SLEDGE: Sequential Labeling of Image Edges for Boundary Detection

Nadia Payet and Sinisa Todorovic
School of EECS, Oregon State University
Kelley Engineering Building, Corvallis, OR, USA
payetn@onid.orst.edu, sinisa@eecs.oregonstate.edu

Abstract

Our goal is to detect boundaries of objects or surfaces occurring in an arbitrary image. We present a new approach that discovers boundaries by sequential labeling of a given set of image edges. A visited edge is labeled as on or off a boundary, based on the edge's photometric and geometric properties, and evidence of its perceptual grouping with already identified boundaries. We use both local Gestalt cues (e.g., proximity and good continuation), and the global Helmholtz principle of non-accidental grouping. A new formulation of the Helmholtz principle is specified as the entropy of a layout of image edges. For boundary discovery, we formulate a new, policy iteration algorithm, called SLEDGE. Training of SLEDGE is iterative. In each training image, SLEDGE labels a sequence of edges, which induces loss with respect to the ground truth. These sequences are then used as training examples for learning SLEDGE in the next iteration, such that the total loss is minimized. For extracting image edges that are input to SLEDGE, we use our new, low-level detector. It finds salient pixel sequences that separate distinct textures within the image. On the benchmark Berkeley Segmentation Datasets 300 and 500, our approach proves robust and effective. We outperform the state of the art both in recall and precision for different input sets of image edges.

1. Introduction

This paper addresses a basic vision problem, that of detecting boundaries of objects or surfaces occurring in an arbitrary image. Shape is widely recognized as one of the most categorical object features [6]. Thus, boundary detection is often used as input to a wide range of higher-level vision tasks, including object recognition [9, 8, 22], scene reconstruction from stereo [55, 70], and tracking [27]. Aimed as an initial step of diverse vision systems, boundary detection in this paper is not informed about specific objects present in the scene, i.e., about their photometric, geometric, and structural properties. Also, it is not based on any assumptions about number, scale, and layout of objects in the scene.

We make a distinction between boundaries and edges. An edge is defined as a sequence of contiguous pixels passing through a ramp of relatively abrupt changes of low-level image properties, such as brightness, or variations of brightness. A boundary, instead, represents the change in subimage ownership between two distinct objects or surfaces present in the image. Since visible boundaries coincide with a subset of edges in the image, we parse the problem of boundary detection into two subproblems: (a) detecting edges, and (b) identifying which of these detected edges represent object boundaries. Thus, two critical ideas lie at the foundation of our approach, as explained below.

First, we use edges as mid-level image features, since they offer a number of advantages over other feature types, such as patches [4] or regions [19], often used in previous work. Edges are dimensionally matched with object boundaries, and thus naturally facilitate boundary detection. Other types of features typically require additional processing for their mapping to boundaries. A set of extracted edges provides rich information about the spatial extent, and local and global spatial interactions among objects occurring in the image. We formalize this information by using the Gestalt principles of grouping [53, 38].

Second, we build on previous work that shows that it is possible to learn to detect boundaries in arbitrary images (e.g., [51, 57]). Given manually annotated training images, our goal is to learn how to optimally combine perceptual-grouping cues with intrinsic properties of the extracted edges for boundary detection. This is challenging because the relative significance of global vs. local image properties, as well as the priority ordering of Gestalt laws significantly varies across different scenes.

Our approach consists of the following major steps. Given a set of training images, we first extract edges from each image. To this end, we use our new salient edge detector that finds contiguous pixel sequences that separate distinct textures within the image. A classifier is then learned to sequentially label the edges as on or off a boundary. All decisions made before visiting an edge are used to provide perceptual-grouping evidence for classifying that edge. For example, an edge will more likely be classified as boundary if it is close to, and continues well previously identified boundaries. The classifier is learned so as to minimize detection error with respect to manually annotated ground truth. When a new image is encountered, its edges are extracted, and then sequentially labeled. Empirical results presented in this paper demonstrate that our approach outperforms the state of the art on benchmark datasets, including the Berkeley segmentation datasets (BSD) 300 and 500 [50, 4], Weizmann Horses [7], and LabelMe [59].

1.1. Motivation

Psychophysical studies have long demonstrated that early human vision strongly favors certain shapes and configurations over others without high-level recognition [53]. Among many theories of perceptual organization that address this phenomenon, Gestalt theory [38], Helmholtz's likelihood principle [29], and Minimum description length (MDL) [30] have had major impact on computer vision [46]. Gestalt theory provides a number of descriptive principles for grouping parts into whole, but does not specify a computational process for achieving this percept. The more formal Helmholtz's likelihood principle specifies the probability of grouping of a set of elements that is low if the placement of the elements is likely to be accidental. MDL provides another related formalism that the grouping should achieve the shortest coding length.

Based on the above theories of perceptual organization, prior work has argued that the key to detecting boundaries in the image is to model the underlying probability distribution governing the ensemble of boundaries and their configurations (e.g., [73, 57]). Such a model would be able to provide quantifiable definitions of perceptual-grouping laws, and their relative significance for boundary detection. To this end, prior work typically resorts to graphical models, such as, Markov Random Fields (MRF) [73], or Conditional Random Fields (CRF) [57]. These graphical models are characterized by: (i) the number of random variables (nodes) and their statistical dependencies (graph connectivity), jointly referred to as the model structure, and (ii) parameters of probability density functions (pdfs) associated with the random variables in the model. While this framework is principled, in practice, various simplification and approximation steps are implemented in inference and learning to handle their computational complexity. First, the model structure is typically manually specified (e.g., pairwise connectivity among neighboring image features). This simplifies the learning problem to estimating only model parameters. However, since the parameters depend on the model structure, such learning may be suboptimal. Also, in inference, it might happen that wrong connectivity of image features to their spatial neighbors may overpower correct evidence provided by other far away features. For example, an edge is very likely classified as background when it is connected in a MRF (or CRF) only to background edges. The issues related to hand-picking MRF/CRF structure have already been studied in the object recognition community (e.g., [33, 1]). Second, tractable inference of MRFs/CRFs typically requires approximations, e.g., Loopy Belief Propagation, or relatively slow Markov Chain Monte Carlo (MCMC) sampling.

We here depart from the above line of thinking. Our thesis is that sequential perceptual grouping is a valid framework for boundary detection. Our motivation comes from the well-recognized processes of human perception, where attention is directed to certain salient locations, sequentially, rather than uniformly to all locations at once [32]. We classify one edge at a time based on its properties and the previous sequence of decisions. The ordering in which the edges are visited is data-driven, where edges with higher confidence in decision have proportionally higher priority, and thus help make decisions in subsequent ambiguous cases. Thus, we use a greedy, policy iteration algorithm, and still achieve better results than the state of the art, MRF/CRF-based approaches.

1.2. Overview of Our Approach

The main steps of our approach are illustrated in Fig. 1.

Step 1: Given an image, we use an edge detector (e.g., Canny) to extract image edges. Step 1 may use any available edge detector, as long as it yields high recall of true boundaries. Our subsequent steps are expected to improve precision of boundary detection. In addition to off-the-shelf edge detectors, we also use our new salient edge detector. It detects contiguous pixel sequences that separate distinct image textures. The extracted edges are then organized in a graph whose nodes correspond to the edges, and arcs capture their spatial relations. The main purpose of this graph is to facilitate computation of Gestalt cues in the subsequent steps, since the graph stores all relevant spatial relations among the edges. When the edge detector produces a hierarchy of edges, then our graph is accordingly hierarchical, where ascendant-descendant arcs capture the recursive embedding of smaller edges within larger ones, and lateral arcs represent neighbor relations of sibling edges (Fig. 1). The root of this hierarchical graph is a virtual node, conveniently introduced to indicate that the longest edges in the image are siblings. Nodes and arcs in the graph are characterized by descriptors of edge properties (e.g., saliency, repeatability in the image, collinearity with neighbors, symmetry, etc.).

Step 2: Nodes in the graph (i.e., edges) are sequentially labeled as boundary or non-boundary, based on their descriptors and perceptual-grouping cues. At each edge, both pairwise and global Gestalt cues (e.g., collinearity and

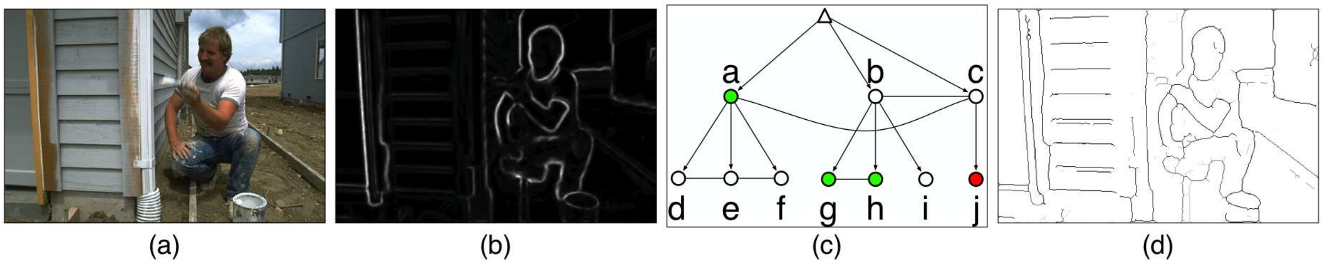


Figure 1. Overview: (a) The input image. (b) Low-level edge detectors, typically output a probability map of occurrence of edges. Analyzing this map at an exhaustive range of values yields a set of distinct edges, at different scales, where some edges may cover the same pixels. (c) The resulting edges are organized in a graph that captures their hierarchical (part-of) and neighbor relationships. SLEDGE sequentially labels each node (i.e., edge) as being on (green) or off (red) an object boundary. (d) The identified boundaries are characterized by SLEDGE's confidence in labeling (darker means higher confidence).

Helmholtz principle) are re-estimated relative to the previously identified boundaries. This is made efficient by utilizing the hierarchical graph structure. For **s**equential **l**abeling of **edge**s we use our new algorithm, called SLEDGE. Training of SLEDGE is iterative. In each iteration, SLEDGE labels a sequence of edges, extracted from each training image as in Step 1. This induces loss with respect to the manually provided ground truth. The training sequences are then used as training examples for learning SLEDGE in the next iteration, such that the total loss is minimized.

Step 3: When a new image is encountered, its edges are sequentially labeled by SLEDGE.

Paper organization: Sec. 2 explains our relationships to prior work, and points out our main contributions. Sec. 3 describes intrinsic, pairwise, and global layout properties of image edges that we use for identifying the evidence of Gestalt laws of grouping. Sec. 4 specifies SLEDGE. Sec. 5 describes our new edge detector that provides input to SLEDGE. Finally, Sec. 6, and Sec. 7 present our experimental evaluation, and concluding remarks.

2. Prior Work and Our Contributions

This section reviews prior work and outlines our contributions. Our review is organized with respect to the three major contributions of this paper—namely, we first point out the novelty of our approach to boundary detection, then, explain our relationships to previous work on sequential labeling, and, finally, describe our contributions in formalizing the Helmholtz principle of perceptual organization.

2.1. Boundary Detection

Prior work has addressed the problem of boundary detection using low-, mid-, and high-level visual cues. Since this paper considers boundary detection with no prior knowledge about objects in the scene, we focus our literature review only on methods that use low- and mid-level features.

