Crisp Boundary Detection Using Pointwise Mutual Information

Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson

Massachusetts Institute of Technology, USA
{phillipi,danielz,dilipkay,eadelson}@mit.edu

Abstract. Detecting boundaries between semantically meaningful objects in visual scenes is an important component of many vision algorithms. In this paper, we propose a novel method for detecting such boundaries based on a simple underlying principle: pixels belonging to the same object exhibit higher statistical dependencies than pixels belonging to different objects. We show how to derive an affinity measure based on this principle using pointwise mutual information, and we show that this measure is indeed a good predictor of whether or not two pixels reside on the same object. Using this affinity with spectral clustering, we can find object boundaries in the image – achieving state-of-the-art results on the BSDS500 dataset. Our method produces pixel-level accurate boundaries while requiring minimal feature engineering.

Keywords: Edge/Contour Detection, Segmentation.

1 Introduction

Semantically meaningful contour extraction has long been a central goal of computer vision. Such contours mark the boundary between physically separate objects and provide important cues for low- and high-level understanding of scene content. Object boundary cues have been used to aid in segmentation [1,2,3], object detection and recognition [4,5], and recovery of intrinsic scene properties such as shape, reflectance, and illumination [6]. While there is no exact definition of the "objectness" of entities in a scene, datasets such as the BSDS500 segmentation dataset [1] provide a number of examples of human drawn contours, which serve as a good objective guide for the development of boundary detection algorithms. In light of the ill-posed nature of this problem, many different approaches to boundary detection have been developed [1,7,8,9].

As a motivation for our approach, first consider the photo on the left in Figure 1. In this image, the coral in the foreground exhibits a repeating pattern of white and gray stripes. We would like to group this entire pattern as part of a single object. One way to do so is to notice that white-next-to-gray co-occurs suspiciously often. If these colors were part of distinct objects, it would be quite unlikely to see them appear right next to each other so often. On the other hand, examine the blue coral in the background. Here, the coral's color is similar to the color of the water behind the coral. While the change in color is subtle along

D. Fleet et al. (Eds.): ECCV 2014, Part III, LNCS 8691, pp. 799–814, 2014.

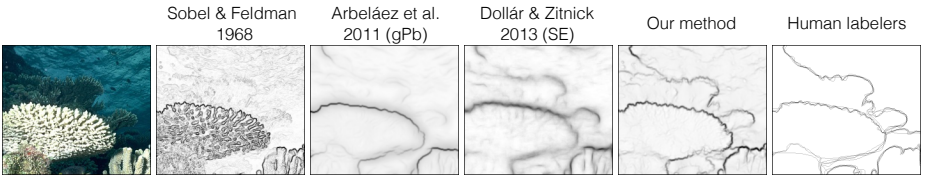

Fig. 1. Our method suppresses edges in highly textured regions such as the coral in the foreground. Here, white and gray pixels repeatedly occur next to each other. This pattern shows up as a suspicious coincidence in the image's statistics, and our method infers that these colors must therefore be part of the same object. Conversely, pixel pairs that straddle the coral/background edges are relatively rare and our model assigns these pairs low affinity. From left to right: Input image; Contours recovered by the Sobel operator [10]; Contours recovered by Dollár & Zitnick 2013 [8]; Contours recovered by Arbeláez et al. (gPb) [1]; Our recovered contours; Contours labeled by humans [1]. Sobel boundaries are crisp but poorly match human drawn contours. More recent detectors are more accurate but blurry. Our method recovers boundaries that are both crisp and accurate.

this border, it is in fact a rather unusual sort of change – it only occurs on the narrow border where coral pixels abut with background water pixels. Pixel pairs that straddle an object border tend to have a rare combination of colors.

These observations motivate the basic assumption underlying our method, which is that the statistical association between pixels *within* objects is high, whereas for pixels residing on different objects the statistical association is low. We will use this property to detect boundaries in natural images.

One of the challenges in accurate boundary detection is the seemingly inherent contradiction between the "correctness" of an edge (distinguishing between boundary and non-boundary edges) and "crispness" of the boundary (precisely localizing the boundary). The leading boundary detectors tend to use relatively large neighborhoods when building their features, even the most local ones. This results in edges which, correct as they may be, are inherently blurry. Because our method works on surprisingly simple features (namely pixel color values and very local variance information) we can achieve both accurate *and* crisp contours. Figure 1 shows this appealing properties of contours extracted using our method. The contours we get are highly detailed (as along the top of the coral in the foreground) and at the same time we are able to learn the local statistical regularities and suppress textural regions (such as the interior of the coral).

It may appear that there is a chicken and egg problem. To gather statistics within objects, we need to already have the object segmentation. This problem can be bypassed, however. We find that natural objects produce probability density functions (PDFs) that are well clustered. We can discover those clusters, and fit them by kernel density estimation, without explicitly identifying objects. This lets us distinguish common pixel pairs (arising within objects) from rare ones (arising at boundaries).

In this paper, we only look at highly localized features – pixel colors and color variance in 3x3 windows. It is clear, then, that we cannot derive long feature vectors with sophisticated spatial and chromatic computations. How can we hope to get good performance? It turns out that there is much more information in the PDFs than one might at first imagine. By exploiting this information we can succeed.

