Generalized Boundaries from Multiple Image Interpretations

Marius Leordeanu, Rahul Sukthankar, and Cristian Sminchisescu

Abstract—Boundary detection is a fundamental computer vision problem that is essential for a variety of tasks, such as contour and region segmentation, symmetry detection and object recognition and categorization. We propose a generalized formulation for boundary detection, with closed-form solution, applicable to the localization of different types of boundaries, such as object edges in natural images and occlusion boundaries from video. Our generalized boundary detection method (Gb) simultaneously combines low-level and mid-level image representations in a single eigenvalue problem and solves for the optimal continuous boundary orientation and strength. The closed-form solution to boundary detection enables our algorithm to achieve state of the art results at a significantly lower computational cost than current methods. We also propose two complementary novel components that can seamlessly be combined with Gb: first, we introduce a soft-segmentation procedure that provides region input layers to our boundary detection algorithm for a significant improvement in accuracy, at negligible computational cost; second, we present an efficient method for contour grouping and reasoning, which when applied as a final post-processing stage, further increases the boundary detection performance.

Index Terms—Edge, boundary and contour detection, occlusion boundaries, soft image segmentation, computer vision.

1 Introduction

Boundary detection is a fundamental computer vision problem with broad applicability in areas such as feature extraction, contour grouping, symmetry detection, segmentation of image regions, object recognition and categorization. Primarily, the task of *edge detection* has concentrated on finding signal discontinuities in the image that mark the transition from one region to another. Therefore, the majority of research on edge detection has focused on low-level cues, such as pixel intensity or color [3], [26], [36], [40], [41]. Recent work has started exploring the problem of *boundary detection* between meaningful scene objects or regions, based on higher-level representations of images, such as optical flow, surface and depth cues [13], [46], [49], segmentation [1], as well as object category specific information [12], [25].

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 (Fig. 11), and occlusion boundaries from optical flow (Figs. 14 and 15). We generalize the classical view of boundaries as sudden signal changes on the original low-level image input [3], [6], [7], [15], [26], [36], [40], to a locally linear (planar or step-wise) model on multiple layers of the input, computed over a relatively large image neighborhood. The layers can be viewed as interpretations of the image resulting from different visual process responses, 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).

Fig. 1. Gb combines different image interpretation layers (first three columns) to identify boundaries (right column) in a unified formulation. In this example Gb uses color, soft-segmentation and optical flow.

Despite the abundance of research on boundary detection, there is no general formulation of this problem that encompasses all types of boundaries, from intensity edges, to semantic regions, objects and occlusion discontinuities. In this paper, we make the popular but implicit intuition of boundaries explicit: boundary pixels mark the transition from one relatively constant region to another, under appropriate low- or high-level interpretations of the image. We 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 or visual process-

- *M. Leordeanu is with the Institute of Mathematics of the Romanian Academy (IMAR). E-mail: marius.leordeanu@imar.ro.*
- *R. Sukthankar is with Google Research and Carnegie Mellon. E-mail: rahuls@cs.cmu.edu.*
- *C. Sminchisescu is with Lund University and IMAR. E-mail: cristian.sminchisescu@math.lth.se. Correspondence should be addressed to all authors, who act as corresponding authors for this paper.*

Digital Object Indentifier 10.1109/TPAMI.2014.17

ing. 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 their position and orientation, with boundaries in other layers. For example, when discontinuities in intensity are correlated with discontinuities in optical flow, texture or other cues, the evidence for a relevant boundary is higher, with boundaries that align across multiple layers typically corresponding to the semantic boundaries that interest humans.

Based on these observations and motivated by the analysis of real world images (see Fig. 2), we develop a compact, integrated boundary model that ca simultaneously consider evidence from different input layers of the image, obtained from both lower and higher levels of visual processing.

Our contributions can be summarized as follows: **1)** We present a novel boundary model, operational over multiple image response layers, which can seamlessly incorporate inputs from visual processes, both low-level and high-level, static or dynamic. **2)** Our formulation provides an efficient closed-form solution that jointly computes the boundary strength and its normal by combining evidence from different input layers. This is in contrast with current approaches [1], [46], [49] that process the low and mid-level layers separately and combine them through multiple complex, computationally demanding stages, in order to detect different types of boundaries. **3)** We recover exact boundary normals through direct estimation rather than by evaluating a coarsely sampled set of orientation candidates [27]; **4)** We only have to learn a small set of parameters, which makes possible to perform efficient training with limited data. Our method bridges the gap between model fitting methods such as [2], [28], and recent successful, but computationally demanding learning-based boundary detectors [1], [46], [49]. **5)** We propose an efficient mid-level soft-segmentation method which offers effective input layers for our boundary detector and significantly improves accuracy at small computational expense (Sec. 6). **6)** We also present an efficient method for contour grouping and reasoning, which further improves the overall performance at minor cost (Sec. 7).

2 Relation to previous work

Our approach relates to both local boundary detectors and mid-level methods based on inference, grouping or optical flow. Here we briefly discuss how the existing literature relates to our work.

Local boundary detection. Classical approaches to edge detection are based on computing local first- or second-order derivatives on gray level images. Most of the early edge detection methods such as [36], [40], are based on the estimation of local first-order derivatives. Second-order spatial derivatives are employed in [26] in order to find edges as the zero crossings of the Laplacian of Gaussian operator. Other approaches use different local filters such as Oriented Energy-based [11], [29], [34] and the scale invariant approach [24]. A key limitation of derivatives is that their sensitivity to noise, stemming from their limited spatial support, can lead to high false positive rates.

Existing vector-valued techniques on multi-images [7], [15], [19] can be simultaneously applied to several channels, but are also limited to using local derivatives of the image. In the multi-channel case, derivatives have an additional limitation: even though true boundaries from one layer could coincide with those from a different layer, their location may not match perfectly — an assumption implicitly made by their restriction of having to perform computations over small local neighborhoods.

We argue that in order to confidently classify boundary pixels and robustly combine multiple layers of information, one must consider much larger neighborhoods, in line with recent methods [1], [27], [37]. A key 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 [1], [27] is to classify each possible 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 increases Pb's computational cost and limits orientation estimates to a particular angular quantization.

Mid-level boundary inference. True image boundaries tend to display certain grouping properties, such as proximity, continuity and smoothness, as observed by Gestalt theorists [31]. There are two main types of approaches that employ global mid-level grouping properties in order to improve boundary detection. The first focuses on grouping edges into contours with boundary detection performed by accumulating global information from such contours. The second, which is complementary, finds contours as boundaries of image regions from mid-level image segmentation.

A classical method that is based on contour grouping is Canny's algorithm [3], which links local edges into connected components thorough hysteresis thresholding. Other early approaches to finding long and smooth contours include [9], [32], [43], [52]. More recent methods formulate boundary detection in a probabilistic framework. JetStream [33] applies a multiple hypothesis probabilistic tracking approach to contour detection. Ren et al. [37] find contours with approximate MAP inference in conditional random fields based on constrained Delaunay

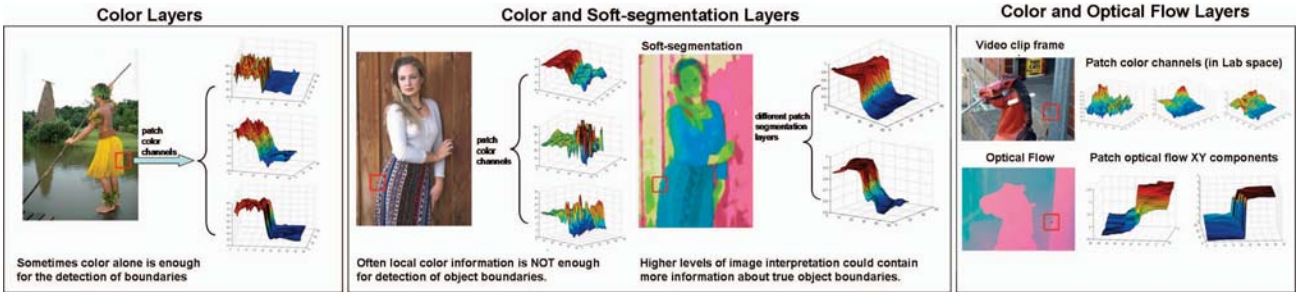

Fig. 2. Our step/ramp boundary model can be seen in different layers of real-world images. Left: a step is often visible in the low-level color channels. Middle: in some cases, no step is visible in the color channels yet the edge is clearly present in the output of a soft segmentation method. Right: in video, moving boundaries are often seen in the optical flow layer. More generally, a strong perceptual boundary at a given location may be visible in several layers, with consistent orientation across layers. Our multi-layer ramp model covers all these cases.

triangulation, relying on edges detected locally using Pb. Edge potentials are functions of Pb's response and the pairwise terms enforce curvilinear continuity. Felzenszwalb and McAllester [10], also use Pb and reformulate the MAP optimization of their graphical model over contours as a weighted min-cover problem, which they approximate with an efficient greedy algorithm. Zhu et al. [54] give an algebraic approach to contour detection using the complex eigenvectors of a random walk matrix over edges in a graph, with local responses again from Pb. Recent work based on Pairwise Markov Networks includes [17], [51].

