fact that we use the whole training set for training. Both we and [8] use the F-measure for training, so we are optimizing essentially the same cost. In [17] a combination of the gPb detector with a segmentation algorithm results in a improvement of the gPb detector's F-measure from .7 to .71. Our detector achieves an F-measure of .712 while not using additional information from segmentation. The performance of the classifier trained using our earlier setup [24] of using a sparse set of training data, with fixed orientation decreases the performance to $F = .7$. This demonstrates the merit of using MIL for orientations and Filterboost.

To validate our contribution in feature extraction we perform the training procedure, but now introducing the new features obtained from the appearance descriptors by discriminative dimensionality reduction. It becomes clear that the descriptors provide an additional boost in performance, which increases to .717.

Regarding testing time, extracting the features of [8] takes 80s on a 3Gh machine, while [16] cut it down to 1s on a GPU. Computing dense descriptors requires 50s in Matlab, but is also easily parallelizable on GPUs. Once features are extracted, evaluating our detector takes 20s in Matlab.

6 Conclusions

In this work we have pushed further the machine learning approach to one of the most basic problems in computer vision, boundary detection. We have obtained state-of-the-art results in the setting of boosting by (i) using a proper training criterion, based on the F-measure (b) exploiting the whole training set during training and (iii) introducing new discriminative features using context information captured by descriptors. In future work we intend to extend the application of these ideas to the detection of other types of low-level features, while also pursuing the exploitation of these better boundaries for object recognition.

References

1. Bradley, J.K., Schapire, R.E.: Filterboost: Regression and classification on large datasets. In: NIPS (2007)
2. Marr, D.: Vision. W.H. Freeman, New York (1982)

3. Konishi, S., Yuille, A.L., Coughlan, J.M., Zhu, S.C.: Statistical Edge Detection: Learning and Evaluating Edge Cues. PAMI 25 (2003)
4. Martin, D., Fowlkes, C., Malik, J.: Learning to Detect Natural Image Boundaries Using Local Brightness, Color, and Texture Cues. PAMI 26, 530–549 (2004)
5. Ren, X., Fowlkes, C., Malik, J.: Scale-invariant contour completion using crfs. In: ICCV (2005)
6. Dollar, P., Tu, Z., Belongie, S.: Supervised Learning of Edges and Object Boundaries. In: CVPR (2006)
7. Arbelaez, P.: Boundary Extraction in Natural Images Using Ultrametric Contour Maps. In: WPOCV (2006)
8. Maire, M., Arbelaez, P., Fowlkes, C., Malik., J.: Using Contours to Detect and Localize Junctions in Natural Images. In: CVPR (2008)
9. Ren, X.: Multiscale helps boundary detection. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 533–545. Springer, Heidelberg (2008)
10. Mairal, J., Leordeanu, M., Bach, F., Hebert, M., Ponce, J.: Discriminative sparse image models for class-specific edge detection and image interpretation. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 43–56. Springer, Heidelberg (2008)
11. Mason, L., Baxter, J., Bartlett, P., Frean, M.: Boosting algorithms as gradient descent. In: NIPS (2000)
12. Viola, P., Platt, J.C., Zhang, C.: Multiple Instance Boosting and Object Detection. In: NIPS (2006)
13. Tola, E., Lepetit, V., Fua, P.: A fast local descriptor for dense matching. In: CVPR (2008)
14. Canny, J.: A Computational Approach to Edge Detection. PAMI 8, 679–698 (1986)
15. Perona, P., Malik, J.: Detecting and Localizing Edges Composed of Steps, Peaks and Roofs. In: ICCV, pp. 52–57 (1990)
16. Catanzaro, B., Sundaram, N., Su, B., Lee, Y., Murphy, M., Keutzer, K.: Efficient high-quality image contour detection. In: ICCV (2009)
17. Arbelaez, P., Maire, M., Fowlkes, C., Malik., J.: From contours to regions: An empirical evaluation. In: CVPR (2009)
18. Freund, Y., Schapire, R.: Experiments with a new Boosting Algorithm. In: ICML (1996)
19. Collins, M., Schapire, R.E., Singer, Y.: Logistic regression, adaboost and bregman distances. In: Machine Learning (2000)
20. Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Ann. Stat. (2000)
21. Joachims, T.: A support vector method for multivariate performance measures. In: ICML (2005)
22. Jansche, M.: Maximum expected f-measure training of logistic regression models. In: HLT 2005 (2005)
23. Friedman, J.H.: Stochastic gradient boosting. Comput. Stat. Data Anal. (2002)
24. Kokkinos, I.: Highly accurate boundary detection and grouping. In: CVPR (2010)
25. Ren, X., Fowlkes, C.C., Malik, J.: Figure/ground assignment in natural images. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 614–627. Springer, Heidelberg (2006)
26. Kokkinos, I., Yuille, A.: Scale Invariance without Scale Selection. In: CVPR (2008)
27. Cook, D., Lee, H.: Dimension reduction in binary response regression. JASA 94 (1999)