Our main contribution is a simple, principled and unsupervised approach to contour detection. Our algorithm is competitive with other, heavily engineered methods. Unlike these previous methods, we use extremely local features, mostly at the pixel level, which allow us to find crisp and highly localized edges, thus outperforming other methods significantly when more exact edge localization is required. Finally, our method is unsupervised and is able to adapt to each given image independently. The resulting algorithm achieves state-of-the-art results on the BSDS500 segmentation dataset.

The rest of this paper is organized as follows: we start by presenting related work, followed by a detailed description of our model. We then proceed to model validation, showing that the assumptions we make truly hold for natural images and ground truth contours. Then, we compare our method to current state-of-the-art boundary detection methods. Finally, we will discuss the implications and future directions for this work.

2 Related Work

Contour/boundary detection and edge detection are classical problems in computer vision, and there is an immense literature on these topics. It is out of scope for this paper to give a full survey on the topic, so only a small relevant subset of works will be reviewed here.

The early approaches to contour detection relied on local measurements with linear filters. Classical examples are the Sobel [11], Roberts [12], Prewitt [13] and Canny [14] edge detectors, which all use local derivative filters of fixed scale and only a few orientations. Such detectors tend to overemphasize small, unimportant edges and lead to noisy contour maps which are hard to use for subsequent higher-level processing. The key challenge is to reduce gradients due to repeated or stochastic textures, without losing edges due to object boundaries.

As a result, over the years, larger (non-local) neighborhoods, multiple scales and orientations, and multiple feature types have been incorporated into contour detectors. In fact, all top-performing methods in recent years fall into this category. Martin et al. [15] define linear operators for a number of cues such as intensity, color and texture. The resulting features are fed into a regression classifier that predicts edge strength; this is the popular Pb metric which gives, for each pixel in the image the probability of a contour at that point. Dollár et al. [16] use supervised learning, along with a large number of features and multiple scales to learn edge prediction. The features are collected in local patches in the image.

Recently, Lim et al. [7] have used random forest based learning on image patches to achieve state-of-the-art results. Their key idea is to use a dictionary of

human generated contours, called Sketch Tokens, as features for contours within a patch. The use of random forests makes inference fast. Dollár and Zitnick [8] also use random forests, but they further combine it with structured prediction to provide real-time edge detection. Ren and Bo [17] use sparse coding and oriented gradients to learn dictionaries of contour patches. They achieve excellent contour detection results on BSDS500.

The above methods all use patch-level measurements to create contour maps, with non-overlapping patches making independent decisions. This often leads to noisy and broken contours which are less likely to be useful for further processing for object recognition or image segmentation. Global methods utilize local measurements and embed them into a a framework which minimizes a global cost over all disjoint pairs of patches. Early methods in this line of work include that of Shashua and Ullman [18] and Elder and Zucker [19]. The paper of Shashua and Ullman used a simple dynamic programming approach to compute closed, smooth contours from local, disjoint edge fragments.

These globalization approaches tend to be fragile. More modern methods include a Conditional Random Field (CRF) presented in [20], which builds a probabilistic model for the completion problem, and uses loopy belief propagation to infer the closed contours. The highly successful gPb method of Arbeláez et al. [1] embeds the local Pb measure into a spectral clustering framework [21,22]. The resulting algorithm gives long, connected contours higher probability than short, disjoint contours.

The rarity of boundary patches has been studied in the literature before, e.g. [23]. We measure rarity based on pointwise mutual information [24] (PMI). PMI gives us a value per patch that allows us to build a pixel-level affinity matrix. This local affinity matrix is then embedded in a spectral clustering framework [1] to provide global contour information. PMI underlies many experiments in computational linguistics [25,26] to learn word associations (pairs of words that are likely to occur together), and recently has been used for improving image categorization [27]. Other information-theoretic takes on segmentation have been previously explored, e.g., [28]. However, to the best of our knowledge, PMI has never been used for contour extraction or image segmentation.

3 Information Theoretic Affinity

Consider the zebra in Figure 2. In this image, black stripes repeatedly occur next to white stripes. To a human eye, the stripes are grouped as a coherent object – the zebra. As discussed above, this intuitive grouping shows up in the image statistics: black and white pixels commonly co-occur next to one another, while white-green combinations are rarer, suggesting a possible object boundary where a white stripe meets the green background.

In this section, we describe a formal measure of the *affinity* between neighboring image features, based on statistical association. We denote a generic pair of neighboring features by random variables A and B, and investigate the joint distribution over pairings $\{A, B\}$.