The most representative approach that identifies edges as boundaries of image regions is global gPb [1]. That model computes local cues at three scales, based on Pb, and builds pairwise links between image pixels from intervening contours. The Ncut eigenvectors associated with the image graph represent soft segmentations whose local edges implicitly encode mid-level global grouping information. Combining these gPb cues with boosting and multiple instance learning were shown to further improve performance [16]. Another recent line of work with strong results [38] combines the gPb framework with sparse coded gradients learned on local patches.

Our work on mid-level inference exhibits conceptual similiarties to these state-of-the-art approaches but the methodology is substantially different. We employ a novel and computationally efficient soft-segmentation method (Sec. 6) as well as a fast contour reasoning method (Sec. 7). A key advantage of our soft-segmentation over using eigenvectors derived from normalized cuts is speed. We observe significant improvements in accuracy by employing such soft segmentations (rather than raw pixels) as input layers in the proposed Gb model. For contour grouping and reasoning (Sec. 7), like recent methods, we also consider curvilinear continuity constraints between neighboring edge pixels. However, instead of relying on expensive probabilistic graphical models or algebraic frameworks that may be difficult to optimize, we decompose the problem into several independent sub-problems that we can solve sequentially. First, we solve the contour grouping task by a connected component method that uses hysteresis thresholding in order to link only those edges that have a similar gradient orientation and are sufficiently close spatially. Local edge responses and orientation are rapidly computed using Gb. Second, we compute global cues from the contours that we have obtained and use them to re-score and classify individual pixels. The contour reasoning step is fast and significantly improves over Gb with color and soft-segmentation layers.

Occlusion boundaries in video. Occlusion detection in video is a relatively recent research area. By capturing the moving scene or through camera movement, one can accumulate evidence about depth discontinuities, in regions where the foreground object occludes parts of the background. State-of-the-art techniques for occlusion boundary detection in video [13], [42], [46], [49] use probabilistic graphical models to model occlusions. They combine the outputs of existing boundary detectors based on information extracted in color images with optical flow, and refine the estimates by means of a global processing step. Different from previous work, ours offers a unified model that can simultaneously consider evidence in all input layers (color, segmentation and optical flow) within a single optimization problem that enables exact computation of boundary strength and its normal.

3 Generalized Boundary Model

Given a $N_x \times N_y$ image I, let the 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, different filter responses, optical flow, or the output of a patch-based binary classifier trained to detect a specific color distribution, a texture pattern,

or a certain object category.[1] Thus, L_k could consist of relatively constant regions separated by boundaries.

We expect boundaries in different layers to not always align precisely. Given several such interpretation or measurement layers of the image, we wish to identify the most consistent boundaries across them. The output of Gb for each point $\mathbf{p}$ on the $N_x \times N_y$ image grid is a real-valued probability that $\mathbf{p}$ lies on a boundary, given the information in all image interpretations L_k centered at $\mathbf{p}$.

We model a boundary region 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 considered in resolving the current boundary detection problem, as multiple layers of interpretations of the image. Fig. 14 illustrates how we perform boundary detection by combining different layers, such as color, soft-segmentation and 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:

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

Here b_k is nonnegative and corresponds to the boundary "height" for layer k at location $\mathbf{p}_0$; $\pi_\epsilon(\mathbf{p})$ 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 $\pi_\epsilon(\mathbf{p}) = \mathbf{p}$, our model reduces to the first-order Taylor expansion of $L_k(\mathbf{p})$ around the current $\mathbf{p}_0$; however, as seen in our experiments, the regimes of small ϵ are the ones that lead to the best boundary detection performance.

As shown in Fig. 3, 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, along the boundary normal, that flattens as we move farther away from the center. 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 an ideal ramp (see Fig. 3), a property 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 [35]. Fig. 2 illustrates how boundaries found by our proposed model correspond to those visible in real-world images and video.

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

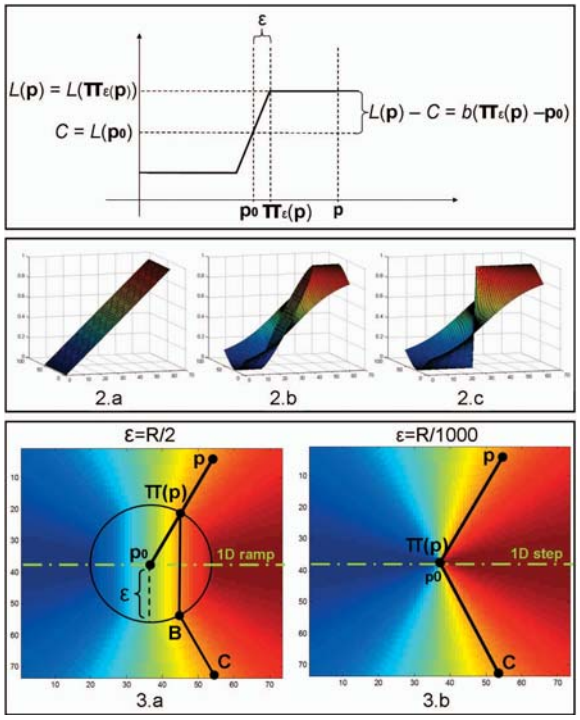

Fig. 3. Top: 1D view of our boundary model. Middle: 2D view of the model with different values of ϵ relative to the window radius R: 2a) $\epsilon > R$; 2b) $\epsilon = R/2$; 2c) $\epsilon = R/1000$. For small ϵ the boundary model is a step along the normal passing through the window center. Bottom: the model, for one layer, viewed from above: 3a) $\epsilon = R/2$; 3b) $\epsilon = R/1000$. The values on the path $[p, \pi(p), B, C]$ are the same. Inside the circle the model is planar and outside is radially constant. For small ϵ the radial line value ($[p_0, C]$) varies linearly with the cosine between that line and the boundary normal.

When the window is far from any boundary, the value of b_k should be near zero, since the only variation in the layer values is due to noise. When we are close to a boundary, b_k becomes large. The term $(\pi_\epsilon(\mathbf{p}) - \mathbf{p}_0)^\top \mathbf{n}(\mathbf{p}_0)$ 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 across 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, and constrained by all layers.