Methods based on low-level features typically classify small patches, centered at each pixel location, as on or off an object boundary. They use a bank of filters to identify abrupt changes in image texture [58, 52, 11, 54, 67], and seek the right scale at which these changes occur [45, 25]. Recent work focuses on learning how to classify descriptors of image patches from manually segmented, training images [41, 42, 51, 18, 39]. However, unlike image edges that we use as basic features, patches are not dimensionally matched to boundaries. As a result, image patches provide only limited, local information about boundary presence. The key difference from our approach is that they classify fixed, precomputed descriptors associated with patches, independently of one another. Instead, we sequentially classify dynamically changing descriptors associated with edges, where both the descriptor values and classification at each step are based on the previous decisions.

Motivated by the Gestalt theories of grouping in human perception [53, 38], another family of methods use mid-level image features, such as edges and regions, for boundary detection. Edge-based approaches, first, extract a clutter of image edges, and then use the Gestalt principles of grouping to select and link a subset of the edges into boundaries [68, 73, 60, 47, 72, 56, 57, 40]. Region-based methods, first, identify image regions using, e.g., Normalized-cut, Meanshift, or Maximally stable extremal regions, then, conduct Gestalt grouping of these regions, and finally take contours of the mergers as boundaries [66, 71, 3, 19]. Due to accounting for more global information in the image carried by mid-level features, these methods typically outperform the patch-based approaches.

Our approach differs in two key aspects. First, some of these methods (e.g., [40]) detect boundaries by iterating two separate steps – namely, classifying each image edge independently based on fixed edge descriptors, and grouping detected boundaries for the next iteration. By contrast, grouping and classifying image edges are not two separate steps in our approach.

Second, the other methods in this family make the assumption that edges (or regions) and their interactions minimize the total energy of a MRF or CRF (e.g., [68, 73, 60, 47, 72]), and thus unify edge classification and grouping under the MRF/CRF inference. One can also view our approach as a simple form of MAP inference on an MRF (or CRF) over image edges, using relaxation labeling. The labels of some nodes in such a hypothetical MRF/CRF are fixed to sequentially classify another node, until all nodes are labeled. We may iteratively continue relaxation re-labeling of the image edges until certain convergence criterion is met (similarly to Loopy Belief Propagation). The main difference from the aforementioned MRF/CRF-based methods is that they typically conduct the MAP inference of only node labels, given a fixed MRF/CRF graph and fixed unary and pairwise potential functions. By contrast, labeling of each edge, in our approach, dynamically modifies both the connectivity and potential functions of the (hypothetical) MRF/CRF. Thus, in our inference, we estimate both the MAP graph structure and node labels.

More sophisticated MRF/CRF models with adaptive graph structures [24] have recently been used for boundary detection. For example, in [57], CRF inference accommodates dynamic changes of the CRF connectivity during loopy belief propagation. They define pairwise potentials only for neighboring boundaries, i.e., edges that are turned “on” in the previous iteration. However, they account only for pairwise Gestalt cues. They ignore important global perceptual grouping cues, which would amount to incorporating higher-order cliques in their CRF.

As our key novelty, using the terminology of the MRF/CRF community, we sequentially re-estimate a higher-order clique of boundaries and their associated potential function. We define this higher-order potential as the Helmholtz likelihood of perceptual organization. Another important difference is that [57] uses a weighted sum of the CRF’s clique potentials, i.e., a linear function of potentials, to label edges. By contrast, our decision function for labeling edges is non-linear, learned by the decision-tree classifier. The advantages of using non-linear decision functions vs. log-linear ones for boundary detection are largely unexplored, and deserve a more thorough investigation, beyond the scope of this paper. Our results demonstrate that even without accounting for global cues of perceptual grouping, we achieve very similar performance (marginally above) to that of [57].

Our work is also related to research on tracking edges for boundary detection. This tracking extracts either one boundary given its starting point [26, 12], or multiple boundaries without knowing their starting points [61, 28, 69, 34, 47, 21]. The edge tracking is typically based on the likelihood that a boundary passes through a particular location in the image, using different edge saliency measures and affinities between them as a function of proximity, curvilinear continuity, and closure [28, 61, 68, 69]. In general, these methods resort to different heuristics: (i) assumptions about the total number of boundaries; (ii) protocols for tracking (e.g., how to start); and (iii) definitions of edge affinities (e.g., using a few Gestalt principles with equally weighted relevance for tracking). Once estimated, these affinities are kept fixed while tracking the edges. For example, in [21], the boundary finding algorithm starts from short edge fragments, and iteratively expands “promising” boundaries, such that their total number is small, and that a total cost of assigning the edges to the boundaries versus background is minimum. Our work differs in several aspects. We use training examples to learn how to estimate affinities between pairs of edges by optimally combining a number of Gestalt cues with intrinsic edge properties. During the course of our sequential edge labeling, the Gestalt cues are adaptively re-estimated with respect to the previously identified boundaries, and thus our edge affinities are continually changing. Also, we use training examples to learn how to optimally track edges under their dynamically changing affinities.

2.2. Sequential Labeling

Sequential edge labeling can be viewed as an instance of the structured prediction problem, where the goal is to predict a (large) collection of related output variables for a given input. A number of structured-prediction formulations are popular in computer vision, including, e.g., CRF [43], SVM-struct [64], and M^{3}N [62]. Recently a new computational framework, called SEARN, has been presented in [31]. SEARN is aimed at integrating search and learning for solving complex structured prediction problems. This general framework is particularly well-suited for our objectives, because it transforms a structured prediction problem into a sequential classification problem, and provides a strong theoretical guarantee that good performance on the derived sequential classification implies good performance of the original structured prediction. As a major advantage over the above formulations, SEARN makes no assumptions about the underlying structure and statistical dependencies of the output variables (e.g., no need for factorizing the joint probability distribution of the output variables, as in CRF). In our case, this means that SEARN will allow relaxing the assumptions (i)–(iii), frequently made by previous work, reviewed in Sec. 2.1, about the total number and layout of boundaries in the image. For these reasons, we use the general framework of SEARN to develop a new algorithm for **s**equential **l**abeling of image **edge**s, called SLEDGE.

2.3. Helmholtz Principle of Grouping

Most methods in computer vision focus on Gestalt laws of grouping, whereas other theories of perceptual organization have received scant attention. This is, in part, due to the well-known difficulties associated with formalizing quantifiable models of these theories. For example, only a few approaches to object boundary detection use the Helmholtz likelihood principle [15, 14, 16]. It states that a grouping as a whole is perceptually "meaningful" if it rarely occurs in a random situation. To define "meaningfulness" of a spatial configuration of edges, existing work typically defines an a priori distribution of edge layouts (e.g., binomial distribution [15]), under the assumption that the edges are statistically independent. They declare a specific configuration as ε-meaningful if the expectation of this configuration is less than ε. In this paper, we relax the independence assumption, and use the entropy of the spatial configuration of edges to globally characterize their layout as a whole. As shown in this paper, relatively low values of this entropy indicate "meaningful" layouts of edges, whereas higher entropy values are obtained for spatial configurations of edges belonging to background clutter. This allows efficient and robust classification of edge layouts as "meaningful" or clutter by our SLEDGE. Thus, instead of committing to the relatively harder task of estimating the prior of edge layouts, as in [15, 14], we formalize the Helmholtz principle within the discriminative framework of our SLEDGE.

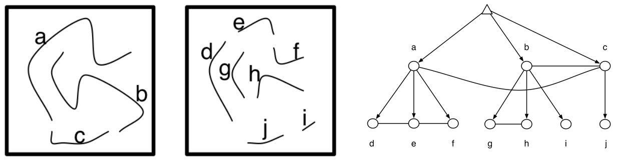
2.4. Our Contributions

1. We formulate a new policy iteration algorithm, called SLEDGE, and use it for labeling edges. SLEDGE introduces a number of modifications to the original SEARN framework, presented in [31], including the use of: (i) iteratively changing classifier confidence for ordering image edges for their sequential labeling, instead of a deterministic ordering used by SEARN; and (ii) voting classifier decisions, instead of probabilistically sampling a classifier from the ensemble of iteratively learned classifiers by SEARN.

2. We account for both pairwise local cues and global evidence of perceptual organization. We exploit the Helmholtz likelihood principle. It states that whenever some large deviation from randomness occurs, a structure is perceived. We formalize the Helmholtz principle as the entropy of the edge layout. The smaller the entropy of a spatial configuration of the edges, the more likely they jointly represent object boundaries.

3. The Graph of Image Edges

Our approach discovers boundaries by sequential labeling of a given set of image edges. In Step 1, we use a low-level detector to identify the image edges, and then extract their intrinsic and layout properties. In Step 2, SLEDGE sequentially labels edges as boundary or non-boundary, based on their descriptors and perceptual grouping cues. In Sec. 5, we give more details on particular edge detectors that we use. In this section, we assume that the edges are already given, and focus on describing their intrinsic and *local* layout properties (Sec. 3.2). Sec. 3.3 then continues with the description of their *global* layout properties.



(a) (b) (c)

Figure 2. The graph of edges: (a) Large edges detected at a certain scale. (b) Smaller edge fragments detected at another scale that coincide with larger ones in (a). (c) Ascendant-descendant arcs in the graph represent the embedding of smaller edges within larger ones. Lateral arcs connect all edge pairs that are estimated as neighbors in the image via Delaunay triangulation (Fig. 4). The figure shows lateral arcs only between sibling nodes, for better visibility. The root node is a virtual node that defines that the largest edges in the image are siblings.

Given a set of image edges, they are represented as nodes in a graph whose connectivity is image- and edge-detector-dependent. The main purpose of this graph is to organize spatial information about the edges in a manner that will make efficient the iterative re-estimation of Gestalt cues during the sequential edge labeling, in Steps 2 and 3. To this end, arcs of the graph capture hierarchical (part-of) and neighbor relationships between the edges, as further explained in subsections 3.1 and 3.2. As one of our key contributions for boundary detection, we also use a global characterization of the spatial configuration of edges. Specifically, we formalize the Helmholtz principle of perceptual organization, and use it as a global grouping evidence for boundary detection. This is discussed in subsection 3.3.

3.1. Building the Graph of Image Edges

When the edge detector outputs a hierarchy of edges (e.g., gPb [49]), ascendant-descendant arcs are established between the graph nodes representing larger (parent) edges and their constituent, smaller edge fragments (children), as shown in Fig. 2. An illustrative example of this embedding is shown in Fig. 3. The root node is a virtual node that defines that the largest edges in the image are siblings. Otherwise, the graph is not hierarchical, but captures only pairwise spatial adjacency of the edges.

