Efficient Closed-Form Solution to Generalized Boundary Detection

Marius Leordeanu[1], Rahul Sukthankar[3,4], and Cristian Sminchisescu[2,1]

[1]Institute of Mathematics of the Romanian Academy
[2]Faculty of Mathematics and Natural Science, University of Bonn
[3]Google Research [4]Carnegie Mellon University

Abstract. Boundary detection is essential for a variety of computer vision tasks such as segmentation and recognition. We propose a unified formulation for boundary detection, with closed-form solution, which is applicable to the localization of different types of boundaries, such as intensity edges and occlusion boundaries from video and RGB-D cameras. Our algorithm simultaneously combines low- and mid-level image representations, in a single eigenvalue problem, and we solve over an infinite set of putative boundary orientations. Moreover, our method achieves state of the art results at a significantly lower computational cost than current methods. We also propose a novel method for soft-segmentation that can be used in conjunction with our boundary detection algorithm and improve its accuracy at a negligible extra computational cost.

1 Introduction

Boundary detection is a fundamental task in computer vision, with broad applicability in areas such as feature extraction, object recognition and image segmentation. The majority of papers on edge detection have focused on using only low-level cues, such as pixel intensity or color [1–5]. Recent work has started exploring the problem of boundary detection based on higher-level representations of the image, such as motion, surface and depth cues [6–8], segmentation [9], as well as category specific information [10, 11].

In this paper we propose a general formulation for boundary detection that can be applied, in principle, to the identification of any type of boundaries, such as general edges from low-level static cues (Figure 6), and occlusion boundaries from motion and depth cues (Figures 1, 7, 8). We generalize the classical view of boundaries from sudden signal changes on the original low-level image input [1–4, 12–14], to a locally linear (planar or step-wise) model on multiple layers of the input, over a relatively large image region. The layers can be interpretations of the image at different levels of visual processing, which could be low-level (e.g., color or grey level intensity), mid-level (e.g., segmentation, optical flow), or high-level (e.g., object category segmentation).

Despite the abundance of research on boundary detection, there is no general formulation of this problem. In this paper, we make the popular but implicit intuition of boundaries explicit: boundary pixels mark the transition from one

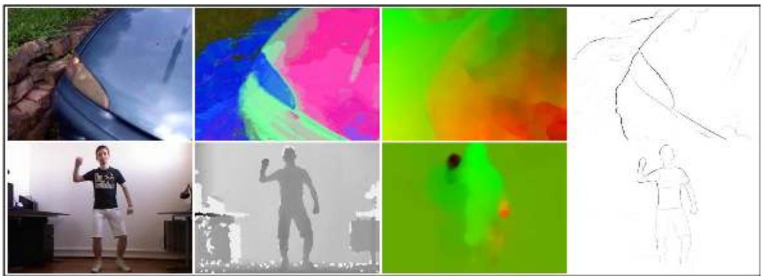

Fig. 1. Our method (Gb) combines, in a unified formulation, different types of information (first three columns) to find boundaries (right column). Top row: Gb uses color, soft-segmentation and optical flow. Bottom row: Gb uses color, depth and optical flow.

relatively constant property region to another, in appropriate interpretations of the image. We can summarize our assumptions as follows:

1. A boundary separates different image regions, which in the absence of noise are almost constant, at some level of image interpretation or processing. For example, at the lowest level, a region could have constant intensity. At a higher-level, it could be a region delimiting an object category, in which case the output of a category-specific classifier would be constant.
2. For a given image, boundaries in one layer often coincide, in terms of position and orientation, with boundaries in other layers. For example, discontinuities in intensity are typically correlated with discontinuities in optical flow, texture or other cues. Moreover, the boundaries that align across multiple layers typically correspond to the semantic boundaries that interest humans.

Based on these observations, we develop a unified model that can simultaneously consider both lower-level and higher-level information.

Classical vector-valued techniques on multi-images [12,13,15] can be simultaneously applied to several image channels, but differ from the proposed approach in a fundamental way: they are specifically designed for low-level input, by using first or second-order derivatives of the image channels, with edge models limited to very small neighborhoods, as needed for approximating the derivatives. Derivatives are very often noisy and usually do not have sufficient spatial support to indicate true object boundaries with high confidence. Moreover, even though edges from one layer coincide with those from a different layer, their location may not match perfectly — an assumption implicitly made by the use of derivatives. We argue that in order to confidently classify boundary pixels and combine multiple layers of information, one must go beyond a few pixels, to much larger neighborhoods, in line with more recent methods [5,9,16,17].