4 A Closed-Form Solution

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 $\pi_\epsilon(\mathbf{p}_i) - \mathbf{p}_0$ for the i-th point of the window. Let $\mathbf{n} = [n_x, n_y]$ be 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 Eq. 1, with unknowns $\mathbf{J}$ and $\mathbf{C}$ (we drop $\mathbf{p}_0$ to simplify notation):

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

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 above equation 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. Thus, we 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 the matrix $\mathbf{J}\mathbf{J}^\top = \|\mathbf{b}\|^2 \mathbf{n}^\top \mathbf{n}$ is symmetric and has rank 1. Then $\mathbf{n}$ can be estimated, in the least-squares sense, in terms of 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 $\mathbf{b} = [b_1, b_2, \ldots, b_K]$ and captures the overall strength of boundaries from all layers simultaneously. If the layers are properly scaled, then $\|\mathbf{b}\|$ can be used as a measure of boundary strength. Once we identify $\|\mathbf{b}\|$, we pass it through a 1D logistic model to obtain the probability of boundary, similar to recent methods [1], [27]. The parameters of the logistic model are learned using standard procedures, detailed in Sec. 5.3. The normal to the boundary $\mathbf{n}$ is then used for non-maximal suppression. Note that $\|\mathbf{b}\|$ is different from the gradient of multi-images [7], [15] or the single channel method of [30], which use second-moment matrices computed from local derivatives. In contrast, we compute the boundary by fitting a model, which, by controlling the window size and ϵ, ranges from planar to step-wise and accumulates information over a small or large patch.

Boundary strength along a given orientation. In some cases we might want to compute the boundary along a given orientation $\mathbf{n}$ (e.g., when the true normal is known *a priori*, or if needed for a specific task). One way to do it is to start from the observation $\mathbf{J}\mathbf{J}^\top = \|\mathbf{b}\|^2 \mathbf{n}^\top \mathbf{n}$ and estimate $\|\mathbf{b}\|^2$ as the minimizer q^* of the Frobenius norm $\|\mathbf{J}\mathbf{J}^\top - q\mathbf{n}^\top \mathbf{n}\|_F$. It is relatively easy to show that the optimal q^* is $\mathbf{vec}(\mathbf{J}\mathbf{J}^\top)^\top \mathbf{vec}(\mathbf{n}^\top \mathbf{n}) \frac{1}{\|\mathbf{n}^\top \mathbf{n}\|_F}$. Both $\mathbf{J}\mathbf{J}^\top$ and $\mathbf{n}^\top \mathbf{n}$ are symmetric positive semidefinite, so their Frobenius inner product $\mathbf{vec}(\mathbf{J}\mathbf{J}^\top)^\top \mathbf{vec}(\mathbf{n}^\top \mathbf{n}) = \mathrm{Tr}(\mathbf{J}\mathbf{J}^\top \mathbf{n}^\top \mathbf{n})$ is nonnegative. This follows from the property that the product of positive semidefinite matrices is also positive semidefinite and the trace of a matrix is equal to the sum of its eigenvalues. Thus, the optimal q^* is nonnegative and we can estimate $\|\mathbf{b}\| \approx \sqrt{q^*}$. The solution for q^* as a simple dot-product between two 4-element vectors provides a very fast procedure to estimate the boundary strength along any orientation, once matrix $\mathbf{M} = \mathbf{J}\mathbf{J}^\top$ is computed. This result, as well as the proposed closed-form solution, are made possible by our novel boundary model. In practice, computing the response of Gb over 8 quantized orientations is almost as fast as obtaining Gb based on the closed-form solution and has similar performance in terms of the F-measure. Our contour reasoning (Section 7) is also robust and performs equally well with quantized orientations.

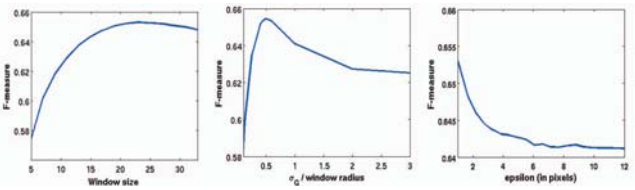

Fig. 4. Evaluation on the BSDS300 test set by varying the window size (in pixels), σ_G of the Gaussian weighting (relative to window radius) and ϵ. One parameter is varied, while the others are set to their optimum, learned from training images. Left: windows with large spatial support give a significantly 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, validating our step-wise model.

Gaussian weighting. We propose to weigh each pixel in a window by an isotropic 2D Gaussian located at the window center $\mathbf{p_0}$. Such a spatially weighting places greater importance on fitting the model to points closer to the window center.

The generalized boundary model is based on Eq. 2. The Gaussian weighting is applied such that the equation still holds, by multiplying each row of the matrices $\mathbf{X}$, $\mathbf{C}$, and $\mathbf{P}$ by the Gaussian weight applied to the corresponding location within the window. This is equivalent to multiplying each side of Eq. 2 with a diagonal matrix $\mathbf{G}$, having diagonal elements $G_{ii} = g(x_i - x_0, y_i - y_0)$, where g is the Gaussian weight applied at location $\mathbf{p}_i = (x_i, y_i)$ relative to the window center $\mathbf{p}_0 = (x_0, y_0)$. We can re-write the equation as:

$$\mathbf{G}\mathbf{X} = \mathbf{G}\mathbf{C} + \mathbf{G}\mathbf{P}\mathbf{J}. \tag{4}$$

The least squares solution for $\mathbf{J}$ in the above overdetermined system of equations is given by (to simplify notation, we denote $\mathbf{A} = \mathbf{G}\mathbf{P}$):

$$\mathbf{J} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{G}\mathbf{X} - (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{G}\mathbf{C}. \tag{5}$$

We observe that $(\mathbf{A}^\top \mathbf{A})^{-1} = ((\mathbf{G}\mathbf{P})^\top \mathbf{G}\mathbf{P})^{-1}$ is the identity matrix multiplied by some scalar $1/\alpha$, and

that $(\mathbf{GP})^\top \mathbf{GC} = (\mathbf{G^2P})^\top \mathbf{C} = \mathbf{0}$, since $\mathbf{G} = \mathbf{G}^\top$, matrix $\mathbf{C}$ has constant columns, and the columns of matrix $\mathbf{G^2P}$ sum to 0. It follows that

$$\mathbf{J} \approx \frac{1}{\alpha}(\mathbf{GP})^\top \mathbf{GX}. \quad (6)$$

Setting $\mathbf{X} \leftarrow \mathbf{GX}$ and $\mathbf{P} \leftarrow \mathbf{GP}$, we obtain the same expression for $\mathbf{J}$ as in Eq. 3, which can also be written as $\mathbf{J} \approx \frac{1}{\alpha}(\mathbf{G^2P})^\top \mathbf{X}$. To simplify notation, for the rest of the paper, we set $\mathbf{X} \leftarrow \mathbf{GX}$ and $\mathbf{P} \leftarrow \mathbf{GP}$, and use $\mathbf{X}$ and $\mathbf{P}$, instead of $\mathbf{GX}$ and $\mathbf{GP}$, respectively.

As seen in Fig. 4, the performance is influenced by the choice of Gaussian standard deviation σ_G, which supports our prior belief that points closer to the boundary should have greater influence on the model parameters. 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 [27] and were also near-optimal on the test set, as shown in Fig. 4. From these experiments, we draw the following conclusions regarding the proposed model: 1) A large window size leads to significantly better performance as more evidence can be integrated in reasoning about boundaries. Note that when the window size is small our model is related to methods based on local approximation of derivatives [3], [7], [15], [19]. 2) The usage of a small ϵ produces boundaries with significantly better localization and strength. It strongly suggests that perceptual boundary transitions in natural images tend to be sudden, rather than gradual. 3) The center-weighting is justified: the model is better fitted if more weight is placed on points closer to the putative boundary.

5 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 [20] to optimize the F-measure (Sec. 5.3). Alg. 1 (Gb) summarizes the proposed approach.