Lateral arcs in the graph are established between neighboring edges. We first define the neighbor relations for sibling nodes (i.e., edges) under the same parent, and then neighbor relations for all other non-sibling nodes. Two sibling edges are called neighbors if there is no other edge that intersects the shortest line connecting their endpoints, and if their endpoints are near each other and sufficiently far

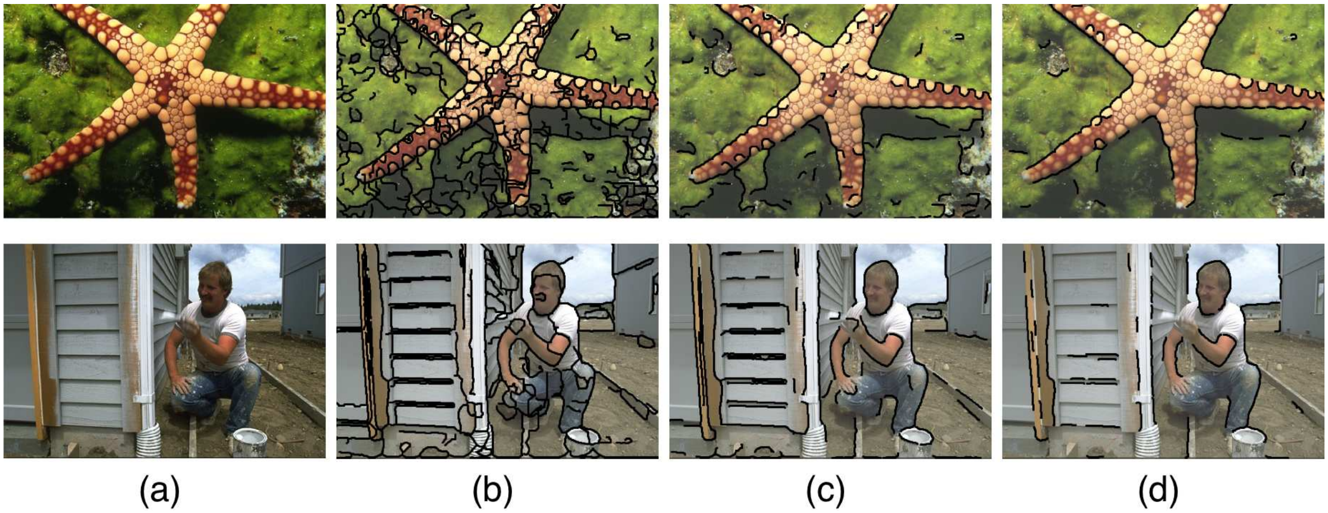


Figure 3. **Multiple layers of the graph of edges.** (a) Original image. (b,c,d) Edges present in the first, second and third layers of the hierarchical graph. We see that long edges break up into multiple edges - see the shadow of the starfish ((c) and (d) in top row) or the right edge of the foreground house ((c) and (d) in bottom row). Isolated edges do not generate children nodes - see the many small edges in the green texture (top row) or the edges on the jeans of the man (bottom row).

from endpoints of the other edges. This is formalized using the standard Delaunay triangulation (DT), as illustrated in Fig. 4. DT is the unique triangulation of a set of vertices in the plane, such that no vertex is inside the circumcircle of any triangle. DT (and its dual Voronoi diagram) conveniently accounts for both local and global spatial interactions among vertices, and thus has been extensively used in the literature to define layout properties of image features [57]. To find the neighbors, we traverse the graph, node by node, and, for each group of sibling edges, we construct the DT of their endpoints. If a pair of endpoints gets connected in the DT, this means that they satisfy the above stated requirements of simultaneous nearness and isolation from the others. If these endpoints are also directly "visible" to each other in the image, then their edges are declared as neighbors. Regarding any other pair of edges, we say that non-sibling edges recursively inherit neighbor relations from their respective parents, i.e., two non-sibling edges are neighbors if their corresponding parents are neighbors. The recursion ends with the virtual root node in the graph. Note that we do not specify a heuristic minimum distance between edges to define neighbors. This allows us to estimate proximity and good continuation between even fairly distant edges in the image, as sometimes is indeed necessary.

3.2. Attributes of Nodes and Arcs

In the previous subsection, we have explained how to build the graph, $G = (V, E, \psi, \phi)$, from a given set of image edges. Nodes $i \in V$, and arcs $(i, j) \in E$ in the graph are characterized by descriptor vectors, $\psi : V \to \mathbb{R}^2_+$, and $\phi : E \to \mathbb{R}^5_+$. The node and arc descriptors are defined in terms of intrinsic and spatial layout properties of the corresponding image edges. In the sequel, we first specify the node attributes $\boldsymbol{\psi}_i$, and then define the arc attributes $\boldsymbol{\phi}_{ij}$.

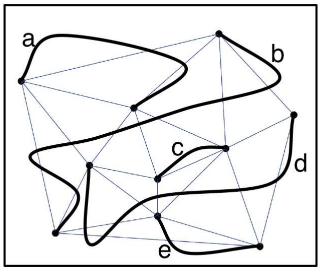


Figure 4. Delaunay triangulation (thin lines) of the endpoints of an example set of edges (bold curves). Delaunay triangulation (DT) is aimed at capturing the evidence of Gestalt grouping among image edges. Two edges are called neighbors if at least one pair of their respective endpoints are connected in the DT, and this connection is not intersected by any edge. For example, the DT correctly finds that only the pair (c,d) are neighbors, reducing the complexity of estimating Gestalt cues for all possible pairs of edges.

3.2.1 Node Attributes

The descriptor of intrinsic edge properties, $\boldsymbol{\psi}_i=[\boldsymbol{\psi}_{i1}, \boldsymbol{\psi}_{i2}]$, associated with every node $i \in V$, includes: (a) measure of edge saliency, and (b) measure of edge repeatability in the image. Since object boundaries are usually salient, the information about edge saliency is recorded as $\boldsymbol{\psi}_{i1}$. Typically, this information can be directly obtained from the low-level edge detector, used in Step 1. For example (see Fig. 1), the detector gPb [49]) outputs a probability map of boundary occurrence in the image, Pb, whose mean value along an edge we use as the saliency of that edge,

$\psi_{i1} = \text{mean}_i(Pb)$. In the case that the low-level detector does not output saliency (e.g., Canny), for each image edge, we estimate its saliency as the mean of magnitude of intensity gradient along the edge, $\psi_{i1} = \text{mean}_i(\text{gradient})$.

We also want to identify repeating edges in the image, because they are likely to arise from image texture, rather than representing object boundaries. To this end, we match all edge pairs in the image. Then, we estimate the degree of repetition of an edge as a function of its total similarity to and number of best matches. This is formalized using the standard link analysis algorithm Page Rank, used by the Google Internet search engine [10]. All image edges are first linked, and the associated similarity, s_{ij}, between the corresponding pair of image edges (i, j) is computed as $s_{ij} = \exp(-\|SC_i - SC_j\|_2)$, where SC denotes the standard shape descriptor Shape Context [5]. Log-polar bins of the Shape Context descriptor are scaled so as to cover the entire edge, which makes s_{ij} scale invariant. Note that s_{ij} is purposefully not rotation invariant, because our goal is to identify edges within image textures. Since image texture is usually characterized by texture elements (a.k.a. textons) which have similar orientation, it is likely that the textons will give rise to edges with similar orientations, and thus s_{ij} should not be rotation invariant. After linking all image edges and computing their similarities, Page Rank is used to iteratively estimate the degree of repetition of each edge as $\psi_{i2} = (1 - \rho) + \rho \sum_j \frac{\psi_{j2}}{\sum_j s_{ij}}$, where ρ is the residual probability, set in our experiments to $\rho = 0.85$. Similar Page-Rank based approaches to estimating the degree of repetition of image features have been used in [36, 44].

3.2.2 Arc Attributes

Pairwise layout properties between image edges are used to specify the descriptor, $\phi_{ij} = [\phi_{ij1}, \ldots, \phi_{ij5}]$. By definition, we set $\phi_{ij} = \mathbf{0}$ for all edge pairs (i, j) that are not neighbors. For neighbors, ϕ_{ij} includes the standard formulations of Gestalt principles of grouping [46, 73, 57].

Let Q_i and Q_j denote the 2D coordinates of two endpoints of neighboring edges i and j. If i and j are siblings then Q_i and Q_j are those points based on which i and j are found to be neighbors in the Delaunay triangulation, as explained in Sec. 3.1. Otherwise, Q_i and Q_j are the closest pair of endpoints. We estimate:

1. Proximity as $\phi_{ij1} = \frac{2\pi\|Q_i - Q_j\|^2}{\min(\text{len}(i), \text{len}(j))^2}$, where $\text{len}(\cdot)$ measures the length of an edge.

2. Collinearity as $\phi_{ij2} = |\frac{d\theta(Q_i)}{dQ_i} - \frac{d\theta(Q_j)}{dQ_j}|$. $\theta(Q) \in [0, 2\pi]$ measures the angle between the x-axis and the tangent of the edge at endpoint Q, where the tangent direction is oriented towards the curve.

3. Co-Circularity as $\phi_{ij3} = |\frac{d^2\theta(Q_i)}{dQ_i^2} - \frac{d^2\theta(Q_j)}{dQ_j^2}|$.

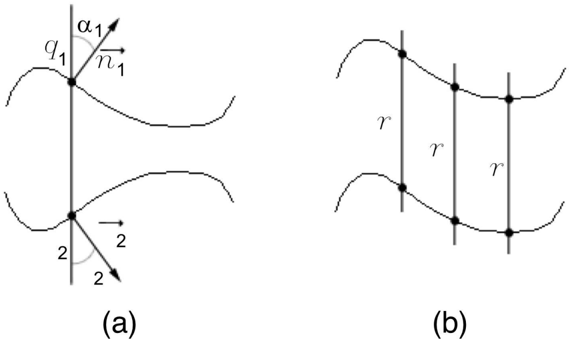


Figure 5. (a) The line between symmetric points q_1 and q_2 lying on two distinct edges subtends the same angle with the respective edge normals. (b) Constant distance r between symmetric points of two distinct edges indicates parallelism of the edges.

4. Symmetry as $\phi_{ij5} = \text{mean}_{q \in i} \left| \frac{d^2 r(q)}{dq^2} \right|$. As illustrated in Fig. 5(a), two points $q_1 \in i$ and $q_2 \in j$ are symmetric iff angles $\alpha_1 = \alpha_2$, where α_1 is subtended by line (q_1, q_2) and normal to i, and α_2 is subtended by line (q_1, q_2) and normal to j.

5. Parallelism as the mean of distance variations $r(q)$ between points q along edge i and their symmetric points on edge j, $\phi_{ij4} = \text{mean}_{q \in i} \left| \frac{dr(q)}{dq} \right|$. Fig. 5(b) shows two parallel edges. The distance between symmetric points remains constant.

Below, we explain in greater detail how to find corresponding symmetric points (q_1, q_2) between two edges, and then estimate the above defined symmetry and parallelism. For each edge, we extract salient, high curvature points, using the standard, scale-invariant algorithm of [63]. To find the symmetric points on two contours, we consider all pairs of salient points (q_1, q_2), where q_1 belongs to the first edge, and q_2 belongs to the second edge, and compute their symmetry cost matrix. The symmetry cost is defined as the absolute difference between the angles α_1 and α_2, depicted in Fig. 5(a). If q_1 and q_2 are symmetric, $\alpha_1 = \alpha_2$ and the cost is equal to 0. After building a symmetry cost matrix for all possible pairs (q_1, q_2), we then use the standard Dynamic Time Warping algorithm to obtain the best symmetric points of the two edges.

This concludes our description of intrinsic and local layout properties of edges.

3.3. A Global Grouping Cue

The Helmholtz principle of grouping describes a phenomenon that "we immediately perceive whatever could not happen by chance" [29]. We formalize this principle as the entropy of a layout of edges in the image. When this entropy is low, the edges are less likely to belong to background clutter.

To characterize the layout of a given set of image edges, we use the generalization of the Voronoi diagram for point patterns to edges, presented in [2]. This generalization has been shown to accurately capture the global spatial interaction among all edges in the image, as illustrated in Fig. 6, and, consequently, lead to performance improvements in object recognition. The intuition underlying this generalization is that edges are exposed to each other via every point that lies on them, and thus the edge-based Voronoi tessellation can be derived from the classical point-based Voronoi diagram. The Voronoi diagram of a point pattern $\mathcal{Q}$ in 2D associates with each point $q \in \mathcal{Q}$ a convex polygon γ_q which is the region closer to q than to any other point $q' \in \mathcal{Q}$, $\gamma_q = \{q'' : q'' \in \mathbb{R}^2, \forall q' \in \mathcal{Q}, \|q - q''\|^2 < \|q' - q''\|^2\}$. Thus, for any nondegenerate distribution of points in 2D, the Voronoi diagram tessellates the 2D space into a set of convex polygons, each containing exactly one of the points in the pattern $\mathcal{Q}$. The Voronoi diagram can be computed very efficiently (for n points, complexity is $O(n \log n)$).