The main advantage of our approach over current methods is the efficient estimation of boundary strength and orientation in a single closed-form computation. The idea behind Pb and its variants [9,16] is to classify each possible

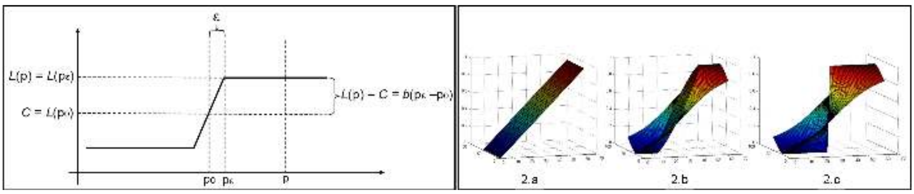

Fig. 2. Left: 1D view of our model. Right: 2D view of our boundary model with different values of ϵ relative to the window size W: 2.a) $\epsilon > W$; 2.b) $\epsilon = W/2$; 2.c) $\epsilon = W/1000$. For small ϵ the model is a step, along the normal passing through the window center.

boundary pixel based on the histogram difference in color and texture information between the two half disks on each side of a putative orientation, for a fixed number of candidate angles. The separate computation for each orientation considerably increases the computational cost and limits orientation estimates to a particular angular quantization, thus affecting the estimated probability of boundary.

We summarize our contributions as follows: **1)** we present a novel boundary model with an efficient closed-form solution for generalized boundary detection; **2)** we recover exact boundary normals through direct estimation rather than evaluating coarsely sampled orientation candidates as in [16]; **3)** we optimize simultaneously over both low and mid-levels of image processing, and can easily incorporate outputs from new image interpretation methods. This is in contrast to current approaches [6, 7, 9] that process low and mid-level layers separately and combine them in different ways to detect different types of boundaries. **4)** we only learn a small set of parameters, enabling efficient training with limited data. Our approach essentially bridges the gap between model fitting methods such as [18, 19], and recent learning-based boundary detectors.

2 Generalized Boundary Model

Given a $N_x \times N_y$ image I, let its k-th layer L_k be some real-valued array, of the same size, whose boundaries are relevant to our task. For example, L_k could contain, at each pixel, values from a color channel, filter responses, optical flow, or the output of a patch-based binary classifier trained to detect a specific color distribution, texture or a certain object category.[1] Thus, L_k could consist of relatively constant regions separated by boundaries.

We expect that boundaries in different layers may not precisely align. Given a set of layers, each corresponding to a particular interpretation of the image, we wish to identify the most consistent boundaries across these layers. The output of our method for each point $\mathbf{p}$ on the $N_x \times N_y$ image grid is a real-valued probability

[1] The output of a discrete-valued multi-class classifier can be encoded as multiple input layers, where each layer represents a given label.

that $\mathbf{p}$ lies on a boundary, given the information in all image interpretations L_k centered at $\mathbf{p}$.

We model a boundary point in layer L_k as a transition, either sudden or gradual, in the corresponding values of L_k along the normal to the boundary. If several K such layers are available, let $\mathbf{L}$ be a three-dimensional array of size $N_x \times N_y \times K$, such that $\mathbf{L}(x, y, k) = L_k(x, y)$, for each k. Thus, $\mathbf{L}$ contains all the information available for the current boundary detection problem, given multiple interpretations of the image. Figure 1 illustrates how we perform boundary detection by combining different layers, such as color, depth, soft-segmentation and optical flow.

Let $\mathbf{p}_0$ be the center of a window $W(\mathbf{p}_0)$ of size $\sqrt{N_W} \times \sqrt{N_W}$, where N_W is the number of pixels in the window. For each image location $\mathbf{p}_0$ we want to evaluate the probability of boundary using the information in $\mathbf{L}$, restricted to that particular window. For any $\mathbf{p}$ within the window, we model the boundary with the following locally linear approximation:

$$L_k(\mathbf{p}) \approx C_k(\mathbf{p}_0) + b_k(\mathbf{p}_0)(\hat{\mathbf{p}}_\epsilon - \mathbf{p}_0)^\top \mathbf{n}(\mathbf{p}_0). \tag{1}$$

Here b_k is nonnegative and corresponds to the boundary "height" for layer k at location $\mathbf{p}_0$; $\hat{\mathbf{p}}_\epsilon$ is the closest point to $\mathbf{p}$ (projection of $\mathbf{p}$) on the disk of radius ϵ centered at $\mathbf{p}_0$; $\mathbf{n}(\mathbf{p}_0)$ is the normal to the boundary and $C_k(\mathbf{p}_0)$ is a constant over the window $W(\mathbf{p}_0)$. Note that if we set $C_k(\mathbf{p}_0) = L_k(\mathbf{p}_0)$ and use a sufficiently large ϵ such that $\hat{\mathbf{p}}_\epsilon = \mathbf{p}$, our model reduces to the first-order Taylor expansion of $L_k(\mathbf{p})$ around the current $\mathbf{p}_0$.

As shown in Figure 2, ϵ controls the steepness of the boundary, going from completely planar when ϵ is large to a sharp step-wise discontinuity through the window center $\mathbf{p}_0$, as ϵ approaches zero. When ϵ is very small we have a step along the normal through the window center, and a sigmoid that flattens as we move farther away from the center, along the boundary normal. As ϵ increases, the model flattens to become a perfect plane for any ϵ greater than the window radius. In 2D, our model is not a ramp, which enables it to handle corners as well as edges. The idea of ramp edges has been explored in the literature before, albeit very differently [20].