Algorithm 1 Gb: Generalized Boundary Detection

Initialize $\mathbf{L}$, with each layer 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 $\mathbf{b}$, θ

The pseudo-code presented in Alg. 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 convolution, as explained next. $\mathbf{X}$ contains values from the input layers, restricted to a particular window, and matrix $\mathbf{J}$ is computed for each window location. Using Eq. 6 and observing that matrix $\mathbf{G^2P}$ does not depend on the window center $\mathbf{p_0} = (x_0, y_0)$, the elements of $\mathbf{J}$ can be computed, for all window locations in the image, by convolving each layer L_k twice, using two filters: $\mathbf{H}_x(x-x_0, y-y_0) \propto g(x-x_0, y-y_0)^2(x_\epsilon - x_0)$ and $\mathbf{H}_y(x-x_0, y-y_0) \propto g(x-x_0, y-y_0)^2(y_\epsilon - y_0)$, where (x, y) is $\mathbf{p}$ and (x_ϵ, y_ϵ) is $\pi_\epsilon(\mathbf{p})$. Specifically, $\mathbf{J_{p_0}}(k, 1) = (\mathbf{L}_k * \mathbf{H}_x)(x_0, y_0)$ and $\mathbf{J_{p_0}}(k, 2) = (\mathbf{L}_k * \mathbf{H}_y)(x_0, y_0)$. Then $\mathbf{M} = \mathbf{JJ}^\top$ can be immediately obtained, for any given $\mathbf{p}_0$. These observations result in an easy-to-code, filtering-based implementation of Gb.[2]

5.1 Relation to Filtering-based Edge Detection

There is an interesting connection between the filters used in Gb (e.g., $\mathbf{H}_x \propto g(x-x_0, y-y_0)^2(x_\epsilon - x_0)$) and Gaussian Derivative (GD) 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 [7]. Since the squared Gaussian $g(x-x_0, y-y_0)^2$ from $\mathbf{H}$ is also Gaussian, the main analytic difference between the two filters lies in our introduction of the projection function $\pi_\epsilon(\mathbf{p})$. For an ϵ that is at least as large as the window radius, the two filters are the same, which means that edge detection with Gaussian Derivatives is equivalent to fitting a linear (planar in 2D) edge model (Fig. 3) with Gaussian weighted least-squares. From this point of view Gb filters could be seen as a generalization of Gaussian Derivatives.

Fig. 5 presents the Gb and GD filters from two different view-points. Gaussian derivatives have the computational advantage of being separable. On the other hand, Gb filters with small ϵ are better suited for real world, perceptual boundaries (see Fig. 2): they are steep perpendicular to the boundary, with a pointed shape along the boundary. This allows a better handling of corners, as seen in the example given in Fig. 5, bottom row. In practice, small ϵ gives significantly better results over large ϵ, both qualitatively and quantitatively: on $BSDS300$ the F-measure drops by about 1.5% when ϵ = window radius is used, with all other parameters being optimized (Fig. 4).

Our approach differs from traditional filtering based edge detectors [18] in the following ways: 1) Gb filters not only the image but also other layers of the image, resulting from different visual processes, low-level or high-level, static or dynamic, such as soft-segmentation or optical flow; 2)Gb boundaries are not computed directly from filter responses, but only after

2. Code available at: `http://www.imar.ro/clvp/code/Gb`, as well as `sites.google.com/site/gbdetector/`, and `http://www.maths.lth.se/matematiklth/personal/sminchis/code/index.html`.

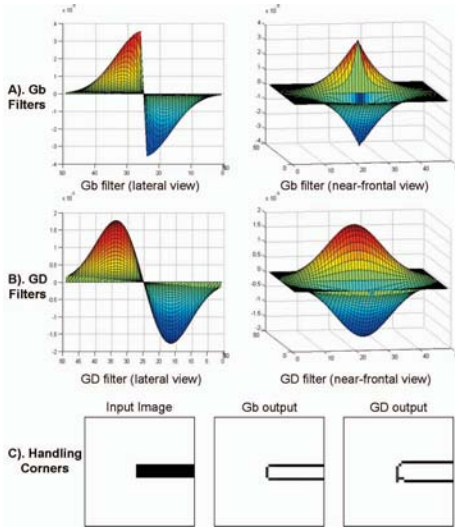

Fig. 5. A. Gb filters. B. Gaussian Derivative (GD) filters C. Output (zoomed in) of Gb and GD (after non-maximal suppression) for a dark line (5 pixels thin), having identical parameters (except for ϵ) and window size $= 19$ pixels. GD has poorer localization and corner handling. Note: asymmetry in GD output is due to numerical issues in the non-maximal suppression.

building matrix $\mathbf{M} = \mathbf{J}\mathbf{J}^\top$ and computing its principal eigenpair (Alg. 1). Gb provides a potentially revealing connection between model fitting and filtering-based edge detection.

5.2 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 is 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 [1], [27]. 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 for each orientation. In Fig. 6, we show the run times for Gb and Pb (based on publicly available code) on a 3.2 GHz desktop. These are MATLAB implementations, run 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 1–2 orders of magnitude faster than Pb (about 40×), independent of the image size (Fig. 6, right). 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.

TABLE 1
Run times: Gb in MATLAB (without using mex files) on a 3.2 GHz desktop vs. Catanzaro et al.'s parallel computation of local cues on Nvidia GTX 280 [5].

Algorithm	Gb (exact)	[5] (exact)	[5] (approx.)
Run time (sec.)	0.473	4.0	0.569

A parallellized implementation of gPb is proposed in [5], where method is implemented directly on a high-performance Nvidia GTX 280 graphics card with 240 CUDA cores. 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 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 [5], but we estimate that it is in the order of 1000 per input image (4 channels $\times$ 8 orientations $\times$ 32 histogram bins $= 1024$). The approximation also requires special processing of the rotated integral images of texton labels, to minimize interpolation artifacts. The authors propose a solution based on Bresenham lines, which may also, to some degree impact the discretization of the rotation angle. In Table 1 we present run time comparisons with Pb's local cues computation from [5]. 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.

5.3 Learning

Our model uses a small number of parameters. Only two parameters (w_0, w_1) are needed for the logistic function that models the probability of boundary (Alg. 1). The role of these parameters is to strengthen or weaken the output, but they do not affect the quantitative performance since the logistic function is monotonically increasing in the eigenvalue of $\mathbf{M}$, λ. Instead, the parameters only affect the F-measure for a fixed, desired threshold. 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

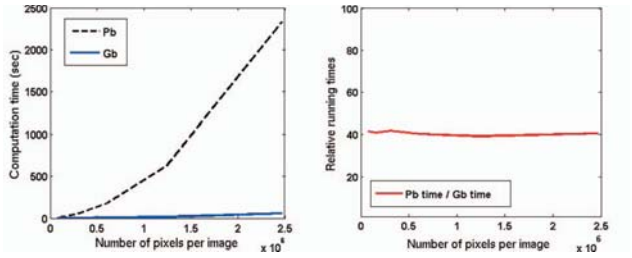

Fig. 6. Left: Edge detection run times on a 3.2 GHz desktop for our MATLAB-only implementation of Gb vs. the publicly available code of Pb [27]. Right: ratio of run time of Pb to run time of Gb, experimentally confirming that Pb and Gb have the same time complexity, but Gb has a significantly lower fixed cost per iteration. 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$\times$ faster.

single scaling for both channels a and b; 2) for soft-segmentation (Sec. 6) we learn a single scaling for all 8 segmentation layers; 3) for optical flow (Sec. 8.2) we learn one parameter for the 2 flow channels, another for the 2 channels of the unit normalized flow, and a third for the flow magnitude.

Learning layer scaling is based on the observation that $\mathbf{M}$ is as a linear combination of matrices $\mathbf{M}_i$ computed separately for each layer scaling s_i:

$$\mathbf{M} = \sum_i s_i^2 \mathbf{M}_i, \tag{7}$$

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's 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. [20], since it does not require an analytic form of the cost and can be easily implemented 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-maximal suppression applied directly on the raw $\sqrt{\lambda}$). Positive samples are the ones sufficiently close (< 3 pixels) to the human-labeled ground truth boundaries.

6 Efficient Soft-Segmentation

In this section we present a novel method to rapidly generate soft image segmentations. Its continuous output is similar to the Ncuts eigenvectors [44], but its computational cost is significantly lower: about 2.5 sec. (3.2 GHz CPU) vs. over 150 sec. required for Ncuts (2.66 GHz CPU [5]) per 0.15 MP image in MATLAB (no mex files). 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.