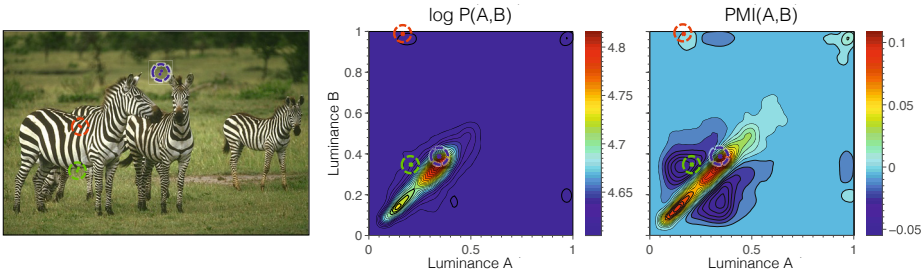

Fig. 2. Our algorithm works by reasoning about the pointwise mutual information (PMI) between neighboring image features. Middle column: Joint distribution of the luminance values of pairs of nearby pixels. Right column: PMI between the luminance values of neighboring pixels in this zebra image. In the left image, the blue circle indicates a smooth region of the image where all points are on the same object. The green circle a region that contains an object boundary. The red circle shows a region with a strong luminance edge that nonetheless does not indicate an object boundary. Luminance pairs chosen from within each circle are plotted where they fall in the joint distribution and PMI functions.

Let $p(A, B; d)$ be the joint probability of features A and B occurring at a Euclidean distance of d pixels apart. We define $P(A, B)$ by computing probabilities over multiple distances:

$$P(A, B) = \frac{1}{Z} \sum_{d=d_0}^{\infty} w(d) p(A, B; d), \tag{1}$$

where w is a weighting function which decays monotonically with distance d (Gaussian in our implementation), and Z is a normalization constant. We take the marginals of this distribution to get $P(A)$ and $P(B)$.

In order to pick out object boundaries, a first guess might be that affinity should be measured with joint probability $P(A, B)$. After all, features that always occur together probably should be grouped together. For the zebra image in Figure 2, the joint distribution over luminance values of nearby pixels is shown in the middle column. Overlaid on the zebra image are three sets of pixel pairs in the colored circles. These pairs correspond to pairs $\{A, B\}$ in our model. The pair of pixels in the blue circle are both on the same object and the joint probability of their colors – green next to green – is high. The pair in the bright green circle straddles an object boundary and the joint probability of the colors of this pair – black next to green – is correspondingly low.

Now consider the pair in the red circle. There is no physical object boundary on the edge of this zebra stripe. However, the joint probability is actually lower for this pair than for the pair in the green circle, where an object boundary did in fact exist. This demonstrates a shortcoming of using joint probability as a measure of affinity. Because there are simply more green pixels in the image than white pixels, there are more chances for green accidentally show up next to

any arbitrary other color – that is, the joint probability of green with any other color is inflated by the fact that most pixels in the image are green.

In order to correct for the baseline rarity of features A and B, we instead model affinity with a statistic related to *pointwise mutual information*:

$$\mathrm{PMI}_\rho(A, B) = \log \frac{P(A, B)^\rho}{P(A)P(B)}. \tag{2}$$

When $\rho = 1$, PMI_ρ is precisely the pointwise mutual information between A and B [24]. This quantity is the log of the ratio between the observed joint probability of $\{A, B\}$ in the image and the probability of this tuple were the two features independent. Equivalently, the ratio can be written as $\frac{P(A|B)}{P(A)}$, that is, how much more likely is observing A given that we saw B in the same local region, compared to the base rate of observing A in the image. When $\rho = 2$, we have a stronger condition: in that case the ratio in the log becomes $P(A|B)P(B|A)$. That is, observing A should imply that B will be nearby and vice versa. As it is unclear a priori which setting of ρ would lead to the best segmentation results, we instead treat ρ as a free parameter and select its value to optimize performance on a training set of images (see Section 4).

In the right column of Figure 2, we see the pointwise mutual information over features A and B. This metric appropriately corrects for the baseline rarities of white and black pixels versus gray and green pixels. As a result, the pixel pair between the stripes (red circle), is rated as more strongly mutually informative than the pixel pair that straddles the boundary (green circle). In Section 6.1 we empirically validate that PMI_ρ is indeed predictive of whether or not two points are on the same object.

4 Learning the Affinity Function

In this section we describe how we model $P(A, B)$, from which we can derive $\mathrm{PMI}_\rho(A, B)$. The pipeline for this learning is depicted in Figure 3(a) and (b). For each image on which we wish to measure affinities, we learn $P(A, B)$ specific to that image itself. Extensions of our approach could learn $P(A, B)$ from any type of dataset: videos, photo collections, images of a specific object class, etc. However, we find that modeling $P(A, B)$ with respect to the internal statistics of each test image is an effective approach for unsupervised boundary detection. The utility of internal image statistics has been previously demonstrated in the context of super-resolution and denoising [29] as well as saliency prediction [30].

Because natural images are piecewise smooth, the empirical distribution $P(A, B)$ for most images will be dominated by the diagonal $A \approx B$ (as in Figure 2). However, we are interested in the low probability, off-diagonal regions of the PDF. These off diagonal regions are where we find changes, including both repetitive, textural changes and object boundaries. In order to suppress texture while still detecting subtle object boundaries, we need a model that is able to capture the low probability regions of $P(A, B)$.