When the window is far from any boundary, the value of b_k will be near zero, since the only variation in the layer values is due to noise. If we are close to a boundary, then b_k becomes large. The term $(\hat{\mathbf{p}}_\epsilon - \mathbf{p}_0)^\top \mathbf{n}(\mathbf{p}_0)$ approximates the sign indicating the side of the boundary: it does not matter on which side we are, as long as a sign change occurs when the boundary is crossed. When a true boundary is present within several layers at the same position ($b_k(\mathbf{p}_0)$ is non-zero and possibly different, for several k) the normal to the boundary should be consistent. Thus, we model the boundary normal $\mathbf{n}$ as common across all layers.

We can now write the above equation in matrix form for all layers, with the same window size and location as follows: let $\mathbf{X}$ be a $N_W \times K$ matrix with a row i for each location $\mathbf{p}_i$ of the window and a column for each layer k, such that $X_{i;k} = L_k(\mathbf{p}_i)$. Similarly, we define $N_W \times 2$ position matrix $\mathbf{P}$: on its i-th row we store the x and y components of $\hat{\mathbf{p}}_\epsilon - \mathbf{p}_0$ for the i-th point of the window.

Let $\mathbf{n} = [n_x, n_y]$ denote the boundary normal and $\mathbf{b} = [b_1, b_2, \ldots, b_K]$ the step sizes for layers $1, 2, \ldots, K$. Also, let us define the rank-1 $2 \times K$ matrix $\mathbf{J} = \mathbf{n}^\top \mathbf{b}$. We also define matrix $\mathbf{C}$ of the same size as $\mathbf{X}$, with each column k constant and equal to $C_k(\mathbf{p}_0)$. We rewrite Equation 1 (dropping the dependency on $\mathbf{p}_0$ for notational simplicity), with unknowns $\mathbf{J}$ and $\mathbf{C}$:

$$\mathbf{X} \approx \mathbf{C} + \mathbf{P}\mathbf{J}. \tag{2}$$

Since $\mathbf{C}$ is a matrix with constant columns, and each column of $\mathbf{P}$ sums to 0, we have $\mathbf{P}^\top \mathbf{C} = \mathbf{0}$. Thus, by multiplying both sides of the equation above by $\mathbf{P}^\top$, we eliminate the unknown $\mathbf{C}$. Moreover, it can be easily shown that $\mathbf{P}^\top \mathbf{P} = \alpha \mathbf{I}$, i.e., the identity matrix scaled by a factor α, which can be computed since $\mathbf{P}$ is known. We finally obtain a simple expression for the unknown $\mathbf{J}$ (since both $\mathbf{P}$ and $\mathbf{X}$ are known):

$$\mathbf{J} \approx \frac{1}{\alpha} \mathbf{P}^\top \mathbf{X}. \tag{3}$$

Since $\mathbf{J} = \mathbf{n}^\top \mathbf{b}$ it follows that $\mathbf{J}\mathbf{J}^\top = ||\mathbf{b}||^2 \mathbf{n}^\top \mathbf{n}$ is symmetric and has rank 1. Then $\mathbf{n}$ can be estimated as the principal eigenvector of $\mathbf{M} = \mathbf{J}\mathbf{J}^\top$ and $||\mathbf{b}||$ as the square root of its largest eigenvalue. $||\mathbf{b}||$ is the norm of the boundary step vector $\mathbf{b} = [b_1, b_2, ..., b_K]$ and captures the overall strength of boundaries from all layers simultaneously. If layers are properly scaled, then $||\mathbf{b}||$ could be used as a measure of boundary strength. Once we identify $||\mathbf{b}||$, we pass it through a one-dimensional logistic model to obtain the probability of boundary, similarly to recent methods [9,16]. The parameters of the logistic model are learned using standard procedures, explained in Section 3.2. The normal to the boundary $\mathbf{n}$ is then used for non-maxima suppression. Note that $||\mathbf{b}||$ is different from the gradient of multi-images [12,13] that is computed from local derivatives, which could be noisy and lack sufficient spatial support. We compute the boundary strength by fitting a model, which, by controlling the window size and ϵ, can vary from a small to a large patch and from planar to step-wise.

Additionally, we propose to weigh the importance of each pixel in a window by an isotropic 2D Gaussian located at the window center $\mathbf{p_0}$. This puts more weight on model fitting errors from data points that are closer to the window center. The idea is implemented by multiplying each row of both $\mathbf{X}$ and $\mathbf{P}$ with the Gaussian weight corresponding to that particular location. We mention that the introduction of Gaussian weighting does not change the model (Equation 2), but only the contributions of data points to the model fitting process: $C_k(\mathbf{p_0})$, with its rows also multiplied by the corresponding Gaussian weights, still cancels out and the final Equation 3 remains valid. As seen in the middle plot of Figure 3, the performance is significantly influenced by the choice of Gaussian standard deviation σ_G, which confirms our assumption that points closer to the boundary should constrain the model parameters more.