Given a set of edges, we first compute the point-based Voronoi tessellation for all pixels along the edges (Fig. 6). Then, for each edge i, we find the union of the Voronoi polygons of the edge's pixels, resulting in a generalized Voronoi cell, $U_i = \cup_{q \in i} \gamma_q$. The Voronoi cell of edge i defines the relative area of its influence in the image, denoted as $P_i = \text{area}(U_i)/\text{area}(\text{image})$. For n image edges, we define the entropy of their layout, H, as

$$H = -\sum_{i=1}^{n} P_i \log P_i \qquad (1)$$

Note that P_i depends on the length of i, and its spatial position and orientation relative to the other edges in the image. Since background clutter consists of edges of different sizes, which are placed in different orientations, and at various distances from the other edges, H of the clutter is likely to take larger values than H of object boundaries. This is demonstrated in the following experiment on 300 natural images of the Berkeley segmentation dataset (BSD) [50]. We have computed H for three distinct sets of edges in each image. The first set of edges consists of only manually annotated object boundaries. The second set consists of edges that are detected in the image (e.g., Canny), but do not coincide with true object boundaries, i.e., background edges. Finally, the third set consists of all edges detected in the image. Fig. 7 shows the three histograms of H values obtained for these three sets across the entire BSD. As can be seen, the histograms of object boundaries, and background edges form two distinct peaks. The entropies of boundary layouts are in general lower than those of background clutter. This suggests that the entropy, defined in (1), allows efficient learning of a classifier which can robustly separate the spatial configurations of boundaries versus those of background edges.

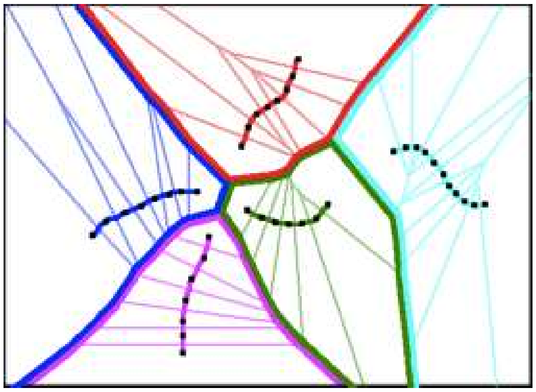


Figure 6. The Voronoi diagram is computed for each pixel (black) of each edge (colored bold curves with black points). Voronoi polygons generated by pixels of an edge are marked with the same color as that edge. A Voronoi cell U_i is the union of all Voronoi polygons generated by pixels on edge i.

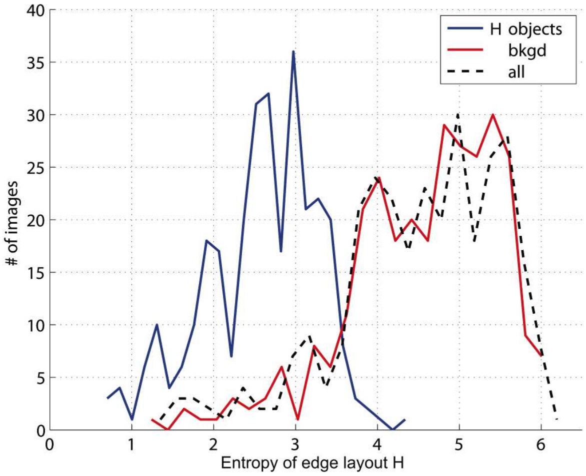


Figure 7. We formalize the Helmholtz principle of grouping as the entropy H of a layout of image edges. Three histograms of H values are obtained for three different sets of edges from all images in the Berkeley segmentation dataset [50]: (a) manually annotated object boundaries, (b) background edges, and (c) all edges. The H values of boundary layouts are in general smaller than the entropies of spatial configurations of all edges (objects+background). This allows a robust separation of boundary vs. background-clutter layouts.

Image edges and their intrinsic and spatial layout properties, described in Sec. 3 and Sec. 3.3, are used in the sequential labeling of edges toward boundary detection, as explained in the next section.

4. Sequential Labeling of Edges

This section presents Step 2 of our approach. We formulate boundary detection as a sequential assignment of binary labels "on" or "off" an object boundary to every edge in the image. Let $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n)$ denote a sequence of descriptor vectors $\boldsymbol{X}_i$ associated with the image edges

$i \in V$ that are sequentially labeled in steps $t = 1, \dots, n$. The set of all sequences of descriptors extracted from all images is denoted as $\mathcal{X} \ni \boldsymbol{X}$. Also, let $\boldsymbol{Y} = (y_1, y_2, \dots, y_n)$ denote their corresponding labels, $y_i \in \{0, 1\}$. The set of all structured predictions is denoted as $\mathcal{Y} \ni \boldsymbol{Y}$. Thus, we specify boundary detection as mapping $f : \mathcal{X} \to \mathcal{Y}$. We below explain how to learn f.

Reinforcement Learning (RL) seems a suitable framework to learn f. RL finds an optimal policy that maps *states* of an environment to the *actions* that an *agent* ought to take in those states, so as to minimize a long-term loss [35]. In our case, the agent is classifier f that takes an action of selecting and labeling one of the unlabeled edges, given the state defined in terms of previously identified boundaries, and, then, receives a loss (e.g., 0 if the labeling is correct, or 1, otherwise). The goal of RL is to learn the optimal policy f that will minimize the expected total loss over all loss-sequences, observed during the sequential labeling of edges in all training images. The optimal policy, in our case, is the human annotation of object boundaries in training images. Existing RL algorithms, however, are only tractable for environments with a few states and actions [35]. Therefore, they cannot be used for our purposes, since we have an exponentially increasing number of states, 3^t, as edges may have three possible labels: "on", "off", or unlabeled.

There is a large body of research aimed at developing feasible RL algorithms. A review of this work is beyond our scope. We here study one of the latest frameworks, called SEARN [31]. It is aimed at integrating search and learning for solving complex structured prediction problems, such as RL. It transforms RL into a classification problem, and shows that good classification performance entails good RL performance. In SEARN, in every time step, a current state of the environment is represented by a vector of observable features, which is then input to a classifier to predict the right action. Thus, learning the optimal policy within SEARN amounts to learning a classifier over state feature vectors so as to minimize a suitably defined loss function.

Within the SEARN framework, we formulate a new algorithm for sequential labeling of edges, called SLEDGE. Below, we briefly review SEARN and problems associated with this framework, and thus set the stage for the later formulation of SLEDGE, and description of our contributions.

4.1. A Review of the SEARN Framework

SEARN applies a classifier (e.g. Support Vector Machines or Decision Trees), f, to a sequence of interdependent data samples, $\boldsymbol{X} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \dots) \in \mathcal{X}$, and thus infers their labels $\boldsymbol{Y} = (y_1, y_2, \dots) \in \mathcal{Y}$. This framework requires that the ordering of data in $\boldsymbol{X}$ is well-defined. To learn f, SEARN uses an iterative batch-learning approach. Specifically, in each iteration τ, the results of classification, $f^{(\tau)}{:}\boldsymbol{X} \to \boldsymbol{Y}^{(\tau)}$, are compared with the ground-truth labels, denoted as $\hat{\boldsymbol{Y}}$. This induces a loss (e.g., Hamming distance), $l(\boldsymbol{Y}^{(\tau)}, \hat{\boldsymbol{Y}})$, which is then used to learn a new classifier $h^{(\tau+1)}$. In the next iteration, SEARN applies $f^{(\tau+1)}$ to $\boldsymbol{X}$, where $f^{(\tau+1)}$ is defined as an interpolation of the new classifier $h^{(\tau+1)}$ with the old policy $f^{(\tau)}$ as

$$f^{(\tau+1)} = \beta h^{(\tau+1)} + (1 - \beta) f^{(\tau)}, \tag{2}$$

where $\beta \in (0, 1]$ is an interpolation constant. This interpolation amounts to a probabilistic sampling of individual classifiers $h^{(1)}, h^{(2)}, \dots, h^{(T)}$ learned in each iteration $\tau = 1, 2, \dots, T$. The classifier sampling is governed by the multinomial distribution, where, from (2), the probability of selecting classifier $h^{(\tau)}$ in iteration T is

$$\alpha_T^{(\tau)} = \beta(1 - \beta)^{T-\tau}, \;\; \tau = 1, 2, \dots, T. \tag{3}$$

Since $\alpha_T^{(\tau)}$ decreases over time, SEARN iteratively moves away from the initial policy, $h^{(1)}$, toward a fully learned policy. After τ reaches the maximum allowed number of iterations, T, the output of learning is the last policy $f^{(T)}$ from which $h^{(1)}$ is eliminated, i.e., the output is $\{h^{(2)}, h^{(3)}, \dots, h^{(T)}\}$ and their associated sampling probabilities $\{\kappa\alpha_T^{(2)}, \kappa\alpha_T^{(3)}, \dots, \kappa\alpha_T^{(T)}\}$. Here, the constant κ re-scales the α's, such that $\sum_{\tau=2}^{T} \kappa\alpha_T^{(\tau)} = 1$.

Suppose that $h^{(1)}$ is an optimal policy. A theorem presented in [31] states that $f^{(T)}$ is "not much worse" than the initial, optimal policy. SEARN is summarized in Alg. 1.

Algorithm 1: SEARN Framework

```
Input  : Data sequences $\boldsymbol{X} \in \mathcal{X}$ and ground-truth labels $\hat{\boldsymbol{Y}} \in \hat{\mathcal{Y}}$;
         Loss-sensitive classifier $h$, and initial $h^{(1)}$;
         Loss function $l$; Interpolation constant $\beta = 0.1$;
         Maximum number of iterations $T$;
Output : Learned policy $f^{(T)}$;
$f^{(1)} = h^{(1)}$;
for $\tau = 1 \dots T$ do
    for all $\boldsymbol{X} \in \mathcal{X}$ do
        Compute predictions $Y^{(\tau)} = f^{(\tau)}(\boldsymbol{X})$;
        Estimate loss $l(\boldsymbol{Y}^{(\tau)}, \hat{\boldsymbol{Y}})$;
    end
    Learn a new classifier $h^{(\tau+1)} \leftarrow h(\mathcal{X}; l)$;
    Interpolate: $f^{(\tau+1)} = \beta h^{(\tau+1)} + (1 - \beta) f^{(\tau)}$
end
Return $f^{(T)}$ without $h^{(1)}$.
```

4.2. SLEDGE

To address 2D images and our particular problem of boundary detection, in this section, we explore a number of modifications of the SEARN framework, originally formulated for 1D data (e.g., text). We formulate SLEDGE within the SEARN framework by specifying the following: (1) the generation of data samples $\boldsymbol{X}$ from the intrinsic and spatial layout properties of image edges (explained in Sec. 4.2.1) (2) two distinct ranking functions, each providing an ordering of $\boldsymbol{X}$ for the sequential labeling (described in Sec. 4.2.2); (3) two loss functions for the iterative learning of policy f (specified in Sec. 4.2.3); and (4) a new interpolation strategy for fusing the classifiers $\{h^{(\tau)}\}_{\tau=2,3,\ldots}$ into policy f (presented in Sec. 4.2.4).