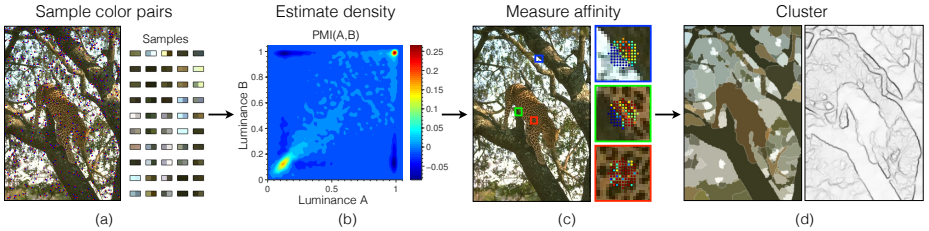

Fig. 3. Boundary detection pipeline: (a) Sample color pairs within the image. Red-blue dots represent pixel pair samples. (b) Estimate joint density P(A,B) and from this get PMI(A,B). (c) Measure affinity between each pair of pixels using PMI. Here we show the affinity between the center pixel in each patch and all neighboring pixels (hotter colors indicate greater affinity). Notice that there is low affinity across object boundaries but high affinity within textural regions. (d) Group pixels based on affinity (spectral clustering) to get segments and boundaries.

We use a nonparametric kernel density estimator [31] since it has high capacity without requiring an increase in feature dimensionality. We also experimented with a Gaussian Mixture Model but were unable to achieve the same performance as kernel density estimators.

Kernel density estimation places a kernel of probability density around every sample point. We need to specify on the form of the kernel and the number of samples. We used Epanechnikov kernels (i.e. truncated quadratics) owing to their computational efficiency and their optimality properties [32], and we place kernels at 10000 sample points per image. Samples are drawn uniformly at random from all locations in the image. First a random position x in the image is sampled. Then features A and B are sampled from image locations around x, such that A and B are d pixels apart. The sampling is done with weighting function $w(d)$, which is monotonically decreasing and gives maximum weight to $d = 2$. The vast majority of samples pairs $\{A, B\}$ are within distance $d = 4$ pixels of each other.

Epanechnikov kernels have one free parameter per feature dimension: the bandwidth of the kernel in that dimension. We select the bandwidth for each dimension through leave-one-out cross-validation to maximize the data likelihood. Specifically, we compute the likelihood of each sample given a kernel density model built from all the remaining samples. As a further detail, we bound the bandwidth to fall in the range $[0.01, 0.1]$ (with features scaled between $[0, 1]$) – this helps prevent overfitting to imperceptible details in the image, such as jpeg artifacts in a blank sky. To speed up evaluation of the kernel density model, we use the kd-tree implementation of Ihler and Mandel [33]. In addition, we smooth our calculation of PMI_ρ slightly by adding a small regularization constant to the numerator and denominator of Eq. 2.

Our model has one other free parameter, ρ. We choose ρ by selecting the value that gives the best performance on a training set of images completely independent of the test set, finding $\rho = 1.25$ to perform best.

5 Boundary Detection

Armed with an affinity function to tell us how pixels should be grouped in an image, the next step is to use this affinity function for boundary detection (Figure 3 (c) and (d)). Spectral clustering methods are ideally suited in our present case since they operate on affinity functions.

Spectral clustering was introduced in the context of image segmentation as a way to approximately solve the Normalized Cuts objective [34]. Normalized Cuts segments an image so as to maximize within segment affinity and minimize between segment affinity. To detect boundaries, we apply a spectral clustering using our affinity function, following the current state-of-the-art solution to this problem, gPb [1].

As input to spectral clustering, we require an affinity matrix, $\mathbf{W}$. We get this from our affinity function PMI_ρ as follows. Let i and j be indices into image pixels. At each pixel, we define a feature vector $\mathbf{f}$. Then, we define:

$$\mathbf{W}_{i,j} = e^{\mathrm{PMI}_\rho(\mathbf{f}_i, \mathbf{f}_j)} \tag{3}$$

The exponentiated values give us better performance than the raw PMI_ρ values. Since our model for feature pairings was learned on nearby pixels, we only evaluate the affinity matrix for pixels within a radius of 5 pixels from one another. Remaining affinities are set to 0.

In order to reduce model complexity, we make the simplifying assumption that different types of features are independent of one another. If we have M subsets of features, this implies that,

$$\mathbf{W}_{i,j} = e^{\sum_{k=1}^{M} \mathrm{PMI}_\rho(\mathbf{f}_i^k, \mathbf{f}_j^k)} \tag{4}$$

In our experiments, we use two feature sets: pixel color (in L*a*b* space) and the diagonal of the RGB color covariance matrix in a 3x3 window around each pixel. Thus for each pixel we have two feature vectors of dimension 3 each. Each feature vector is decorrelated using a basis computed over the entire image (one basis for color and one basis for variance).

Given $\mathbf{W}$, we compute boundaries by following the method of [1]: first we compute the generalized eigenvectors of the system $(\mathbf{D} - \mathbf{W})\mathbf{v} = \lambda \mathbf{D}\mathbf{v}$, where $\mathbf{D}_{i,i} = \sum_{j \neq i} \mathbf{W}_{i,j}$. Then we take an oriented spatial derivative over the first N eigenvectors with smallest eigenvalue ($N = 100$ in our experiments). This procedure gives a continuous-valued edge map for each of 8 derivative orientations. We then suppress boundaries that align with image borders and are within a few pixels of the image border. As a final post-processing step we apply the Oriented Watershed Transform (OWT) and create an Ultrametric Contour Map (UCM) [1], which we use as our final contour maps for evaluation.