In our experiments we used a window radius equal to 2% of the image diagonal, $\epsilon = 1$ pixel, and Gaussian σ_G equal to half of the window radius. These parameters produced the best F-measure on the BSDS300 training set [16] and

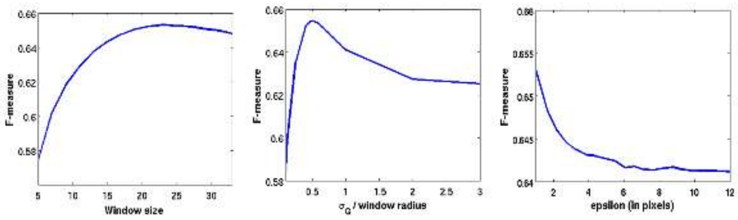

Fig. 3. Evaluation on BSDS300 test set by varying the window size (in pixels), σ_G of the Gaussian weighting (relative to window radius) and ϵ. One parameter is varied, while the others are set to their optimum (learned from training images). Left: windows with large spatial support give a significantly better accuracy. Middle: points closer to the boundary should contribute more to the model, as evidenced by the best $\sigma_G \approx$ half of the window radius. Right: small ϵ leads to better performance, confirming the usefulness of our step-wise model.

were also near-optimal on the test set, as shown in Figure 3. We draw the following conclusions about our model: 1) a large window size leads to significantly better performance as more evidence can be used in reasoning about boundaries. Note that when the window size is small our model becomes similar to methods based on local approximation of derivatives [4, 12, 13, 15]. 2) the usage of a small ϵ produces boundaries with significantly better localization and strength. It strongly suggests that boundary transitions in natural images tend to be sudden, not gradual. 3) the Gaussian weighting is justified: the model is better fitted if more weight is placed on points closer to the boundary.

3 Algorithm

Before applying the main algorithm we scale each layer in $\mathbf{L}$ according to its importance, which may be problem dependent. We learn the scaling of layers from training data using a direct search method [21] to optimize the F-measure (Section 3.2). Algorithm 1 (termed Gb) summarizes the proposed approach.

The pseudo-code presented in Algorithm 1 gives a description of Gb that directly relates to our boundary model. Upon closer inspection we observe that elements of $\mathbf{M}$ can also be computed exactly by convolving each layer L_k twice, using two different kernels: $H_x(x-x_0, y-y_0) \propto g(x-x_0, y-y_0)^2(x_\epsilon - x_0)$ and $H_y(x-x_0, y-y_0) \propto g(x-x_0, y-y_0)^2(y_\epsilon - y_0)$, and then combining the results. Here $g(x-x_0, y-y_0)$ is the Gaussian weight applied at location $(x-x_0, y-y_0)$ and $(x_\epsilon, y_\epsilon) = \mathbf{p}_\epsilon$. This observation leads to a straightforward implementation.[2] Note the analytic difference between our filters and Derivative of Gaussian filters (i.e., $G_x(x-x_0, y-y_0) \propto g(x-x_0, y-y_0)(x-x_0)$), which could be used for computing the gradient of multi-images [13]. While Gaussian derivatives have the computational advantage of being separable, when used for computing the

[2] Code available online at: `http://www.imar.ro/clvp/code/Gb`

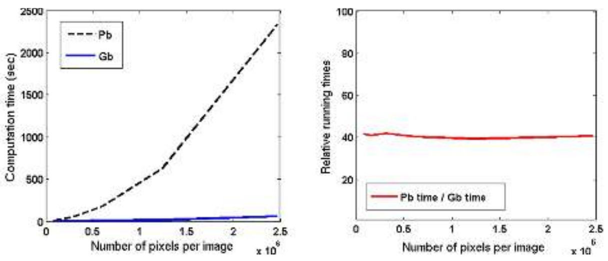

Fig. 4. Left: Edge detection run times on a 3.2 GHz desktop for our MATLAB implementation of Gb vs. the publicly available code of Pb [16]. Right: ratio of run time of Pb to run time of Gb. Each algorithm runs over a single scale and uses the same window size, which is a constant fraction of the image size. Here, Gb is 40× faster.

Algorithm 1 Gb: Generalized Boundary Detection

```
Initialize $\mathbf{L}$, scaled appropriately.
Initialize $w_0$ and $w_1$.
Pre-compute matrix $\mathbf{P}$
for all pixels $\mathbf{p}$ do
    $\mathbf{M} \leftarrow (\mathbf{P}^\top \mathbf{X_p})(\mathbf{P}^\top \mathbf{X_p})^\top$
    $(\mathbf{v}, \lambda) \leftarrow$ principal eigenpair of $\mathbf{M}$
    $b_\mathbf{p} \leftarrow \frac{1}{1+\exp(w_0 + w_1\sqrt{\lambda})}$
    $\theta_\mathbf{p} \leftarrow \text{atan2}(v_y, v_x)$
end for
return $\mathbf{b}$, $\theta$
```

gradient of multi-images they produce boundaries of inferior quality (see Table 2).

3.1 Computational Complexity

The overall complexity of Gb is straightforward to derive. For each pixel $\mathbf{p}$, the most expensive step is computing the matrix $\mathbf{M}$, which has $O((N_W + 2)K)$ complexity, where N_W denotes the number of pixels in the window and K the number of layers. $\mathbf{M}$ is a 2×2 matrix, so computing its eigenpair $(\mathbf{v}, \lambda)$ is a closed-form operation, with small fixed cost. Thus, for a fixed N_W and a total of N pixels per image the overall complexity is $O(KN_WN)$. If N_W is a fraction f of N, then complexity becomes $O(fKN^2)$.