We assume that the color of any image pixel has a certain probability of occurrence, given the semantic region (e.g., object) to which it belongs -the image is formed by a composition of semantic regions with distinct color distributions, which are location independent given the region. Thus, colors of any image patch are generated from a certain, patch-dependent, linear combination (mixture) of these finite number of distributions: if the patch is from a single region then it will have a single generating distribution; if the patch is in between regions then it will be generated from a mixture of distributions depending on the patch location relative to those regions. Let $\mathbf{c}$ be an indicator vector of some image patch, such that $c_j = 1$ if color j is present in the patch and 0 otherwise. Then $\mathbf{c}$ is a multi-dimensional Bernoulli random variable drawn from its mixture: $\mathbf{c} \sim \sum_i \pi_i(\mathbf{c})\mathbf{h}_i$.

Based on this model, the space of all $\mathbf{c}$'s from a given image will contain redundant information, reflecting the regularity of real-world scenes through the underlying generative distributions. We *discover* the linear subspace of these distributions, that is its *eigendistributions* $\mathbf{v_i}$'s, by applying PCA to a sufficiently large set of indicator vectors $\mathbf{c}$ sampled uniformly from the image. Then, for any given patch, the generating *foreground* distribution of its associated indicator vector $\mathbf{c}$ could be approximated by means of PCA reconstruction: $\mathbf{h_F}(\mathbf{c}) \approx \mathbf{h}_0 + \sum_i ((\mathbf{c} - \mathbf{h}_0)^\top \mathbf{v}_i)\mathbf{v}_i$. Here $\mathbf{h}_0$ is the sample mean, the overall empirical color distribution of the whole image.

We consider the *background* distribution to be one that is as far as possible (in the subspace) from the foreground, by using the same coefficients but with opposite sign: $\mathbf{h_B}(\mathbf{c}) \approx \mathbf{h}_0 - \sum_i ((\mathbf{c} - \mathbf{h}_0)^\top \mathbf{v}_i)\mathbf{v}_i$. Then $\mathbf{h_F}(\mathbf{c})$ and $\mathbf{h_B}(\mathbf{c})$ are used to obtain the foreground (F) posterior probability for each image pixel i, based on its color x_i, by applying Bayes' rule with equal priors:

$$P^{(\mathbf{c})}(\mathbf{F}|x_i) = \frac{\mathbf{h}_\mathbf{F}^{(\mathbf{c})}(x_i)}{\mathbf{h}_\mathbf{F}^{(\mathbf{c})}(x_i) + \mathbf{h}_\mathbf{B}^{(\mathbf{c})}(x_i)} \approx \frac{\mathbf{h}_\mathbf{F}^{(\mathbf{c})}(x_i)}{2\mathbf{h}_0}. \tag{8}$$

Given an image patch, we quickly obtain a posterior probability of foreground (F) for each image pixel, resulting in a soft figure/ground segmentation (Fig. 9). These figure/ground segmentations are similar in spirit to the segmentation *hints* based on alpha matting [23], used by Stein et al. [47] for full object segmentation. The figure/ground segmentations are often redundant when different patches are centered at different locations on the same object–a direct result of the first stage, when a reduced subspace for color distributions is learned. Thus, many of such soft

Fig. 7. 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 ≈2.5 seconds per 0.15 MP image in MATLAB.

figure/ground probability maps can be compressed to obtain a few representative soft figure/ground segmentations of the same image, as detailed next.

We perform the same classification procedure for n_s (≈ 70) patches uniformly sampled on a regular image grid and obtain n_s figure/ground segmentations. We compress this set of soft-segmentations by performing (a different, second level) 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 soft figure/ground map. Finally, we use, for each image pixel, the coefficients of its first 8 principal dimensions to obtain a set of 8 soft-segmentations. These soft-segmentations are used as input layers to our boundary detection method. Figs. 7 and 8 show examples of the first three such soft-segmentations on the RGB color channels. Our method is much faster (one to two orders of magnitude) than computing the Ncuts eigenvectors previously used for boundary detection [1] and provides a useful mid-level representation of the image that can significantly improve boundary detection. It has also been incorporated into efficient segmentation-aware descriptors [50].

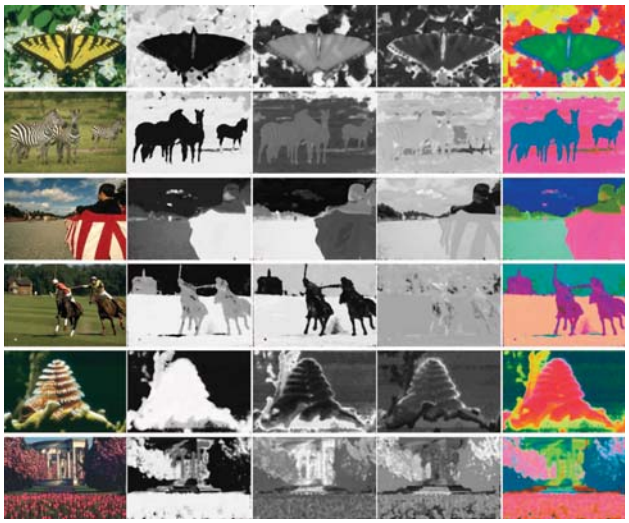

Fig. 8. Soft-segmentation results obtained using our method. First column: input image. Columns 2–4: the first 3 dimensions of our soft-segmentations, shown separately. Last column: all 3 dimensions shown together on the RGB channels.

7 Contour Grouping and Reasoning

Pixels that belong to true boundaries tend to form long smooth contours that obey Gestalt grouping principles such as continuity and proximity. By linking edges into contours and considering different properties of these contours we can re-evaluate the probability of boundary at each pixel and further improve the boundary detection accuracy. The idea is intuitive: individual pixels from strong contours (long, smooth and with high boundary strength) are more likely to belong to true boundaries than noisy edges that cannot be grouped along such contours.

Our approach to using contours for boundary detection is the following (Fig. 10): first, find contours by linking edges, for which we use our earlier approach from [21] (Sec. 7.1); second, for each edge pixel from a given contour, re-evaluate its probability of boundary by considering its own local boundary strength (given by the Gb algorithm) together with different geometric and appearance properties of its corresponding contour, such as: length, smoothness, average and maximum boundary strength (Sec. 7.2).

7.1 Contour grouping

We group the edges into contours by using a method very similar to [21]. First, we form connected components by linking pairs of boundary pixels (i, j) that are both sufficiently close (i.e., adjacent within 1.5 pixels) and satisfy collinearity and proximity constraints,

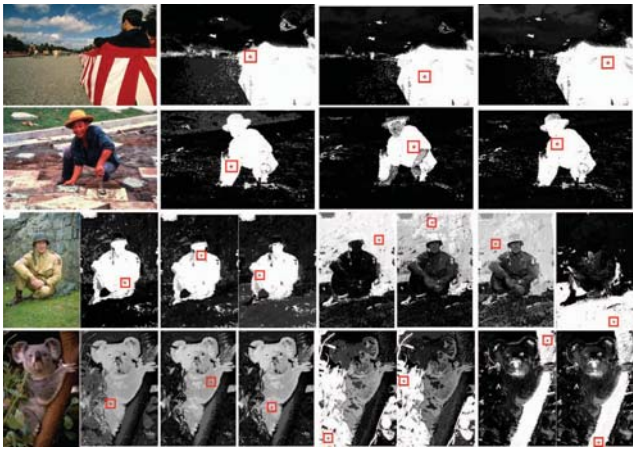

Fig. 9. Examples of soft figure/ground segmentations based on the current patch, shown in red. Note that the figure/ground soft-segmentations are similar for patches centered on different locations of the same object; this justifies our final PCA compression.