In addition to the above approach, we also consider a multiscale variant. To incorporate multiscale information, we build an affinity matrix at three different image scales (subsampling the image by half in each dimension for each subsequent scale). To combine the information across scales, we use the multigrid,

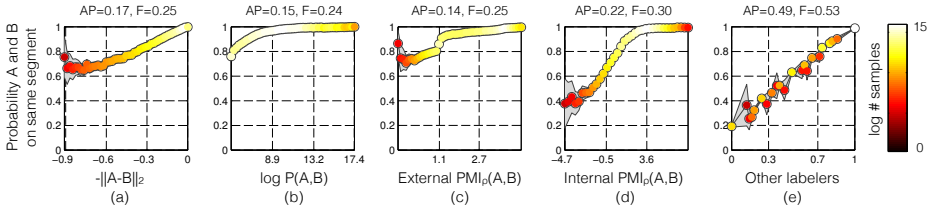

Fig. 4. Here we show the probability that two nearby pixels are on the same object segment as a function of various cues based on the pixel colors A and B. From left to right the cues are: (a) color difference, (b) color co-occurrence probability based on internal image statistics, (c) PMI based on external image statistics, (d) PMI based on internal image statistics, and (e) theoretical upper bound using the average labeling of $N-1$ human labelers to predict the Nth. Color represents number of samples that make up each datapoint. Shaded error bars show three times standard error of the mean. Performance is quantified by treating each cue as a binary classifier (with variable threshold) and measuring AP and maximum F-measure for this classifier (sweeping over threshold).

multiscale angular embedding algorithm of [35]. This algorithm solves the spectral clustering problem while enforcing that the edges at one scale are blurred versions of the edges at the next scale up.

6 Experiments

In this section, we present the results of a number of experiments. We first show that PMI is effective in detecting object boundaries. Then we show benchmarking results on the BSDS500 dataset. Finally, we show some segmentation results that are derived using our boundary detections.

6.1 Is PMI$_\rho$ Informative about Object Boundaries?

Given just two pixels in an image, how well can we determine if they span an object boundary? In this section, we analyze several possible cues based on a pair of pixels, and show that PMI$_\rho$ is more effective than alternatives.

Consider two nearby pixels with colors A and B. In Figure 4 we plot the probability that a random human labeler will consider the two pixels as lying on the same object segment as a function of various cues based on A and B.

To measure this probability, we sampled 20000 nearby pairs of pixels per image in the BSDS500 training set, using the same sampling scheme as in Section 4. For each pair of pixels, we also sample a random labeler from the set of human labelers for that image. The pixel pair is considered to lie on the same object segment if that labeler has placed them on the same segment.

A first idea is to use color difference $\|A-B\|_2$ to decide if the two pixels span a boundary (Figure 4(a); note that we use decorrelated L*a*b* color space with

Table 1. Evaluation on BSDS500

Algorithm	ODS	OIS	AP
Canny [14]	0.60	0.63	0.58
Mean Shift [36]	0.64	0.68	0.56
NCuts [37]	0.64	0.68	0.45
Felz-Hutt [38]	0.61	0.64	0.56
gPb [1]	0.71	0.74	0.65
gPb-owt-ucm [1]	0.73	0.76	0.73
SCG [9]	**0.74**	0.76	0.77
Sketch Tokens [7]	0.73	0.75	0.78
SE [8]	**0.74**	0.76	0.78
Our method – SS, color only	0.72	0.75	0.77
Our method – SS	0.73	0.76	**0.79**
Our method – MS	**0.74**	**0.77**	0.78

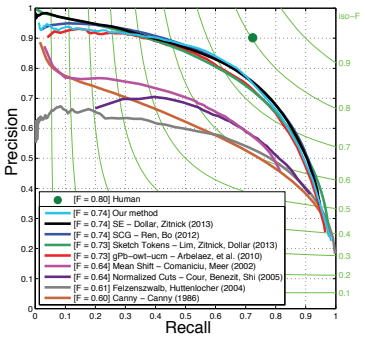

Fig. 5. Precision-recall curve on BSDS500. Figure copied from [8] with our results added.

values normalized between 0 and 1). Color difference has long been used as a cue for boundary detection and unsurprisingly it is predictive of whether or not A and B lie on the same segment.

Beyond using pixel color difference, boundary detectors have improved over time by reasoning over larger and larger image regions. But is there anything more we can squeeze out of just two pixels?

Since boundaries are rare events, we may next try $\log P(A, B)$. As shown in Figure 4(b), rarer color combinations are indeed more likely to span a boundary. However, $\log P(A, B)$ is still a poor predictor.

Can we do better if we use PMI? In Figure 4(c) and (d) we show that, yes, $\text{PMI}_\rho(A, B)$ (with $\rho = 1.25$) is quite predictive of whether or not A and B lie on the same object. Further, comparing Figure 4(c) and (d), we find that it is important that the statistics for PMI_ρ be adapted to the test image itself. Figure 4(c) shows the result when the distribution $P(A, B)$ is learned over the entire BSDS500 training set. These *external* statistics are poorly suited for modeling individual images. On the other hand, when we learn $P(A, B)$ based on color co-occurrences *internal* to an image, PMI_ρ is much more predictive of the boundaries in that image (Figure 4(d)).