The running time of Gb compares favorably to that of Pb [9, 16]. Pb in its exact form has complexity $O(fKN_oN^2)$, where N_o is a discrete number of candidate orientations. Both Gb and Pb are quadratic in the number of image pixels. However, Pb has a significantly larger fixed cost per pixel as it requires the computation of histograms for each individual image channel and orientation. In Figure 4, we show the run times for Gb and Pb (publicly available code) on a 3.2GHz desktop in MATLAB, on the same images, using the same window size and a single scale. While Gb produces boundaries of similar quality (see Table 2), it is consistently faster than Pb (about 40×), independent of the image size (Figure 4, right plot). For example, on 0.15 MP images the times are: 19.4 sec for Pb vs. 0.48 sec for Gb; to process 2.5 MP images, Pb takes 38 min while Gb only 57 sec.

A fast parallel implementation of gPb [9] is proposed in [22]. The authors implement the method directly on the high-performance Nvidia GTX 280 graphics card with a high degree of parallelism (30 multiprocessors). Local Pb is computed at three different scales. The authors offer two implementations for local cues: one for the exact computation and the other for a faster approximate computation that uses integral images and is linear in the number of image pixels. The approximation has $O(fKN_oN_bN)$ time complexity, where N_b is the number of histogram bins for the different image channels and N_o is the number of candidate orientations. Note that N_oN_b is large in practice and affects the overall running time considerably. It requires computing (and possibly storing) a large number of integral images, one for each combination of (histogram bin, image channel, orientation). The actual number is not explicitly stated in [22], but we estimate that it is in the order of one thousand per input image (4 channels × 8 orientations × 32 histogram bins = 1024). The approximation also requires special processing for the rotated integral images of texton labels, to minimize interpolation artifacts. The authors propose a solution based on Bresenham lines, which further affects the discretization of the rotation angle. In Table 1 we present run time comparisons with Pb's local cues computation from [22]. Our exact implementation of Gb (using 3 color layers) in MATLAB is 8 times faster than the exact parallel computation of Pb over 3 scales on GTX 280.

Table 1. Run times: Gb implementation in MATLAB on a 3.2 Ghz desktop vs. Catanzaro et al.'s parallel computation of local cues on Nvidia GTX 280 [22].

Algorithm	Gb (exact)	[22] (exact)	[22] (approx.)
Run time (sec.)	0.473	4.0	0.569

3.2 Learning

Our model uses a small number of parameters. Only two parameters (w_0, w_1) are needed for the logisic function that models the probability of boundary (Algorithm 1). For layer scaling the maximum number of parameters needed is equal to the number of layers. We reduce this number by tying the scaling for layers of the same type: 1) for color (in CIELAB space) we fix the scale of L to 1 and learn a single scaling for both channels a and b; 2) for soft-segmentation (Section 4) we learn a single scaling for all 8 segmentation layers; 3) for optical flow (Section 5.2) we learn one parameter for the 2 flow channels, another for

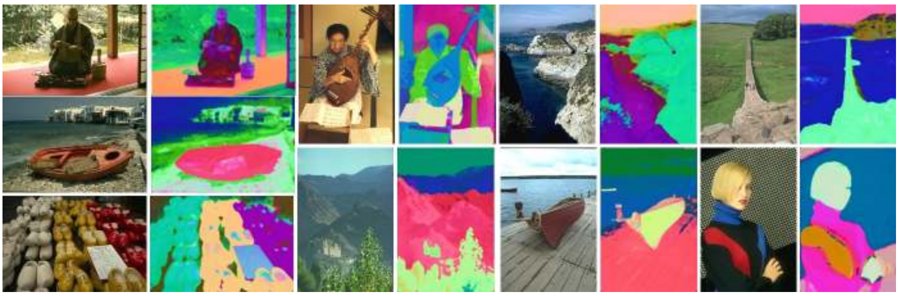

Fig. 5. Soft-segmentation results from our method. The first 3 dimensions of the soft-segmentations are shown on the RGB channels. Computation time for soft-segmentation is less than 2 seconds per 0.15 MP image in MATLAB.

the 2 channels of the unit normalized flow, and a third for the flow magnitude; 4) for RGB-D images (Section 5.3) we need one additional scaling for depth.

Learning the weights of layers is based on the observation that the matrix $\mathbf{M}$ can be written as a linear combination of matrices $\mathbf{M}_i$ computed for each scaling s_i separately:

$$\mathbf{M} = \sum_i s_i^2 \mathbf{M}_i, \tag{4}$$

where $\mathbf{M}_i \leftarrow (\mathbf{P}^\top \mathbf{X}_i)(\mathbf{P}^\top \mathbf{X}_i)^\top$ and $\mathbf{X_i}$ is the submatrix of $\mathbf{X}$, with the same number of rows as $\mathbf{X}$ and with columns corresponding only to those layers that are scaled by s_i. It follows that the largest eigenvalue of $\mathbf{M}$, $\lambda = \frac{1}{2}(\mathrm{tr}(\mathbf{M}) + \sqrt{\mathrm{tr}(\mathbf{M})^2 - \det(\mathbf{M})/4})$, can be computed from s_i and the elements of $\mathbf{M_i}$'s. Thus, the F-measure, which depends on (w_0, w_1) and λ, can also be computed over the training data as a function of the parameters (w_0, w_1) and s_i, which have to be learned. To optimize the F-measure, we use the direct search method of Lagarias et al. [21], since it does not require an analytic form of the cost and can be easily applied in MATLAB by using the `fminsearch` function. In our experiments, the positive and negative training edges were sampled at equally spaced locations on the output of Gb using only color, with all channels equally scaled (after non-maxima suppression applied directly on the raw $\sqrt{\lambda}$). Positive samples are the ones sufficiently close (less than 3 pixels) to the human-labeled ground truth boundaries.