4.2.1 Input Data to SLEDGE

SLEDGE sequentially labels descriptor vectors, $\boldsymbol{X}$=$(\boldsymbol{X}_1,\ldots,\boldsymbol{X}_n)$, associated with image edges i=$1,\ldots,n$. The descriptors are computed online, as the sequential labeling visits one edge at a time, since they depend on the layout of previously detected boundaries. In each labeling step $t = 1,\ldots,n$, we first use a ranking function (see Sec. 4.2.2) to select the next edge to be labeled, and then compute its descriptor(s), as follows.

The entire set of edges V can be divided, at step t, into edges labeled as "on" or "off", and unlabeled edges, $V = V_{\text{on}}^{(t)} \cup V_{\text{off}}^{(t)} \cup V_{\text{un}}^{(t)}$. To compute the descriptors of unlabeled edges $i \in V_{\text{un}}^{(t)}$, we retrieve from graph $G = (V,E,\phi,\psi)$ the following edge properties: (a) intrinsic properties $\boldsymbol{\psi}_i$; (b) pairwise spatial relations $\boldsymbol{\phi}_{ij}$; and (c) global layout entropy H (defined in Sec. 3.2.1–3.2.2 and Sec. 3.3). This retrieval is efficient, since G, i.e., $\boldsymbol{\psi}_i$ and $\boldsymbol{\phi}_{ij}$ have already been computed in Step 1. Specifically, for each $i \in V_{\text{un}}^{(t)}$, selected by the ranking function, we generate the descriptors $\boldsymbol{X}_i^{(t)} = \{\boldsymbol{x}_{i1}^{(t)},\ldots,\boldsymbol{x}_{ij}^{(t)},\ldots\}$, indexed by object boundaries $j \in V_{\text{on}}^{(t)}$ that are detected by SLEDGE in the previous $(t-1)$ labeling steps. We define

$$\boldsymbol{x}_{ij}^{(t)} = [\boldsymbol{\psi}_i,\ \boldsymbol{\phi}_{ij}^{(t)},\ H_i^{(t)}], \tag{4}$$

where $H_i^{(t)}$ is estimated for the spatial configuration of i and all detected boundaries, i.e., all edges in $\{V_{\text{on}}^{(t)} \cup i\}$. In this way, each $\boldsymbol{x}_{ij}^{(t)}$ captures the Gestalt cues (proximity, collinearity, co-circularity, parallelism, and symmetry) and Helmholtz grouping between the unlabeled i and the detected boundaries. For example, if i is close and collinear with any of the boundaries $j \in V_{\text{on}}^{(t)}$ then i is more likely to lie on an object boundary. The same holds, if $H_i^{(t)}$ reduces by adding i to $V_{\text{on}}^{(t)}$. The goal of SLEDGE is to learn the relative significance of these cues to boundary detection.

Each descriptor x_{ij} in $\boldsymbol{X}_i^{(t)}$ is fed separately to the classifiers and assigned a label, as specified in 8. Edge i receives the highest-confidence label across $\boldsymbol{X}_i^{(t)}$.

4.2.2 Two Ranking Functions

The SEARN framework requires a well-defined ordering of the input sequence of data $\boldsymbol{X}$. To this end, we specify two alternative ranking functions, aimed at selecting an edge to be labeled at every step $t = 1,\ldots,n$, such that its labeling reduces uncertainty about the occurrence of object boundaries in the image. Since edge descriptors are computed with respect to the previously detected boundaries, the edges selected early on facilitate the subsequent labeling of more ambiguous ones.

The first ranking function, R_1, is based on the heuristic assumption that longer and more salient contours are more informative about objects and their layout in the scene than smaller edges. Therefore, R_1 uses graph G and its structure to order the edges breadth-first, i.e., by their length. Sibling edges are ordered by their degree of saliency, $\boldsymbol{\psi}_{i1}$, defined in Sec. 3.2.1. At labeling step t, the descriptors $\boldsymbol{X}_i^{(t)}$ of the largest and most salient edge among the unlabeled ones, $i \in V_{\text{un}}^{(t)}$, are computed, as explained in Sec. 4.2.1. Edge i is then classified with the highest-confidence label across $\boldsymbol{X}_i^{(t)}$.

The second ranking function, R_2, is based on the classifier confidence in labeling the edges. R_2 avoids the heuristic assumptions of R_1. The main difference is that R_2 does not select an edge first and then computes its descriptors. Rather, at each t, it first computes and classifies the descriptors $\boldsymbol{X}_i^{(t)}$ of *all* unlabeled edges, and then selects $\boldsymbol{x}_{ij}^{(t)}$ whose labeling is characterized by the highest confidence. This, in turn, simultaneously selects and labels the corresponding edge i in step t. Note that R_2 is computationally more expensive than R_1, since for all nodes $i \in V_{\text{un}}^{(t)}$, we need to re-compute all $\boldsymbol{X}_i^{(t)}$ in every step t. We reduce the complexity of computing $\boldsymbol{X}_i^{(t)}$ by precomputing graph G and edge properties $\boldsymbol{\psi}_i$ and $\boldsymbol{\phi}_{ij}$ in Step 1.

4.2.3 Two Loss Functions

The SEARN framework requires that the loss function be specified to learn the optimal policy f, which will minimize the expected total loss over all loss-sequences of sequential edge labeling in all training images. The optimal policy, in our case, should produce a sequence of edge labels that is equivalent to the manually annotated sequence. In this subsection, we define two alternative loss functions.

The first loss function that we use is the standard Hamming loss that counts the total number of individual differences between the predicted output and ground-truth labels in the two sequences, $\boldsymbol{Y} = (y_1,\ldots,y_n)$ and $\hat{\boldsymbol{Y}} = (\hat{y}_1,\ldots,\hat{y}_n)$, as

$$L_H(\boldsymbol{Y},\hat{\boldsymbol{Y}}) = \sum_{i=1}^{n} \mathbf{1}\left[y_i \neq \hat{y}_i\right], \tag{5}$$

where $\mathbf{1}$ is the indicator function.

The Hamming loss favors correct labeling of all edges, since an error made at any edge carries the same relative weight. This may direct the learning algorithm to try to correctly label numerous small, non-salient edges, while ignoring some long and salient ones. To address this problem, we specify another loss function, L_F, that uses the F-measure of recall and precision associated with a specific edge. F-measure is one of the most common performance measures for boundary detection [51]. It is defined as the harmonic mean of recall and precision of all image pixels that are detected as lying along an object boundary. A large value of $F(\boldsymbol{Y}, \hat{\boldsymbol{Y}})$ indicates good precision and recall, and corresponds to a low loss. Thus, we specify

$$L_F(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = 1 - F(\boldsymbol{Y}, \hat{\boldsymbol{Y}}). \tag{6}$$

Note that L_H coarsely counts errors at the edge level, while L_F is estimated finely at the pixel level.

4.2.4 Majority Voting of Classifier Decisions

SEARN iteratively learns the optimal policy f, as in (2), which represents a probabilistic mixture of classifiers $\{h^{(\tau)}\}$ $\tau = 2, 3, \ldots, T$. A new $\boldsymbol{X}$ is sequentially labeled by probabilistically sampling one classifier from $\{h^{(\tau)}\}$, according to the multinomial distribution, parameterized by $\{\alpha_T^{(\tau)}\}$, defined in (3). We modify this classification scheme. We run all classifiers and weight their decisions by the corresponding $\{\alpha_T^{(\tau)}\}$. The heaviest vote is our outcome. Voting decisions of several classifiers has been shown to improve performance and reduces overfitting of each individual classifier [37, 17, 23].

After T learning iterations, SLEDGE estimates the confidence $P(f(\boldsymbol{X}_i){=}y_i)$ that edge descriptor $\boldsymbol{X}_i$ receives label $y_i \in \{0, 1\}$ as

$$P(f(\boldsymbol{X}_i){=}y_i) = \sum_{\tau=2}^{T} \kappa \alpha_T^{(\tau)} P(h^{(\tau)}(\boldsymbol{X}_i){=}y_i), \tag{7}$$

where the constant κ re-scales the α's, such that $\kappa \sum_{\tau=2}^{T} \alpha_T^{(\tau)} = 1$. Also, $P(h^{(\tau)}(\boldsymbol{X}_i){=}y')$ is the confidence of classifier $h^{(\tau)}$ when predicting the label of edge i. For example, when using the classical decision trees as the basic classifier family for h, this confidence can be defined as the percentage of elements with label y' in the leaf nodes of the decision tree, where all descriptors $\boldsymbol{X}_i = \{\boldsymbol{x}_{ij}\}$ have fallen. When using an SVM classifier, one can define the confidence of a particular instance to be proportional to its margin, i.e. its distance to the decision boundary. Note that $P(f(\boldsymbol{X}_i){=}0) + P(f(\boldsymbol{X}_i){=}1) = 1$.

Finally, SLEDGE classifies each edge descriptor $\boldsymbol{X}_i$ as

$$y_i = \arg\max_{y'} P(f(\boldsymbol{X}_i){=}y'). \tag{8}$$

Note that when the confidence $P(f(\boldsymbol{X}_i){=}y_i)$ is relatively low, then the ranking function R_2, described in Sec. 4.2.2, will defer the labeling of edge i, and give advantage to the other edges to be labeled before i. This, in turn, will allow capturing new evidence for perceptual grouping that may eventually remove the uncertainty about i.

Remark: We run into the problem of highly unbalanced data, because the number of negative examples (i.e., background edges) exceeds positives ones (i.e., object boundaries). For training SLEDGE, we rebalance the data by using the standard procedure of under-sampling. We sample the majority class examples so that there are as many negatives as positives. In the data mining literature, this is considered the best technique for handling unbalanced data, compared to other alternatives [20].

SLEDGE with ranking R_2 is summarized in Alg. 2.

Algorithm 2: SLEDGE with Ranking R_2

```
Input  : Set of training images $\mathcal{I} = \{I_1, I_2 \ldots\}$;
         Extracted edges $\{V(I_1), V(I_2), \ldots\}$;
         Ground-truth labels $\{\hat{\boldsymbol{Y}}(I_1), \hat{\boldsymbol{Y}}(I_2), \ldots\}$;
         Loss-sensitive classifier $h$, and initial $h^{(1)}$;
         Loss function $l$; Interpolation constant $\beta = 0.1$;
         Maximum number of iterations $T$
Output : Learned policy $f^{(T)}$
Initialize: $V_{\text{un}}^{(1)} = V$;
for $\tau = 1, \ldots, T$ do
    Initialize the set of descriptor sequences $\mathcal{X} = \emptyset$;
    for all $I \in \mathcal{I}$ do
        $V = V(I)$; $n = |V(I)|$; $\hat{\boldsymbol{Y}} = \hat{\boldsymbol{Y}}(I)$;
        for $t = 1, \ldots, n$ do
            for $i \in V_{un}^{(t)}$ do
                Compute $\boldsymbol{X}_i^{(t)} {=} \{\boldsymbol{x}_{ij}^{(t)} {:} j {\in} V_{\text{on}}^{(t)}\}$ as in (4);
                Compute $y_i = f^{(\tau)}(\boldsymbol{X}_i)$ as in (2), (8);
            end
            Select $i$ from $V_{\text{un}}^{(t)}$ with max confidence in $y_i$;
            Add $y_i$ to $\boldsymbol{Y}^{(\tau)}$;
            $V_{\text{un}}^{(t)} \leftarrow V_{\text{un}}^{(t)} \setminus \{i\}$;
        end
        Estimate loss $L(\boldsymbol{Y}^{(\tau)}, \hat{\boldsymbol{Y}})$;
        Add the estimated descriptor sequence $\boldsymbol{X}$ to $\mathcal{X}$;
    end
    Learn a new classifier $h^{(\tau+1)} \leftarrow h(\mathcal{X}; L)$;
    Interpolate: $f^{(\tau+1)} = \beta h^{(\tau+1)} + (1 - \beta) f^{(\tau)}$
end
Return $f^{(T)}$ without $h^{(1)}$.
```

5. Detecting Salient Edges

This section presents Step 1 of our approach, aimed at extracting salient edges in a given image. Step 1 may use any available edge detector, as long as it yields high recall of true boundaries. Our subsequent steps are expected to improve precision by labeling a subset of the detected edges as boundaries.