6.2 Benchmarks

We run experiments on three versions of our algorithm: single scale using only pixel colors as features (labeled as *SS, color only*), single scale using both color and color variance features (*SS*), and multiscale with both color and variance features (*MS*). Where possible, we compare against the top performing previous contour detectors. We choose the Structured Edges (SE) detector [8] and gPb-owt-ucm detector [1] to compare against more extensively. These two methods currently achieve state-of-the-art results. SE is representative of the supervised learning approach to edge detection, and gPb-owt-ucm is representative of affinity-based approaches, which is also the category into which our algorithm falls.

BSDS500: The Berkeley Segmentation Dataset [39,1] has been frequently used as a benchmark for contour detection algorithms. This dataset is split into 200 training images, 100 validation images, and 200 test images. Although our algorithm requires no extensive training, we did tune our parameters (in particular ρ) to optimize performance on the validation set. In Table 1 and Figure 5, we report our performance on the test set. ODS refers to the F-measure at the optimal threshold across the entire dataset. OIS refers to the per-image best F-measure. AP stands for area under the precision-recall curve. On each of these popular metrics, we match or outperform the state-of-the-art. It is also notable that our *SS, color only* method gets results close to the state-of-the-art, as this method only uses *pixel pair colors* for its features. We believe that this result is noteworthy as it shows that with carefully designed nonlinear methods, it is possible to achieve excellent results without using high-dimensional feature spaces and extensive engineering.

In Figure 8 we show example detections by our algorithm on the BSDS500 test set. These results are with our *MS* version with $\rho = 1.25$. We note that our results have fewer boundaries due to texture, and crisper boundary localization. Further examples can be seen in the supplementary materials.

High Resolution Edges: One of the striking features of our algorithm is the high resolution of its results. Consider the white object in Figure 6. Here our algorithm is able to precisely match the jagged contours of this object, whereas gPb-owt-ucm incurs much more smoothing. As discussed in the introduction, good boundary detections should be both "correct" (detecting real object boundaries) and "crisp" (precisely localized along the object's contour). The standard BSDS500 metrics do not distinguish between these two criteria.

However, the benchmark metrics do include a parameter, r, related to crispness. A detected edge can be r pixels away from a ground truth edge and still be considered a correct detection. The standard benchmark code uses $r = 4.3$ pixels for BSDS500 images. Clearly, this default setting of r cannot distinguish whether or not an algorithm is capturing details above a certain spatial frequency. Varying r dramatically affects performance (Figure 7). In order to benchmark on the task of detecting "crisp" contours, we evaluate our algorithm on three settings of r: r_0, $r_0/2$, and $r_0/4$, where $r_0 = 4.3$ pixels, the default setting.

In Figure 7, we plot our results and compare against SE (with non-maximal suppression) and gPb-owt-ucm. While all three methods perform similarly at $r = r_0$, our method increasingly outperforms the others when r is small. This quantitatively demonstrates that our method is matching crisp, high resolution contours better than other state-of-the-art approaches.

Speed: Recently several edge detectors have been proposed that optimize speed while also achieving good results [8,7]. The current implementation of our method is not competitive with these fast edge detectors in terms of speed. To achieve our *MS* results above, our highly unoptimized algorithm takes around 15 minutes per image on a single core of an Intel Core i7 processor.

However, we can tune the parameters of our algorithm for speed at some cost to resolution. Doing so, we can match our state-of-the-art results (ODS=0.74,

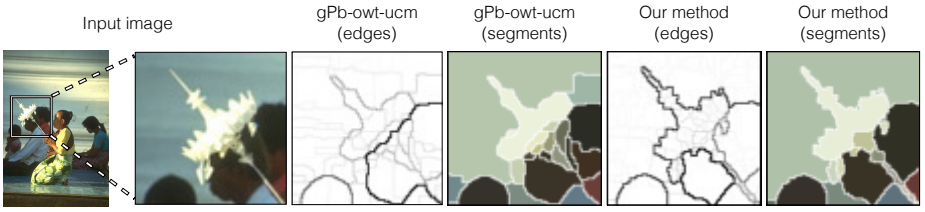

Fig. 6. Here we show a zoomed in region of an image. Notice that our method preserves the high frequency contour variation while gPb-owt-ucm does not.

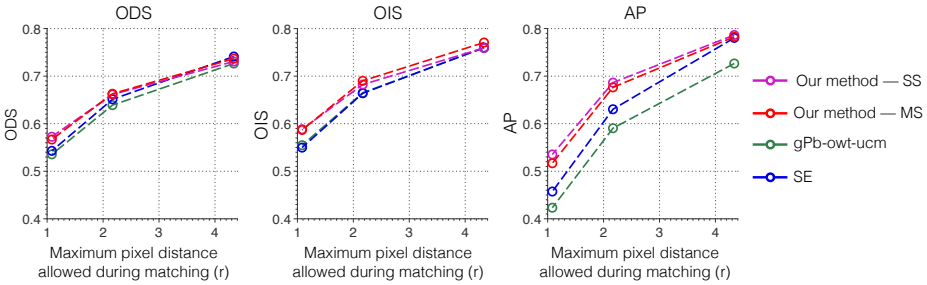

Fig. 7. Performance as a function of the maximum pixel distance allowed during matching between detected boundaries and ground truth edges (referred to as r in the text). When r is large, boundaries can be loosely matched and all methods do well. When r is small, boundaries must be precisely localized, and this is where our method most outperforms the others.