4 An Efficient Soft-Segmentation Method

In this section we present a novel method to rapidly generate soft image segmentations. Its continuous output is similar to the eigenvectors computed by Ncuts [23], but its computational cost is significantly lower: under 2 sec (3.2 GHz CPU) vs. over 150 sec required for Ncuts (2.66 GHz CPU [22]) per 0.15MP image in MATLAB. We briefly describe it here because it serves as a fast mid-level representation of the image that significantly improves the boundary detection accuracy over raw color alone. While we describe this method in the context of color, we emphasize that it is general enough to integrate a variety of other image features, such as texture.

The method is motivated by the observation that regions of semantic interest (such as objects) can often be modeled with a certain, potentially complex, color distribution: each possible color has a certain probability of occurrence, given the region. Specifically, we assume that the colors of any image patch are generated from a distribution that is a linear combination of a finite number of color probability distributions belonging to the regions of interest in the image.

Let $\mathbf{c}$ be an indicator vector associated with some patch from the image, such that $c_i = 1$ if color i is present in the patch and 0 otherwise. If we assume that the image is formed by a composition of regions with colors generated from a few color distributions, then we can consider $\mathbf{c}$ to be a multi-dimensional random variable drawn from a mixture of distributions $\mathbf{h}_i$: $\mathbf{c} \sim \sum_i \pi_i \mathbf{h}_i$. The linear subspace of these distributions can be automatically learned by PCA applied to a the set of indicator vectors $\mathbf{c}$, sampled uniformly from the image. Once the subspace is discovered, for any patch P sampled from the image and its associated indicator vector $\mathbf{c}$, its generating distribution (considered to be the distribution of the foreground) can be obtained by PCA reconstruction: $\mathbf{h_F}(\mathbf{c}) \approx \mathbf{h_0} + \sum_\mathbf{i} (\mathbf{c} - \mathbf{h_0})^\top \mathbf{v_i}$. The distribution of the background is also obtained from the PCA model using the same coefficients, but with opposite sign: thus we obtain a background distribution that is as far as possible (in the subspace) from the foreground: $\mathbf{h_B}(\mathbf{c}) \approx \mathbf{h_0} - \sum_\mathbf{i} (\mathbf{c} - \mathbf{h_0})^\top \mathbf{v_i}$.

Having computed the figure/ground distributions, we classify whether each location in the image belongs to the same region as the current patch P. If we perform the same classification procedure for n_s (≈ 150) patches uniformly sampled on the image grid, we obtain n_s figure/ground segmentations for the same image. At a final step, we again perform PCA on vectors collected from all pixels in the image; each vector is of dimension n_s and corresponds to a certain image pixel, such that its i-th element is equal to the value at that pixel in the i-th figure/ground segmentation. Finally we use, for each image pixel, the coefficients of the first 8 principal dimensions to obtain a set of 8 soft-segmentations which represent a compressed version of the entire set of n_s segmentations. These soft-segmentations are used as input layers to our boundary detection method, and are similar in spirit to the normalized cuts eigenvectors computed for *gPb* [9]. In Figure 5 we show examples of the first three such soft-segmentations on the RGB color channels.

5 Experiments

To evaluate the generality of our proposed method, we conduct experiments on detecting boundaries in image, video and RGB-D data. First, we show results on static images using only color. Second, we perform experiments on occlusion boundary detection in short video clips. Multiple frames, closely spaced in time,

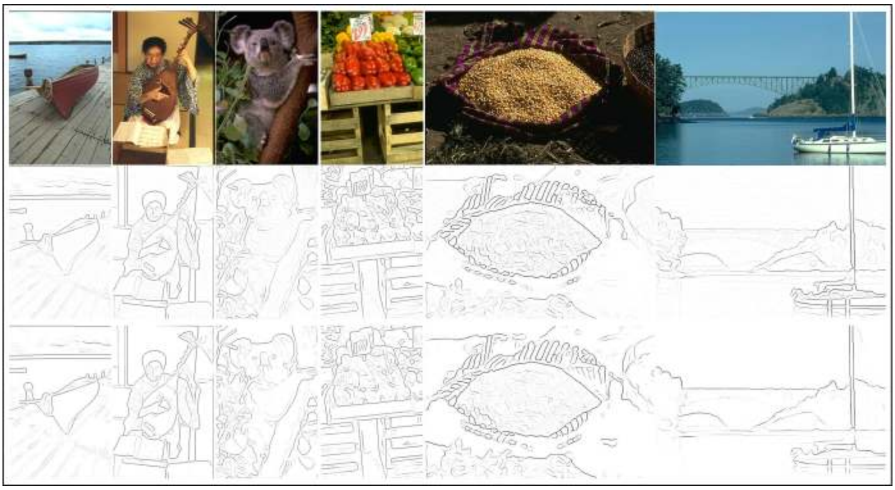

Fig. 6. Top row: input images from BSDS300 dataset. Middle row: output of Gb using only color layers. Bottom row: output of Gb using both color and our soft-segmentation.