While we are aware of the many powerful edge detectors that exist in the literature, our goal in this paper is not to an-

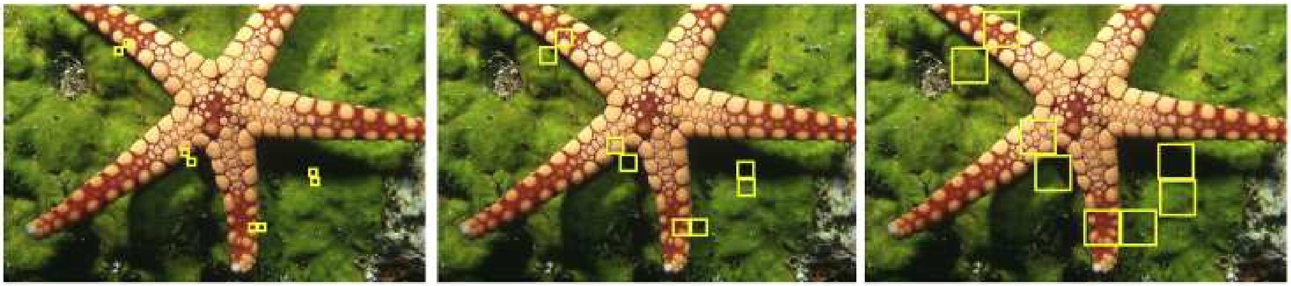


Figure 8. **Texture variations at different scales.** Original image with pairs of windows at different scales overlapped in yellow. For each pair of windows around pixel q, we examine their texture properties to evaluate the probability of q to lie on a salient edge. Left to right are examples of windows of size 11×11, 21×21 and 41×41.

alyze all of them. Instead, we aim to show that any detector that produces edges with high recall is good enough for our labeling algorithm to outperform the state of the art on the object boundary detection task.

For this reason, we choose to test SLEDGE on edges obtained with: (a) the Canny edge detector [11], (b) our texture-based edge detector that we will describe in the subsequent paragraph, and (c) the state-of-the-art gPb detector [49]. Our intention is to cover a reasonable range of edge detectors, from the simplest intensity-based Canny detector, via an intermediate texture-based detector, to the most complex gPb detector. In the following, we present our texture-based edge detector, tPb. We then compare its performance with the state-of-the-art boundary detectors on the BSD.

Building the edge probability map tPb. A salient edge separates distinct image textures. Since real-world surfaces are typically characterized by unique texture, salient edges are more likely to represent boundaries than other edges in the image. Our goal is thus to robustly estimate for each pixel its probability of lying on a salient edge. This probability is large if image textures differ in the vicinity of a particular pixel, or small, otherwise.

We analyze texture variations in a pixel's neighborhood at different scales, as illustrated in Fig 8. Given pixel q, we randomly place a large number of pairs of windows in the image, $\{(r_1^k, r_2^k)\}$, $k = 1, 2, \ldots, K$, so that one side of every window contains q. Each pair of windows defines two image areas whose texture properties are compared to estimate the texture variation at q. We use the unnormalized variance, $\varepsilon^2(r)$, of pixel values within each window, r, to characterize its texture, $\varepsilon^2(r) = \sum_{p\in r}(p - \overline{r})^2$, where p denotes pixels in r, and $\overline{r}$ denotes their mean value. Texture difference between a pair of windows, $\Delta(r_1, r_2)$, is estimated, as in [65, 13], as the compression error

$$\Delta(r_1, r_2) = |\varepsilon^2(r_1 \cup r_2) - \varepsilon^2(r_1) - \varepsilon^2(r_2)| \ . \qquad (9)$$

After randomly placing K pairs of windows, the probability that pixel q belongs to a salient edge is estimated as

$$P(q \in \text{edge}) = \frac{1}{K}\sum_{k=1}^{K}\Delta(r_1^k, r_2^k) \qquad (10)$$

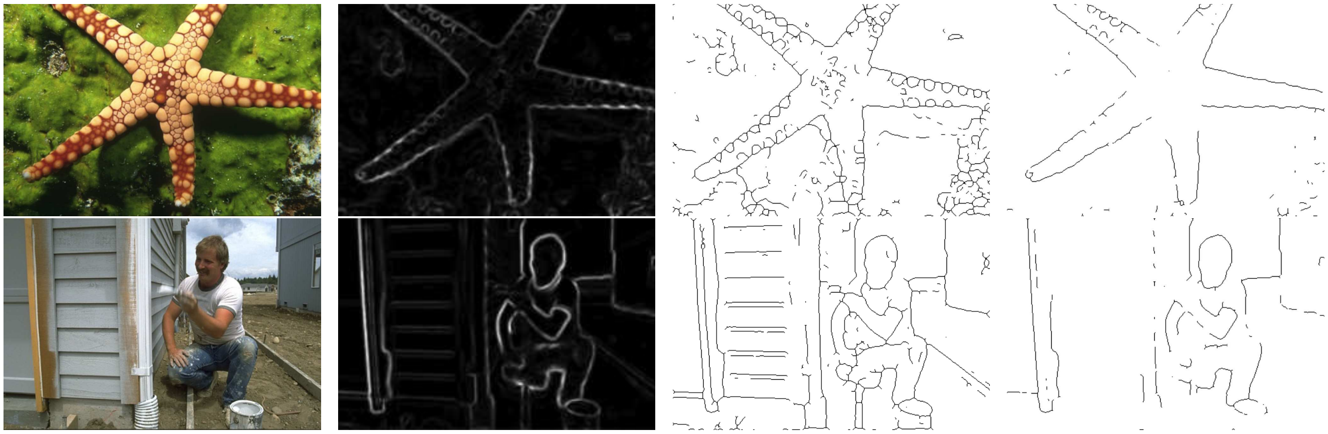
These probabilities are computed for each pixel and assembled into a probability map of edge saliences, tPb. The saliency of a particular edge i, ψ_{i1} (Sec. 3.2.1), is defined as the mean value of tPb along the edge i, $\psi_{i1} = \text{mean}_i(\text{tPb})$.

Since a salient edge may occur at any image location, and it may separate textures occupying arbitrarily large image areas, the window locations and sizes are sampled from the uniform distribution. The computation of $P(q \in \text{edge})$ is efficient since $\varepsilon^2(r)$ can be computed in $O(1)$ using integral images.

Fig. 9(b) shows two examples of the probability map of edge saliences tPb. This map is thresholded at all probability levels and edges extracted using a standard non-maximum suppression technique. Fig. 9(c,d) shows edges extracted at two different probability levels. These multi-scale edges are then assembled in the graph of edges G that captures the recursive embedding of smaller edges within larger ones, as well as their neighbor relationships (G is presented in Sec. 3).

Evaluating tPb. For each pixel q, we place $K = 20$ pairs of windows (r_1, r_2) and we pick at random a window size (from 5×5 to 41×41) and an orientation (from $0°$ to $360°$). We can then compute the probability that q lies on an edge with (10).

To evaluate the performance of tPb on the object boundary detection task, we adopt the methodology developed by Martin et. al [51]. This framework considers two aspects of detection performance. Precision P measures the fraction of true positives in the edges produced by a detector. Recall R measures the fraction of ground-truth boundaries detected. For detectors like tPb, that have real-valued outputs, one obtains a curve parameterized by detection threshold. The global F-measure, defined as the harmonic mean of precision and recall, $F = \frac{2PR}{(P+R)}$, provides a useful summary score for the algorithm.



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

Figure 9. **From edge probability maps to individual edges.** (a) Original image. (b) Probability map of edge saliences, tPb. We threshold this map for all gray levels τ and use non-maximum suppression to extract individual edges. Here, we illustrate two such levels, $\tau = 25$ (c), and $\tau = 75$ (d). The multiscale edges are then organized in a graph G that captures the recursive embedding of smaller edges within larger ones, as well as their neighbor relationships.

In Fig.10, we compare the performance of tPb to that of Canny [11], Felz06 [21], Zhu07 [72], Ren08 [56], Mairal08 [48] and gPb [49] on the BSD. We see that in the low precision, high recall regime, our texture-based edge detector, using only simple image features and no learning, compare favorably to other approaches, that use a battery of sophisticated features and learning. In particular, note that after point A, our algorithm has better precision than all other techniques.

Although tPb produces edges with lower precision than most techniques, note that there is no need to produce edge maps with high precision, for our purposes. We only need the extraction of salient edges to be fast, and have high recall of true boundaries. We will then delegate the responsibility of correcting errors, i.e. improving precision, to SLEDGE. Fig. 10 shows that with tPb, we are able to obtain very large recall for a relatively large range of thresholds (from 0 to 0.3), i.e. most of the ground-truth boundaries are present in the set of edges which are input to SLEDGE.

6. Results

This section presents qualitative and quantitative evaluation of our approach on images from the BSD [50, 4], Weizmann Horses [7], and LabelMe [59] datasets. BSD300 (resp. BSD500) consists of 300 (resp. 500) natural images, manually segmented by a number of different subjects. The Weizmann Horses dataset consists of 328 side-view images of horses that are also manually segmented. For the LabelMe dataset, we select the 218 annotated images of the Boston houses 2005 subset. The challenges of these datasets have been extensively discussed in the past, and include, but are not limited to, clutter, illumination variations, occlusion.

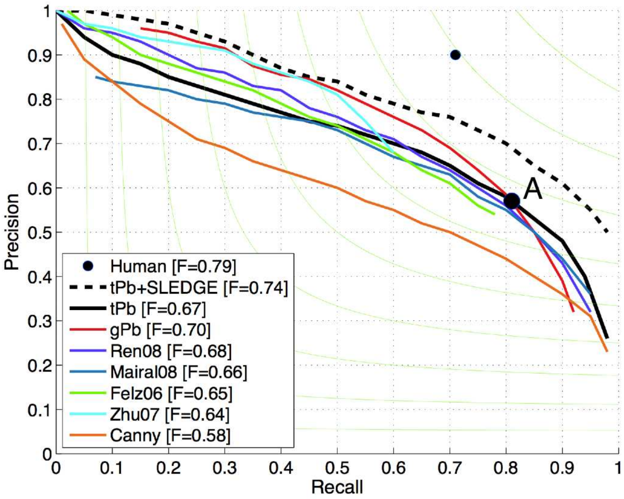


Figure 10. **Evaluating edge detection on the Berkeley dataset [50].** Leading approaches to edge detection are ranked according to their F-measure (harmonic mean of precision and recall) with respect to human ground truth. The green lines are level sets of the F-measure, ranging from 0.1 (lower left) to 0.9 (upper right). Our method performs similarly to the state of the art approaches (gPb [49], Ren08 [56], Mairal08 [48], Felz06 [21], Zhu07 [72]), and achieves a maximum F-measure of 0.67 on this dataset. Note that for high recall values, and particularly after point A, our texture-based edge detector has the highest precision. The dashed curve presents our tPb+SLEDGE results for boundary detection, not edge detection.