OIS=0.77 AP=0.80 on BSDS500 using the standard $r = 4.3$) in about 30 seconds per image (again on a single core of an i7 processor). The tradeoff is that the resulting boundary maps are not as well localized (at $r = 1.075$, this method falls to ODS=0.52, OIS=0.53, AP=0.43, which is well below our full resolution results in Figure 7). The speed up comes from 1) downsampling each image by half and running our *SS* algorithm, 2) approximating $\mathrm{PMI}_\rho(A, B)$ using a random forest prior to evaluation of $\mathbf{W}$, and 3) using a fixed kernel bandwidth rather than adapting it to each test image. Code for both fast and high resolution variants of our algorithm will be available at `http://web.mit.edu/phillipi/crisp_boundaries`.

6.3 Segmentation

Segmentation is a complementary problem to edge detection. In fact, our edge detector automatically also gives us a segmentation map, since this is a byproduct of producing an Ultrametric Contour Map [1]. This ability sets our approach, along with gPb-owt-ucm, apart from many supervised edge detectors such as SE, for which a segmentation map is not a direct byproduct. In Figure 9, we compare results of segmentations with our contours to those of gPb contours. Notice that in the coral image, our method recovers the precise shape of the bottom, reddish

Fig. 8. Contour detection results for a few images in the BSDS500 test set, comparing our method to gPb [1] and SE [8]. In general, we suppress texture edges better (such as on the fish in the first row), and recover crisper contours (such as the leaves in upper-right of the fifth row). Note that here we show each method without edge-thinning (that is, we leave out non-maximal suppression in the case of SE, and we leave out OWT-UCM in the case of gPb and our method).

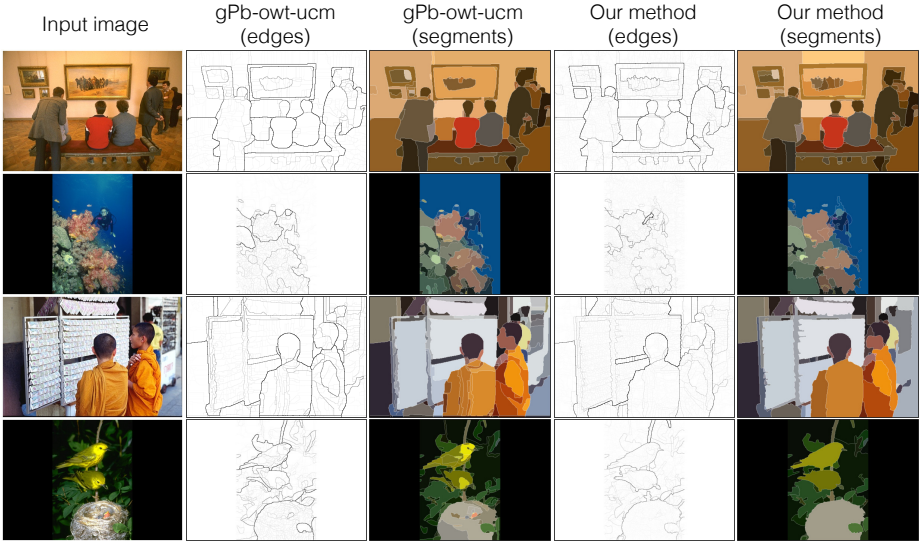

Fig. 9. Example segmentations for a few images in the BSDS500 test set, comparing the results of running OWT-UCM segmentation on our contours and those of gPb [1].

coral, while gPb-owt-ucm misses some major features of the contour. Similarly, in the bird image, our method captures the beak of the top bird, whereas gPb-owt-ucm smooths it away.

7 Discussion

In this paper, we have presented an intuitive and principled method for contour detection which achieves state-of-the-art results. We have shown that, contrary to recent trends, it is possible to achieve excellent boundary detection results using very local information and low-dimensional feature spaces. This is achieved through the use of a novel statistical framework based on pointwise mutual information.

In future work, we plan to extend the learning to videos or multiple images. This could be used to build statistical model of specific objects. Such a model would have direct applications in object detection and recognition.

Acknowledgements. We thank Joseph Lim, Zoya Bylinskii, and Bill Freeman for helpful discussions. This work is supported by NSF award 1212849 Reconstructive Recognition, and by Shell Research. P. Isola is supported by an NSF graduate research fellowship.