provide significantly more information about dynamic scenes and make occlusion boundary detection possible, as shown in recent work [6–8, 24]. Third, we experiment with RGB-D video frames and show that depth can be effectively combined with color and optical flow to detect moving occlusion boundaries.

5.1 Boundaries in Static Color Images

We evaluate Gb on the well-known BSDS300 dataset [16] (Figure 6). We compare the accuracy and computational time of Gb with Pb [16], Gaussian derivatives (GD) for the gradient of multi-images [15], and Canny [4] edge detectors (Table 2). Canny uses brightness information, Gb and GD use brightness and color, whereas Pb uses brightness, color and texture information. Gb and GD use the same window size and Gaussian scale. For Gb we present two results, one using color (C), and the other using both color and soft-segmentation based on color (C+S). The total time reported for Gb (C+S) includes all processing: computing soft-segmentations and boundary detection. Even though Pb does not use segmentation we believe that our comparison is fair, since the total time for Gb (C+S) is more than 6 times faster than Pb in MATLAB. Also, Pb has the advantage of using learned textons, whereas Gb (C+S) uses only color. To test our model's robustness to overfitting we performed 30 different learning experiments for Gb (C+S) using 30 images randomly sampled from BSDS300 training set and obtained the same F-measure on the 100 images test set (measured $\sigma < 0.1\%$). Ren's method [17] obtains a higher F-measure of 0.68 on this dataset by combining the output of Pb at three scales, but the same multi-scale method could use

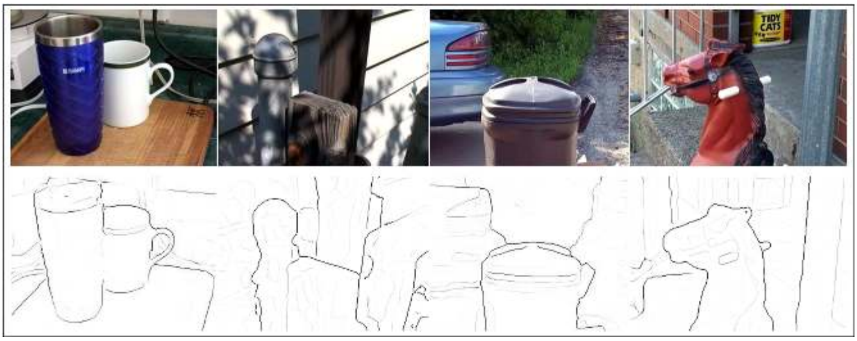

Fig. 7. Example boundary detection results on the CMU Motion Dataset

Gb instead. The state of the art global Pb [9, 22] achieves an F-measure of 0.70 by using Ncuts soft-segmentations. Our formulation is general and could easily incorporate better soft-segmentations as extra layers for improved performance. In fact, given a pool of figure/ground segments using CPMC [25], we obtained higher quality soft-segmentations by applying the same PCA reconstruction procedure from Section 4. This raised Gb's F-measure to 0.70 [26].

Table 2. Comparison of accuracy (F-measure) and total running time on BSDS. For Gb (C+S), the running time includes the computation of soft-segmentations.

Algorithm	Gb (C+S)	Gb (C)	Pb [16]	GD [15]	Canny [4]
F-measure	**0.67**	0.65	0.65	0.62	0.58
Total time (sec.)	3.0	0.5	19.5	0.3	0.1

5.2 Occlusion Boundaries in Video

State-of-the-art techniques for occlusion boundary detection in video are based on combining, in various ways, the outputs of existing boundary detectors for static color images with optical flow, followed by a global processing phase [6–8, 24]. Table 3 compares Gb against reported results on the CMU Motion Dataset [6] We use, as one of our layers, the flow computed using Sun et al.'s public code [27]. Additionally, Gb uses color and soft segmentation (Section 4). In contrast to the other methods [6–8, 24], which require significant time for processing and optimization, we require less than 1.6 seconds on average to process 230×320 images from the CMU dataset (excluding Sun et al.'s flow computation). Figure 7 shows qualitative results.

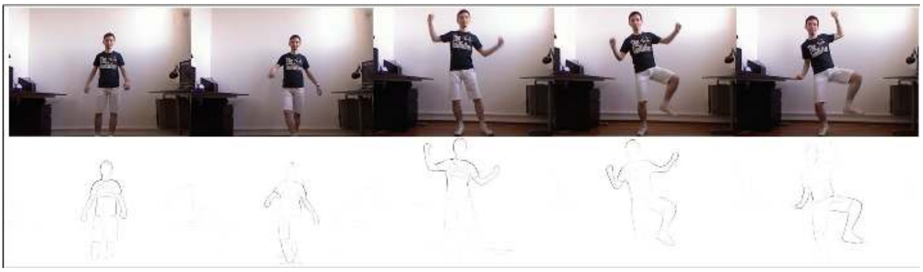

Fig. 8. Detecting occlusion boundaries in RGB-D by combining color, depth and flow.