Below, we present our default training and testing setups.

Training. We train SLEDGE on the 200 training images of BSD300. For each image, we compute the edge probability map tPb and extract multiscale edges, as described in Sec. 5. We build the graph of image edges G and compute the attributes of all nodes and arcs. We also convert the manually annotated object boundaries to ground truth labels for all edges in the training images. Our initial classifier $h^{(1)}$ is a fully grown C4.5 decision tree, that chooses the split attribute based on the normalized information gain criterion. The attributes considered for $h^{(1)}$ are only the intrinsic parameters ψ_i introduced in Sec. 3.2.1. In further iterations of SLEDGE, the classifiers are pruned C4.5 decision trees, which are learned on all attributes ψ_i, ϕ_{ij} and H_i, as discussed in Sec. 4.2.1. C4.5 pruning uses the standard confidence factor of 0.25. The interpolation constant β, mentioned in Sec. 4.1, is set to $\beta = 0.1$. Also, we use R_2 as the ranking function, L_F as the loss function, and voting to combine the output of the decision tree classifiers.

Testing. Given a new test image, we compute tPb and extract multiscale edges. We build the graph of edges G, and let SLEDGE sequentially label all edges in G. Performance is measured as the average precision and recall of the boundary map produced by SLEDGE, as defined in [51].

In the following two subsections, we first test SLEDGE on the three datasets, and show that we compare favorably to the state of the art. Then, we evaluate specific aspects of our approach by introducing variations to the aforementioned default setup. These variations concern using different: initial conditions, edge labeling strategy, classifier type, interpolation scheme, ranking function, and loss function.

6.1. Object Boundary Detection by SLEDGE

Qualitative evaluation: Figures 11, and 12 illustrate the order in which SLEDGE selects edges to label in two example images of BSD. We refer the reader to the respective captions for comments on the underlying Gestalt principles that SLEDGE uses for labeling.

Fig. 15 demonstrates high accuracy of our boundary detection on the BSD. Detection is good even in the following cases: (i) when objects have complex textures, e.g., coral in Fig. 15(c) and ground in Fig. 15(b); (ii) when the boundaries are blurred or jagged, e.g., in Fig. 16 (d); and (iii) when boundaries form complex layouts, e.g., in Fig. 16(c). Filter-based methods, or approaches based on image decimation and smoothing would typically fail to accurately delineate topologically complex spots in which several texture boundaries meet. We are also able to detect boundaries where there is no strong gradient, which gPb [49] typically fails to extract, e.g., see the edge that separates the two cows in Fig. 16(a).

We compare our algorithm with a related method, presented in [40], and mentioned in Sec. 2. Fig. 17 shows two illustrative examples, where the 50 most confident boundaries are color-coded in HSV space (red means more confident). SLEDGE finds prominent boundaries that [40] fails to detect, e.g., see the back of the kangaroo. On the other hand, SLEDGE labels the edges in the rock texture on the left of the animal as boundaries, whereas [40] does not. A likely reason we fail on these texture edges is that they continue each other well, and thus reinforce each other to be jointly labeled as boundaries.

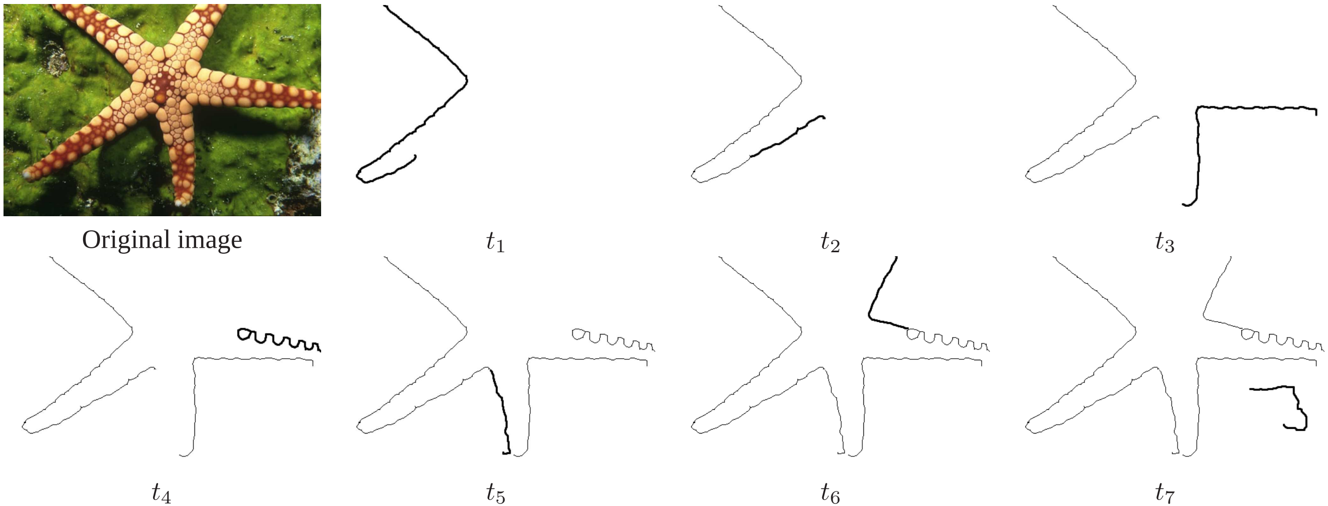
Fig. 18 illustrates cases when SLEDGE labels edges incorrectly. For example, in Fig. 18(top row), the shadow on the horse's neck is classified as boundary, because its edge saliency ψ_{i1} is high, and also because it would be a valid continuation of the front leg. Similarly, in Fig. 18(bottom row), the shadow on the paved area generates a set of collinear edges. In both cases, gPb [49] also characterizes these edges by a high probability of being on a boundary. Note that SLEDGE accurately assigns high confidence to the boundaries of the electric pole, unlike gPb [49].

Quantitative evaluation: SLEDGE computes for each edge i the confidence that it lies on a boundary. This produces a boundary probability map that we use to evaluate our performance. To this end, we again use the boundary evaluation methodology presented in [51].

First, we examine the boundary map output by SLEDGE, and compare it to the initial edge probability map. We use edges detected with the same algorithm for training and for testing. Fig. 13 compares the edges obtained with (a) Canny detector [11], (b) our tPb detector and (c) gPb detector [49], before and after they have been labeled by SLEDGE. Sequential labeling significantly improves performance for all edge detectors, especially in the high-recall-low-precision regime. This demonstrates that SLEDGE does not require a very good initial set of edges for good performance. We note however that the higher the precision of original input edges, the higher the precision of output boundaries. For example, Fig. 13 shows that the precision of gPb+SLEDGE is higher than the precision of Canny+SLEDGE. Interestingly, our tPb edge detector combined with SLEDGE is as good as the gPb detector combined with SLEDGE. This demonstrates that our approach can successfully work with noisy input edges, most of which are not object boundaries, as long as the input set contains a high recall of boundaries.

Fig. 14 compares the average precision and recall of tPb+SLEDGE with gPb of [49] on BSD, Weizmann Horses and LabelMe. SLEDGE is trained on 200 training images from BSD. Results in Fig. 14 are reported as the average performance of SLEDGE on 100 test images from BSD, and on all images from the Weizmann and LabelMe datasets. As can be seen, our technique outperforms gPb on all three datasets.

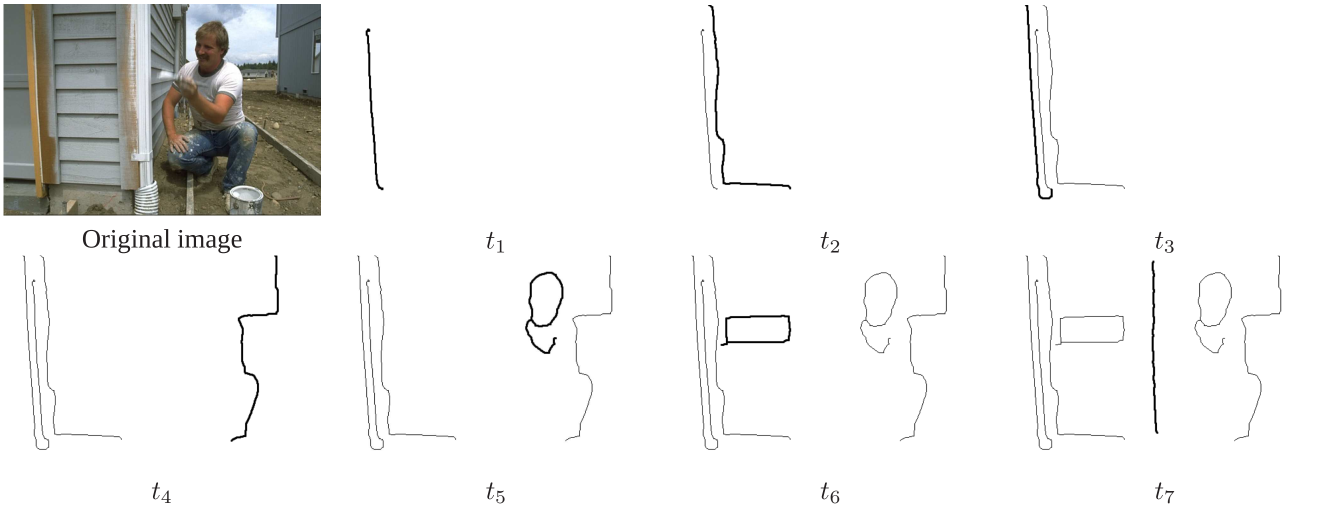
Running-time: Training SLEDGE on 200 BSD training images takes about 24 hours, on a 2.66GHz, 3.4GB



Original image t_1 t_2 t_3

t_4 t_5 t_6 t_7

Figure 11. **Sequential labeling of edges by SLEDGE.** We illustrate a few iterations of SLEDGE, and the order in which object boundaries are selected at times $t_1 < t_2 < t_3 < \ldots < t_T$. For visibility, we do not show the intermediate edges that are classified as non-boundaries. At time t_1, SLEDGE selects a long, prominent edge. Then at time t_2, it picks a smaller edge that is close and collinear to the first boundary. Thus, SLEDGE uses the principle of good continuation of edges at t_2. At t_3, t_4 and t_7, it labels other long edges. At times t_5 and t_6, we note that the selected edges are not the longest, nor the most salient ones. They are edges that connect other boundaries together (t_5), or that again continue well an existing boundary (t_6).