References

1. Arbeláez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 33(5), 898–916 (2011)
2. Arbeláez, P., Hariharan, B., Gu, C., Gupta, S., Bourdev, L., Malik, J.: Semantic segmentation using regions and parts. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3378–3385. IEEE (2012)
3. Levin, A., Weiss, Y.: Learning to combine bottom-up and top-down segmentation. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3954, pp. 581–594. Springer, Heidelberg (2006)
4. Shotton, J., Blake, A., Cipolla, R.: Contour-based learning for object detection. In: Tenth IEEE International Conference on Computer Vision, ICCV 2005, vol. 1, pp. 503–510. IEEE (2005)
5. Opelt, A., Pinz, A., Zisserman, A.: A boundary-fragment-model for object detection. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 575–588. Springer, Heidelberg (2006)
6. Barron, J., Malik, J.: Shape, illumination, and reflectance from shading. Technical report, Berkeley Tech. Report (2013)
7. Lim, J.J., Zitnick, C.L., Dollár, P.: Sketch tokens: A learned mid-level representation for contour and object detection. In: CVPR, pp. 3158–3165 (2013)
8. Dollár, P., Zitnick, C.: Structured Forests for Fast Edge Detection. In: ICCV (2013)
9. Xiaofeng, R., Bo, L.: Discriminatively trained sparse code gradients for contour detection. In: NIPS, pp. 593–601 (2012)
10. Sobel, I., Feldman, G.: A 3x3 isotropic gradient operator for image processing (1968)
11. Duda, R.O., Hart, P.E., et al.: Pattern classification and scene analysis, vol. 3. Wiley, New York (1973)
12. Roberts, L.G.: Machine Perception of Three-Dimensional Solids. PhD thesis, Massachusetts Institute of Technology (1963)
13. Prewitt, J.M.: Object enhancement and extraction. Picture Processing and Psychopictorics 10(1), 15–19 (1970)
14. Canny, J.: A computational approach to edge detection. IEEE Transactions on Pattern Analysis and Machine Intelligence (6), 679–698 (1986)
15. Martin, D.R., Fowlkes, C.C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(5), 530–549 (2004)
16. Dollár, P., Tu, Z., Belongie, S.: Supervised learning of edges and object boundaries. In: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 1964–1971. IEEE (2006)
17. Ren, X., Bo, L.: Discriminatively trained sparse code gradients for contour detection. In: NIPS (2012)
18. Sha'ashua, A., Ullman, S.: Structural saliency: The detection of globally salient structures using a locally connected network. In: ICCV (1988)
19. Elder, J.H., Zucker, S.W.: Computing contour closure. In: Buxton, B.F., Cipolla, R. (eds.) ECCV 1996. LNCS, vol. 1064, pp. 399–412. Springer, Heidelberg (1996)
20. Ren, X., Fowlkes, C.C., Malik, J.: Scale-invariant contour completion using conditional random fields. In: Tenth IEEE International Conference on Computer Vision, ICCV 2005, vol. 2, pp. 1214–1221. IEEE (2005)
21. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 888–905 (2000)

22. Malik, J., Belongie, S., Leung, T., Shi, J.: Contour and texture analysis for image segmentation. International Journal of Computer Vision 43(1), 7–27 (2001)
23. Zoran, D., Weiss, Y.: Natural images, gaussian mixtures and dead leaves. In: NIPS (2012)
24. Fano, R.M.: Transmission of information: A statistical theory of communications. American Journal of Physics 29, 793–794 (1961)
25. Church, K.W., Hanks, P.: Word association norms, mutual information, and lexicography. Computational Linguistics 16(1), 22–29 (1990)
26. Chambers, N., Jurafsky, D.: Unsupervised learning of narrative event chains. In: ACL, pp. 789–797 (2008)
27. Bengio, S., Dean, J., Erhan, D., Ie, E., Le, Q., Rabinovich, A., Shlens, J., Singer, Y.: Using web co-occurrence statistics for improving image categorization. ArXiv preprint ArXiv:1312.5697 (2013)
28. Mobahi, H., Rao, S., Yang, A., Sastry, S., Ma, Y.: Segmentation of natural images by texture and boundary compression. International Journal of Computer Vision 95, 86–98 (2011)
29. Zontak, M., Irani, M.: Internal Statistics of a Single Natural Image. In: CVPR (2011)
30. Margolin, R., Tal, A., Zelnik-Manor, L.: What makes a patch distinct? In: 2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1139–1146. IEEE (2013)
31. Parzen, E., et al.: On estimation of a probability density function and mode. Annals of Mathematical Statistics 33(3), 1065–1076 (1962)
32. Epanechnikov, V.A.: Non-parametric estimation of a multivariate probability density. Theory of Probability & its Applications 14(1), 153–158 (1969)
33. Ihler, A., Mandel, M.: `http://www.ics.uci.edu/~ihler/code/kde.html`
34. Shi, J., Malik, J.: Normalized cuts and image segmentation. PAMI 22(8), 888–905 (2000)
35. Maire, M., Yu, S.X.: Progressive Multigrid Eigensolvers for Multiscale Spectral Segmentation. In: ICCV (2013)
36. Comaniciu, D., Meer, P.: Mean shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(5), 603–619 (2002)
37. Cour, T., Benezit, F., Shi, J.: Spectral segmentation with multiscale graph decomposition. In: CVPR (2005)
38. Felzenszwalb, P.F., Huttenlocher, D.P.: Efficient graph-based image segmentation. International Journal of Computer Vision 59(2), 167–181 (2004)
39. Martin, D., Fowlkes, C., Tal, D.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: ICCV (2001)