5.3 Occlusion Boundaries in RGB-D Video

The third set of experiments uses RGB-D video of a moving person. We combine low-level color and depth input with large-displacement optical flow [28]. Figure 1 shows an example of the input layers and the output of our method. We learned this model's parameters from only three human-labeled images of silhouettes. Figure 8 shows qualitative results. Note that in a single formulation, Gb detects the moving occlusion boundaries and successfully learns to ignore most of the other ones.

Table 3. Occlusion boundary detection on the CMU Motion Dataset.

Algorithm	Gb	Sundberg et al. [7]	He & Yuille [8]	Sargin et al. [24]	Stein et al. [6]
F-measure	**0.62**	0.61	0.47	0.57	0.48

6 Conclusions

We present Gb, a novel model and algorithm for generalized boundary detection. Our method effectively combines multiple low-level and mid-level interpretation layers of an input image in a principled manner to achieve competitive results on standard datasets at a significantly lower computational cost than current methods. Gb's broad real-world applicability is demonstrated through qualitative and quantitative results on detecting boundaries in natural images, occlusion boundaries in video and moving object boundaries in RGB-D data.

Acknowledgements

This work was supported by CNCS-UEFICSDI, under PNII RU-RC-2/2009, PCE-2011-3-0438, and CT-ERC-2012-1.

References

1. Roberts, L.: Machine perception of three-dimensional solids. In: Optical and Electro-Optical Information Processing. MIT Press (1965) 159–197
2. Prewitt, J.: Object enhancement and extraction. In: Picture Processing and Psychopictorics. Academic Press, New York (1970) 75–149
3. Marr, D., Hildtreth, E.: Theory of edge detection. In: Proc. Royal Society. (1980)
4. Canny, J.: A computational approach to edge detection. PAMI **8** (1986) 679–698
5. Ruzon, M., Tomasi, C.: Edge, junction, and corner detection using color distributions. PAMI **23** (2001)
6. Stein, A., Hebert, M.: Occlusion boundaries from motion: Low-level detection and mid-level reasoning. IJCV **82** (2009)
7. Sundberg, P., Brox, T., Maire, M., Arbelaez, P., Malik, J.: Occlusion boundary detection and figure/ground assignment from optical flow. In: CVPR. (2011)
8. He, X., Yuille, A.: Occlusion boundary detection using pseudo-depth. In: ECCV. (2010)
9. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. PAMI **33** (2011)
10. Mairal, J., et al.: Discriminative sparse image models for class-specific edge detection and image interpretation. In: ECCV. (2008)
11. Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: ICCV. (2011)
12. Kanade, T.: Image understanding research at CMU. In: DARPA IUW. (1987)
13. Di Senzo, S.: A note on the gradient of a multi-image. CVGIP **33** (1986)
14. Cumani, A.: Edge detection in multispectral images. CVGIP **53** (1991)
15. Koschan, M., Abidi, M.: Detection and classification of edges in color images. Signal Processing Magazine, Special Issue on Color Image Processing **22** (2005)
16. Martin, D., Fawlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. PAMI **26** (2004)
17. Ren, X.: Multi-scale improves boundary detection in natural images. In: ECCV. (2008)
18. Meer, P., Georgescu, B.: Edge detection with embedded confidence. PAMI **23** (2001)
19. Baker, S., Nayar, S.K., Murase, H.: Parametric feature detection. In: DARPA Image Understanding Workshop. (1997)
20. Petrou, M., Kittler, J.: Optimal edge detectors for ramp edges. PAMI **13** (1991)
21. Lagarias, J., Reeds, J.A., Wright, M.H., Wright, P.E.: Convergence properties of the Nelder-Mead simplex method in low dimensions. SIAM Optimization **9** (1998)
22. Catanzaro, B., Su, B.Y., Sundaram, N., Lee, Y., Murphy, M., Keutzer, K.: Efficient, high-quality image contour detection. In: ICCV. (2009)
23. Shi, J., Malik, J.: Normalized cuts and image segmentation. PAMI **22** (2000)
24. Sargin, M., Bertelli, L., Manjunath, B., Rose, K.: Probabilistic occlusion boundary detection on spatio-temporal lattices. In: ICCV. (2009)
25. Carreira, J., Sminchisescu, C.: Constrained parametric min-cuts for automatic object segmentation. In: CVPR. (2010)
26. Leordeanu, M., Sukthankar, R., Sminchisescu, C.: Generalized boundaries from multiple image interpretations. In: Techincal Report, Institute of Mathematics of the Romanian Academy. (August 2012)
27. Sun, D., Roth, S., Black, M.: Secrets of optical flow estimation and their principles. In: CVPR. (2010)
28. Brox, T., Bregler, C., Malik, J.: Large displacement optical flow. In: CVPR. (2009)