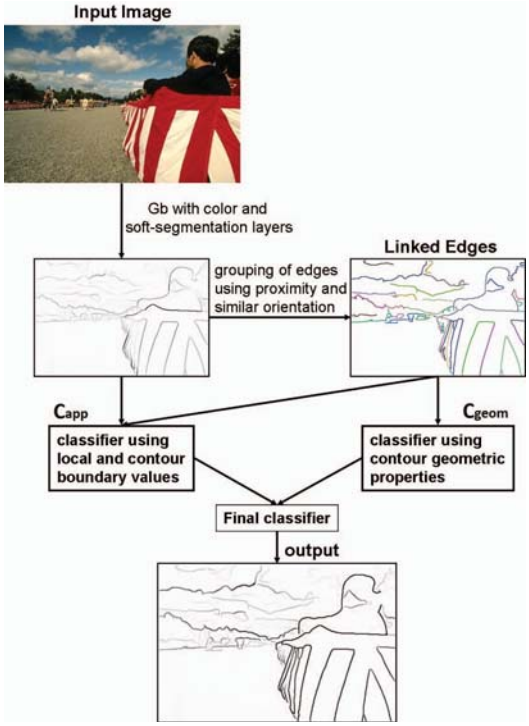

Fig. 10. Overview of the contour reasoning framework. First, an edge map is obtained from the input image using a Gb model with color and soft segmentation layers. Then edges are linked to form contours. Next, using two logistic classifiers we label the edge pixels based on different properties of their contours: appearance (boundary strength) or geometry (length and smoothness). The outputs of these two classifiers are then combined to obtain the final boundary map.

ensuring that the components only contain smooth contours. For each connected component c we form its weighted adjacency matrix $\mathbf{A}$ such that A_{ij} is positive if edge pixels (i, j) are connected and 0 otherwise:

$$A_{ij} = \begin{cases} 1 - \frac{\theta_{ij}^2}{\sigma_\theta^2} & \text{if } (i,j) \text{ are neighbors and } \theta_{ij} < \sigma_\theta \\ 0 & \text{otherwise,} \end{cases}$$

where $\theta_{ij} \geq 0$ is the smallest (positive) angle between the boundary normals at pixels i and j and σ_θ is a predefined threshold. The value of A_{ij} increases with the similarity between the boundary normals at neighboring pixels. Therefore, smooth contours have larger average values in their adjacency matrix $\mathbf{A}$.

Let p be the current pixel and $c(p)$ the label of its contour. The following two geometric cues are computed for each contour (used by the contour-based boundary classification method, explained in Section 7.2): 1). the contour length, computed as the number of pixels of component $c(p)$ normalized by the length of the image diagonal; 2). the average contour smoothness estimated as the sum of elements in $\mathbf{A}_{c(p)}$ divided by the length of $c(p)$.

7.2 Contour reasoning

Our classification scheme has two stages, with a structure similar to a two-layer neural network, having logistic linear classifiers at all nodes, both hidden and final output (Figure 10). The main difference is in the training and its design: each node and connection is chosen manually and training is performed sequentially, bottom-up, from the first level to the last.

At the first level in the hierarchy, we first train a *geometry-only* logistc boundary classifier C_{geom} (using standard linear logistic regression) applied to each contour pixel using two features: the length and average smoothness of the contour fragment (computed as explained in Sec. 7.1). Second, also at the first level, we train an *appearance-only* logistic edge classifier C_{app} (again applied to each contour pixel, trained by linear logistic regression) using the following three cues: local boundary strength at the current pixel, average and maximum boundary strength over the contour. The soft outputs of these two classifiers become inputs to the final linear logistic boundary classifier, at the second level in the hierarchy, which is also trained using logistic regression. The separate training of each classifier is performed due to its efficiency, but more sophisticated learning methods could also be employed for fine-tuning parameters. The framework (Fig. 10) is related to late fusion schemes from semantic video analysis and indexing [45], [53], in which separate independent classifiers are trained and their outputs are then combined using a second-level classifier. The steps of our contour grouping and reasoning algorithm are:

1) Run the Gb method described in Alg. 1 with non local maximal suppression to obtain thin edges.

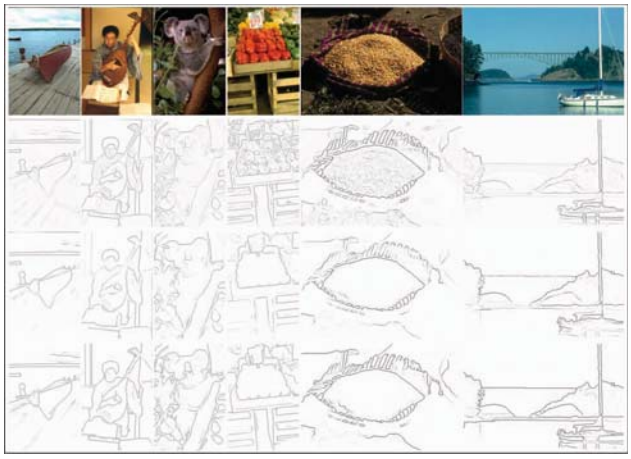

Fig. 11. Top row: input images from BSDS300 dataset. Second row: output of a Gb model that uses only color layers. Third row: output of a Gb model that uses both color and soft-segmentation. Bottom row: output of a more complex Gb model that leverages color, soft segmentation and contour reasoning.

2) Remove the edge pixels with low boundary strength.
3) Group the surviving edges into contours using the method from Sec. 7.1.
4) For each contour pixel compute the probability of boundary using the *geometry-only* logistic classifier C_{geom}.
5) For each contour pixel compute the probability of boundary using the *appearance-only* logistic classifier C_{app}.
6) For each contour pixel compute the final probability of boundary with a logistic classifier that combines the outputs of C_{geom} and C_{app}.

8 Experiments

To evaluate the generality of our proposed method, we conduct experiments on detecting boundaries in both images and video. First, we show results on static images. Second, we perform experiments on occlusion boundary detection in short video clips.

8.1 Boundaries in Static Color Images

We evaluate Gb on the well-known BSDS300 dataset [27] (Fig. 11). We compare the accuracy and computational time of Gb with other published methods (see Table 2). For Gb we present results using color (C), color and soft-segmentation (C+S), and color, soft-segmentation and contour grouping (C+S+G). We also include results on gray-scale images. The total times reported for Gb include all processing needed (MATLAB-only, without compiled mex files): for example, for Gb(C+S+G) the reported

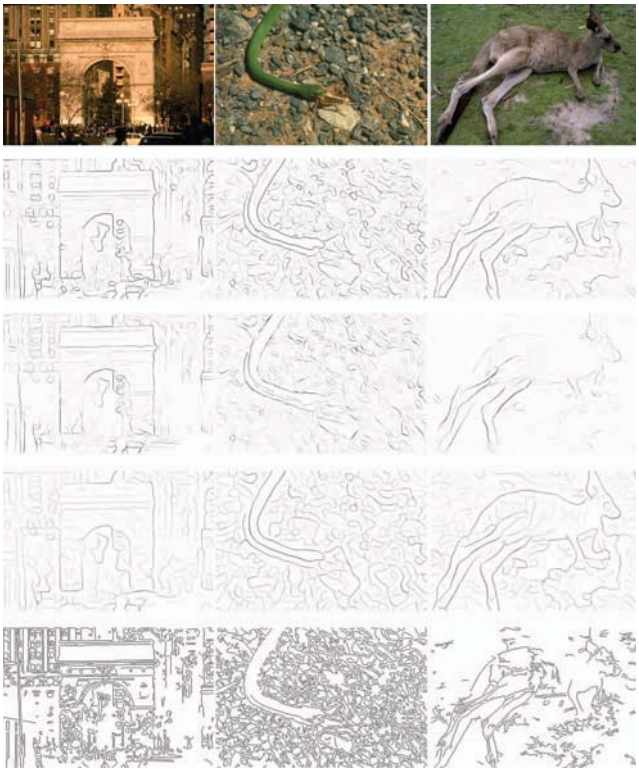

Fig. 12. Qualitative boundary detection results for images from BSDS300 (first row), obtained with a Gb model that uses only color layers (second row), Pb (third row), GD (fourth row), and Canny (last row).

times include computing soft-segmentations, boundary detection (Alg. 1) and contour reasoning. Gb achieves a competitive F-measure of 0.69 very fast, compared to current state of the art techniques. For example, the method of [37] obtains an F-measure of 0.68 on this dataset by combining the output of Pb at three scales. Note that the same multi-scale method could use Gb instead, which can potentially improve the overall performance of our approach. Global Pb [1], [5] achieves an F-measure of 0.70 by using the significantly more expensive Ncuts soft-segmentations. Note that our formulation is general and could incorporate other segmentations (such as Ncuts, CPMC [4], or compositional methods [14]).

Our proposed Gb is competitive even when using only color layers alone at a processing speed of 0.5 sec. per image in pure Matlab. In Fig. 12 we present a few comparative results of four different local, single scale boundary detectors: Gb using only color, Pb [27], Gaussian derivatives (GD) for the gradient of multi-images [19], and Canny [3] edge detectors (Table 2). Canny uses brightness information, Gb and GD use brightness and color and Pb uses brightness, color and texture. Gb, Pb and GD use the same window size.

The benefit from soft-segmentation: In Fig. 13 we present the output of Gb using only the first 3 di-

Fig. 13. Boundary detection examples, obtained using a Gb model that uses only the first 3 dimensions of our soft-segmentations as input layers. Note: color layers were *not* used here.

mensions of our soft-segmentations as input layers (no color information was used). We came to the following conclusions: 1) while soft-segmentations do not separate the image into disjoint regions (as hard-segmentation does), their boundaries are correlated especially with occlusions and whole object boundaries (as also confirmed by our results on CMU Motion Dataset [46]); 2) soft-segmentations cannot capture the fine details of objects or texture, but, in combination with raw color layers, they can significantly improve Gb's performance on detecting general boundaries in static natural images.

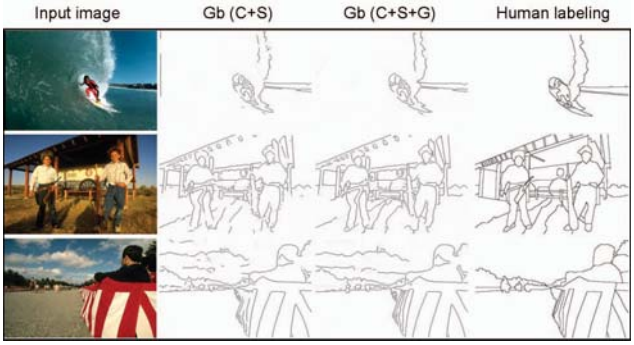

Fig. 14. Output of a Gb model using color and soft segmentation layers, without contours (second column) and with contours (third column) after thresholding at the optimal F-measure. The use of global contour reasoning produces a cleaner output.

The benefit from contour reasoning: Besides the solid improvement of 2% in F-measure over Gb(C+S) and Gb (graylevel), the contour reasoning module (denoted by G in Table 2), which runs in less than 0.5 sec. in MATLAB per 0.15 MP image, brings the following qualitative advantage (see also Fig. 14): during this last stage, edge pixels belonging to the same contour tend to end up with very similar probabilities of boundaries. This outcome is intuitive, since pixels belonging to one contour should either be accepted or rejected together. Thus, for any given decision threshold, the surviving edge map will look clean, with very few isolated edge pixels. The qualitative improvement after contour reasoning is visually striking after cutting by the optimal threshold (see Fig. 14).

TABLE 2
Comparison of F-measure and total runtime in MATLAB. For Gb (C+S+G) it includes the computation of soft-segmentations (S) and contour reasoning (G).

Algorithm	F-measure	Total run time (sec.)
Gb (C+S+G)	**0.69**	**6.0**
Gb (C+S)	0.67	5.5
Gb (C)	0.65	0.5
Gb (graylevel+G)	0.63	0.2
Gb (graylevel)	0.61	0.15
SCG Ren and Bo [38]	**0.715**	> 175
gPb - Arbelaez et al. [1]	0.70	175
Multiscale - Ren [37]	0.68	60
Mairal et al. [25]	0.66	NA
BEL - Dollar et al. [8]	0.66	NA
Felzenszwalb et al. [10]	0.65	NA
CRF - Ren et al. [39]	0.64	NA
Pb - Martin et al. [27]	0.65	20
GD (C) [19]	0.62	0.3
GD (graylevel) [19]	0.58	0.1
Canny (graylevel) [3]	0.58	0.1

To test our model's robustness to overfitting we performed 30 different learning experiments for Gb (C+S) using 30 images randomly sampled from the BSDS300 training set. As a result, we obtained the same F-measure on the 100 images test set (measured $\sigma < 0.1\%$), confirming that the representation and the parameter learning procedure are robust.

In Fig. 12 we present some additional examples of boundary detection. We show boundaries detected with Gb(C), Pb [27], GD and the Canny edge detector [3]. Both Gb and GD use only color layers, identically scaled, with the same window size and Gaussian weighting. GD is based on the gradient of multi-images [19], which we computed using Derivative of Gaussian filters. While in classical work on edge detection from multi-images [7], [19] the channel gradients are computed over small image neighborhoods, in our paper we use Derivative of Gaussian filters of the same size as the window applied to Gb, for a fair comparison. Note that Pb uses color and texture, while Canny is based only on brightness with derivatives computed at a fine scale. For Canny we used the MATLAB function *edge* with thresholds $[0.1, 0.25]$. Run-times in MATLAB-only per image, for

Fig. 15. Gb results on the CMU Motion Dataset.

each method, are: Gb - 0.5 sec., Pb - 19.4 sec., GD - 0.3 sec., and Canny - 0.1 sec.

These examples confirm, once again, that: 1) Gb models that use only color layers produce boundaries that are of similar quality as Pb's; 2) methods based on local derivatives, such as Canny, cannot produce high quality boundaries on difficult, highly-textured, color images; and 3) using Gaussian Derivatives with a large standard deviation could remove noisy edges (reduce false positives) and improve boundary strength, but at the cost of poorer localization and detection (increases the false negative rate). Note that Gaussian smoothing suppresses the high-frequencies, which are important for boundary detection. In contrast, our generalized boundary model, with a regime of small ϵ, is more sensitive in localization and correctly captures the general notion of boundaries as *sudden* transitions among different image regions.

8.2 Occlusion Boundaries in Video

Multiple video frames, closely spaced in time, provide significantly more information about dynamic scenes and make occlusion boundary detection possible, as shown in recent work [13], [42], [46], [49]. 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 [13], [42], [46], [49]. Table 3 compares Gb against reported results on the CMU Motion Dataset [46] We use, as one of our layers, the flow computed using Sun et al.'s public code [48]. Additionally, Gb uses color and soft segmentation (Sec. 6), as described in the previous sections. In contrast to other methods [13], [42], [46], [49], which require significant time for processing and optimization, we require less than 1.6 sec. on average to process 230×320 images from the CMU dataset (excluding Sun et al.'s flow computation). Fig. 15 shows qualitative results.

TABLE 3
Occlusion boundary detection, using a Gb model with optical flow layer, on the CMU Motion Dataset.

Algorithm	F-measure
Gb	**0.62**
Sundberg et al. [49]	0.61
He & Yuille [13]	0.47
Sargin et al. [42]	0.57
Stein et al. [46]	0.48

9 Conclusions

We have presented Gb, a novel model and algorithm for generalized boundary detection. Gb effectively combines multiple low- and mid-level interpretation layers of an image in a principled manner, and resolves their constraints jointly, in closed-form, in order to compute the exact boundary strength and orientation. Consequently, Gb achieves state of the art results on published datasets at a significantly lower computational cost than current methods. For mid-level inference, we present two efficient methods for soft-segmentation, and contour grouping and reasoning, which significantly improve the boundary detection performance at negligible computational cost. Gb's broad real-world applicability is demonstrated through quantitative and qualitative results on the detection of boundaries in natural images and the identification of occlusion boundaries in video.

Acknowledgments

The authors would like to thank Andrei Zanfir for helping with parts of the code. This work was supported in part by CNCS-UEFICSDI, under PNII RU-RC-2/2009, PCE-2011-3-0438, PCE-2012-4-0581, and CT-ERC-2012-1.

References

[1] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5), 2011.
[2] S. Baker, S. K. Nayar, and H. Murase. Parametric feature detection. In *DARPA IUW*, 1997.
[3] J. Canny. A computational approach to edge detection. *PAMI*, 8(6), 1986.
[4] J. Carreira and C. Sminchisescu. CPMC: Automatic object segmentation using constrained parametric min-cuts. *PAMI*, 34(7), 2012.
[5] B. Catanzaro, B.-Y. Su, N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer. Efficient, high-quality image contour detection. In *ICCV*, 2009.
[6] A. Cumani. Edge detection in multispectral images. *CVGIP*, 53(1), 1991.
[7] S. Di Senzo. A note on the gradient of a multi-image. *CVGIP*, 33(1), 1986.
[8] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, 2006.
[9] J. Elder and S. Zucker. Computing contour closures. In *ECCV*, 1995.
[10] P. Felzenszwalb and D. McAllester. A min-cover approach for finding salient curves. In *CVPRW*, 2006.
[11] W. T. Freeman and E. H. Adelson. The design and use of steerable filters. *PAMI*, 13(9), 1991.
[12] B. Hariharan, P. Arbelaez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, 2011.
[13] X. He and A. Yuille. Occlusion boundary detection using pseudo-depth. In *ECCV*, 2010.
[14] A. Ion, J. Carreira, and C. Sminchisescu. Image segmentation by figure-ground composition into maximal cliques. In *ICCV*, 2011.
[15] T. Kanade. Image understanding research at CMU. In *DARPA IUW*, 1987.

[16] I. Kokkinos. Boundary detection using f-measure, filter- and feature-(f^3) boost. In *ECCV*, 2010.
[17] I. Kokkinos. Highly accurate boundary detection and grouping. In *CVPR*, 2010.
[18] S. Konishi, A. Yuille, and J. Coughlan. Fundamental bounds on edge detection: An information theoretic evaluation of different edge cues. In *CVPR*, 1999.
[19] M. Koschan and M. Abidi. Detection and classification of edges in color images. *SPM-SICIP*, 22(1), 2005.
[20] J. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright. Convergence properties of the Nelder-Mead simplex method in low dimensions. *SIAM Optimization*, 9(1), 1998.
[21] M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *CVPR*, 2007.
[22] M. Leordeanu, R. Sukthankar, and C. Sminchisescu. Efficient closed-form solution to generalized boundary detection. In *ECCV*, 2012.
[23] A. Levin, D. Lischinski, and Y. Weiss. A closed form solution to natural image matting. In *CVPR*, 2006.
[24] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *IJCV*, 30(2), 1998.
[25] J. Mairal, M. Leordeanu, F. Bach, M. Hebert, and J. Ponce. Discriminative sparse image models for class-specific edge detection and image interpretation. In *ECCV*, 2008.
[26] D. Marr and E. Hildtreth. Theory of edge detection. In *Proc. Royal Society*, 1980.
[27] D. Martin, C. Fawlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5), 2004.
[28] P. Meer and B. Georgescu. Edge detection with embedded confidence. *PAMI*, 23(12), 2001.
[29] M. Morron and R. Owens. Detecting and localizing edges composed of steps, peaks and roofs. *Pattern Recognition Letters*, 1987.
[30] M. Nitzberg and T. Shiota. Nonlinear image filtering with edge and corner enhancement. *PAMI*, 14(8), 1992.
[31] S. Palmer. *Vision science: photons to phenomenology*. 1999.
[32] P. Parent and S. Zucker. Trace inference, curvature consistency, and curve detection. *PAMI*, 11(8), 1989.
[33] P. Perez, A. Blake, and M. Gagnet. Jetstream: Probabilistic contour extraction with particles. In *ICCV*, 2001.
[34] P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks and roofs. In *ICCV*, 1990.
[35] M. Petrou and J. Kittler. Optimal edge detectors for ramp edges. *PAMI*, 13(5), 1991.
[36] J. Prewitt. Object enhancement and extraction. In *Picture Processing and Psychopictorics*. Academic Press, 1970.
[37] X. Ren. Multi-scale improves boundary detection in natural images. In *ECCV*, 2008.
[38] X. Ren and L. Bo. Discriminatively trained sparse code gradients for contour detection. In *NIPS*, 2012.
[39] X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using conditional random fields. In *ICCV*, 2005.
[40] L. Roberts. Machine perception of three-dimensional solids. In *Optical and Electro-Optical Information Processing*. 1965.
[41] M. Ruzon and C. Tomasi. Edge, junction, and corner detection using color distributions. *PAMI*, 23(11), 2001.
[42] M. Sargin, L. Bertelli, B. Manjunath, and K. Rose. Probabilistic occlusion boundary detection on spatio-temporal lattices. In *ICCV*, 2009.
[43] A. Shashua and S. Ullman. Structural saliency: the detection of globally salient structures using a locally connected network. In *ICCV*, 1988.
[44] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8), 2000.
[45] C. Snoek, M. Worring, and A. Smeulders. Early versus late fusion in semantic video analysis. In *ACM-ICM*, 2005.
[46] A. Stein and M. Hebert. Occlusion boundaries from motion: Low-level detection and mid-level reasoning. *IJCV*, 82(3), 2009.
[47] A. Stein, T. Stepleton, and M. Hebert. Towards unsupervised whole-object segmentation: Combining automated matting with boundary detection. In *CVPR*, 2008.
[48] D. Sun, S. Roth, and M. Black. Secrets of optical flow estimation and their principles. In *CVPR*, 2010.
[49] P. Sundberg, T. Brox, M. Maire, P. Arbelaez, and J. Malik. Occlusion boundary detection and figure/ground assignment from optical flow. In *CVPR*, 2011.
[50] E. Trulls, I. Kokkinos, A. Sanfeliu, and F. Moreno-Noguer. Dense segmentation-aware descriptors. In *CVPR*, 2013.
[51] N. Widynski and M. Mignotte. A particle filter framework for contour detection. In *ECCV*, 2012.
[52] L. Williams and D. Jacobs. Stochastics completion fields: A neural model of illusory contour shape and salience. In *ICCV*, 1995.
[53] G. Ye, D. Liu, I.-H. Jhuo, and S.-F. Chang. Robust late fusion with rank minimization. In *CVPR*, 2012.
[54] Q. Zhu, G. Song, and J. Shi. Structural saliency: the detection of globally salient structures using a locally connected network. In *ICCV*, 1988.

Marius Leordeanu is a senior research scientist at the Institute of Mathematics of the Romanian Academy. Marius received his Ph.D. in Robotics from Carnegie Mellon University in 2009 and Bachelor degrees in Mathematics and Computer Science from the City University of New York, 2003. His research focuses on computer vision and machine learning, with contributions in learning and optimization for matching and probabilistic graphical models, object category recognition, object tracking, 3D modeling of urban scenes, video understanding and boundary detection.

Rahul Sukthankar is a scientist at Google Research, an adjunct research professor in the Robotics Institute at Carnegie Mellon and courtesy faculty in EECS at the University of Central Florida. He was previously a senior principal researcher at Intel Labs, a senior researcher at HP/Compaq Labs and research scientist at Just Research. He received his Ph.D. in Robotics from Carnegie Mellon in 1997 and his B.S.E. in Computer Science from Princeton in 1991. His current research focuses on computer vision and machine learning, particularly in the areas of object recognition, video understanding and information retrieval.

Cristian Sminchisescu has obtained a doctorate in Computer Science and Applied Mathematics with an emphasis on imaging, vision and robotics at INRIA, France, under an Eiffel excellence doctoral fellowship, and has done postdoctoral research in the Artificial Intelligence Laboratory at the University of Toronto. He is a member in the program committees of the main conferences in computer vision and machine learning (CVPR, ICCV, ECCV, NIPS, AISTATS), area chair for ICCV07-13, and an Associate Editor of IEEE PAMI. He has given more than 100 invited talks and presentations and has offered tutorials on 3d tracking,recognition and optimization at ICCV and CVPR, the Chicago Machine Learning Summer School, the AERFAI Vision School in Barcelona and the Computer Vision Summer School (VSS) in Zurich. His research interests are in the area of computer vision (3D human pose estimation, semantic segmentation) and machine learning (optimization and sampling algorithms, structured prediction, and kernel methods).