Original image t_1 t_2 t_3

t_4 t_5 t_6 t_7

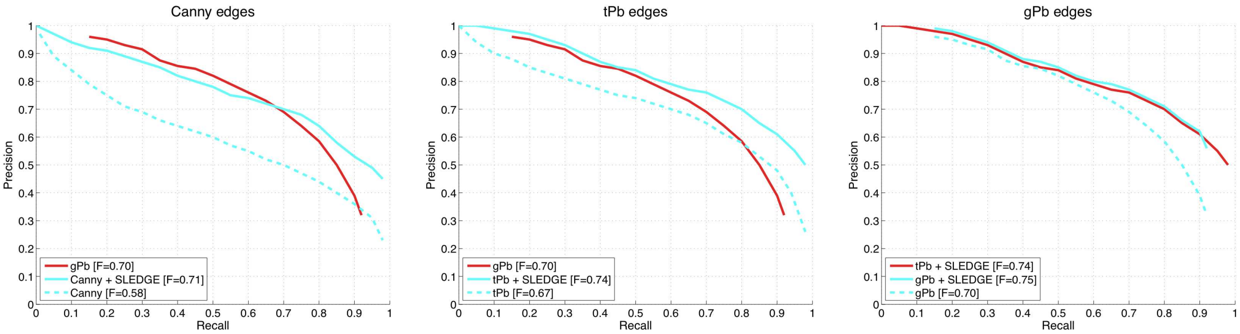
Figure 12. **Sequential labeling of edges by SLEDGE.** See the caption of Fig. 11. At time t_1, SLEDGE selects a long, prominent edge. Then at times t_2, t_3 and t_7, it selects edges that are parallel to the first boundary. The grouping principles used for edge labeling at times t_4, t_5 and t_6 are not obvious.

RAM PC. Computing the boundary probability map by SLEDGE for a new image takes 20-40sec, depending on the number of edges in the image.

6.2. Evaluating Particular Aspects of SLEDGE

This section presents quantitative evaluation of how different design choices modify performance of SLEDGE.

Initial conditions. The early errors of SLEDGE in the sequence might be critical, because they could sequentially propagate to the entire sequence. To test this aspect of our algorithm, we compare our default setup with the following variant of SLEDGE. We create 100 different initial configurations by randomly picking b true boundaries as the initial set of correctly labeled edges. The randomization serves to eliminate any bias in picking certain boundaries (e.g., long ones). Then, we let SLEDGE continue labeling the remaining edges. The pairwise and global features are computed based on the initial set of randomly selected true boundaries. Finally, we compute recall and precision averaged over these 100 distinct initializations. These tests serve to evaluate SLEDGE performance when there is no error at the beginning of the labeling sequence. Table 1 shows the



(a) (b) (c)

Figure 13. **Improvement in boundary detection.** We compare boundary detection performance on BSD in two cases: outputs of existing edge detectors are input to SLEDGE (solid line), and outputs of existing edge detectors are simply declared as boundary detection without using SLEDGE (dashed line). The detectors include: (a) Canny [11], (b) tPb, and (c) gPb [49]. Training and testing are performed with the same type of edges, i.e. if we test SLEDGE on Canny edges, it means it has been trained with Canny edges. We see that SLEDGE is able to improve the precision of any initial set of edges. Even using Canny edges as input to SLEDGE gives higher precision higher than gPb of [49], for a large range of threshold levels in the high-recall-low-precision regime.

	$b = 2$	$b = 4$	$b = 6$
Increase in recall [%]	21 ± 2.3	33 ± 2.4	61 ± 1.84
Increase in precision [%]	23 ± 3.5	25 ± 1.9	49 ± 2.1

Table 1. **Initial conditions.** Increase in recall and precision in %, at equal error rate, relative to those of our default setup, averaged over 100 random initializations consisting of $b \in \{2, 4, 6\}$ true boundaries as the initial set of correctly labeled edges.

average increase in recall and precision, at equal error rate, relative to those of our default setup, for $b = 2, 4, 6$. As can be seen, when errors at the beginning of the labeling sequence are manually eliminated, our performance gain is relatively small. This suggests that our default setup is relatively insensitive to initial labeling errors.

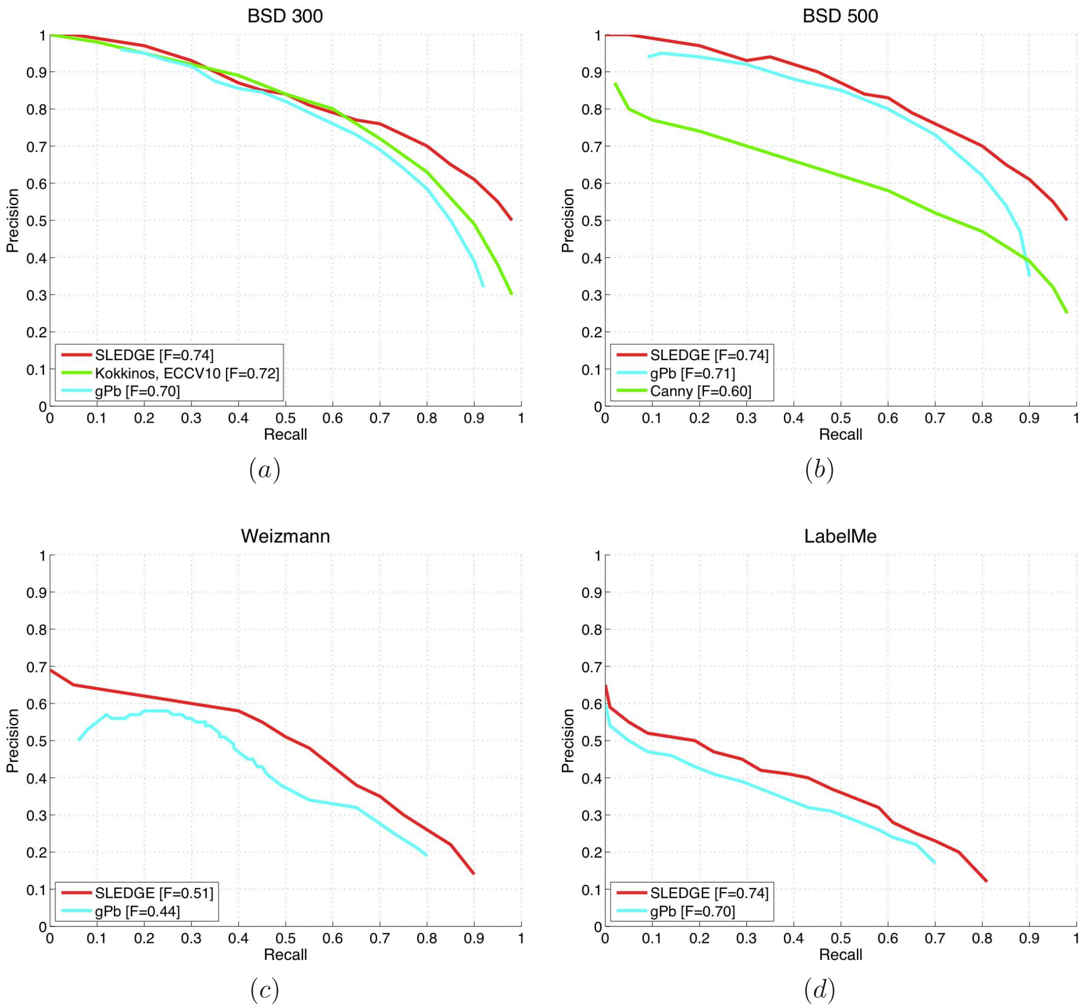
Edge labeling strategy. We test how performance varies when we allow SLEDGE to continue relabeling edges after all the edges have already been labeled once. Specifically, after all edges have been visited, we remove them from the labeled set and send them to the unlabeled set. The pairwise and global features are still computed based on the current labeling of boundaries. We iterate SLEDGE until it reaches convergence, i.e., no edge changes the label. On the 100 test images of the BSD300, at equal error rate, SLEDGE with relabeling improves by 0.2% in precision, and by 0.5% in recall, relative to the performance of the default SLEDGE with just one labeling pass. Thus, allowing edges to be relabeled, in our approach, only marginally improves precision and recall, but increases complexity. Therefore, it seems that the iterative relabeling of edges is not justified in our approach.

Layout entropy. We measure the influence of the layout entropy H on the performance of SLEDGE. We obtain that at equal error rate, the performance of SLEDGE with H improves by 3.2% in precision and by 4.1% in recall compared to the performance of SLEDGE without H.

Classifiers. From our experiments, a non-linear classifier is more suitable than a linear one for separating boundaries from background edges. Classification accuracy of SLEDGE with decision trees compared to that of SLEDGE with a linear SVM-type classifier is 91% vs. 72%. For training SVMs (as in the case of decision trees), we use a balanced number of positive and negative examples – the under-sampling procedure mentioned in Sec 4.2.4-Remark – with the complexity parameter $C = 1.0$. When more sophisticated classifiers are used, e.g. SVM with RBF and random forests, we observe only a relatively small gain in accuracy over the decision tree, which does not justify the increase in complexity. Indeed, the total computation time primarily depends on the complexity of the chosen classifier as all classifiers have to be run at each iteration of SLEDGE.

Ranking functions. Fig. 19 compares the performance of SLEDGE on BSD, for ranking function R_1 or R_2, described in Sec. 4.2.2. While R_1 selects the next edge in a breadth-first manner, R_2 ranks first the edge that the classifier is the most confident about. From Fig. 19, R_2 outperforms R_1. The edges that have already been labeled provide important perceptual-grouping cues for labeling of subsequent edges, and thus it is critical to initially label those edges which would be least harmful for later decisions, i.e., edges with the highest confidence in classification.

Loss functions. Fig. 20 shows the performance of SLEDGE when using the loss function L_H or L_F, specified in Sec. 4.2.3. L_H coarsely counts errors at the edge level, while L_F is a loss estimated finely at the pixel level. As expected, SLEDGE with L_F produces boundaries with slightly better precision than SLEDGE with L_H. L_F is



(a) (b)

(c) (d)

Figure 14. **Boundary detection.** Comparison of tPb+SLEDGE and gPb [49] for (a) BSD 300 [50], (b) BSD 500 [4], (c) Weizmann Horses [7] and (d) LabelMe [59]. For the BSD 300, we also compare SLEDGE to the method of [39]. We outperform gPb on all four datasets, and obtain a better F-measure than [39] on the BSD 300.

a more appropriate loss function for the boundary detection problem because unlike L_H, it favors the selection of long, salient edges. Since all descendants of an edge labeled as boundary are automatically labeled as boundary, the number of edges in the unlabeled set decreases much faster if long boundaries get labeled early on. We verify that SLEDGEwith L_F runs about 3 times faster than SLEDGE-with L_H.

Interpolation scheme. We stop learning new classifiers when the average performance on the training data does not change. For BSD, this happens after about 10 iterations. Fig. 21 compares the performance of SLEDGE with these 10 classifiers, as the interpolation scheme changes from sampling to weighted majority voting of the classifiers. As can be seen, voting significantly outperforms sampling.

In the following subsection, we evaluate the relevance of distinct Gestalt principles for boundary detection.

6.3. Relative Significance of Gestalt Laws

We are interested in how SLEDGE schedules (i.e., prioritizes) distinct Gestalt principles for edge labeling. To this end, we perform the following analysis: For each node u in a decision tree that corresponds to a split on the Gestalt rule $g_i \in \{\psi_{i2}, \phi_{i1}, \ldots, \phi_{i5}\}$, we count the number of its ancestors, k, that are also splits on the Gestalt rules. $k+1$ corresponds to the ranking of the splitting criterion g_i, i.e. the position of this rule in the schedule of all Gestalt rules. Note that some splits might not correspond to a Gestalt rule, e.g. splits on ψ_{i1}, the saliency of edge i. For each Gestalt rule g_i, we build a histogram of rankings of the nodes that split on g_i in all the decision trees that form the final classifier f. We report in Table 2 the relative rankings of each Gestalt principle.

This result could be used for estimating a prior distribution of scheduling the Gestalt principles for edge labeling

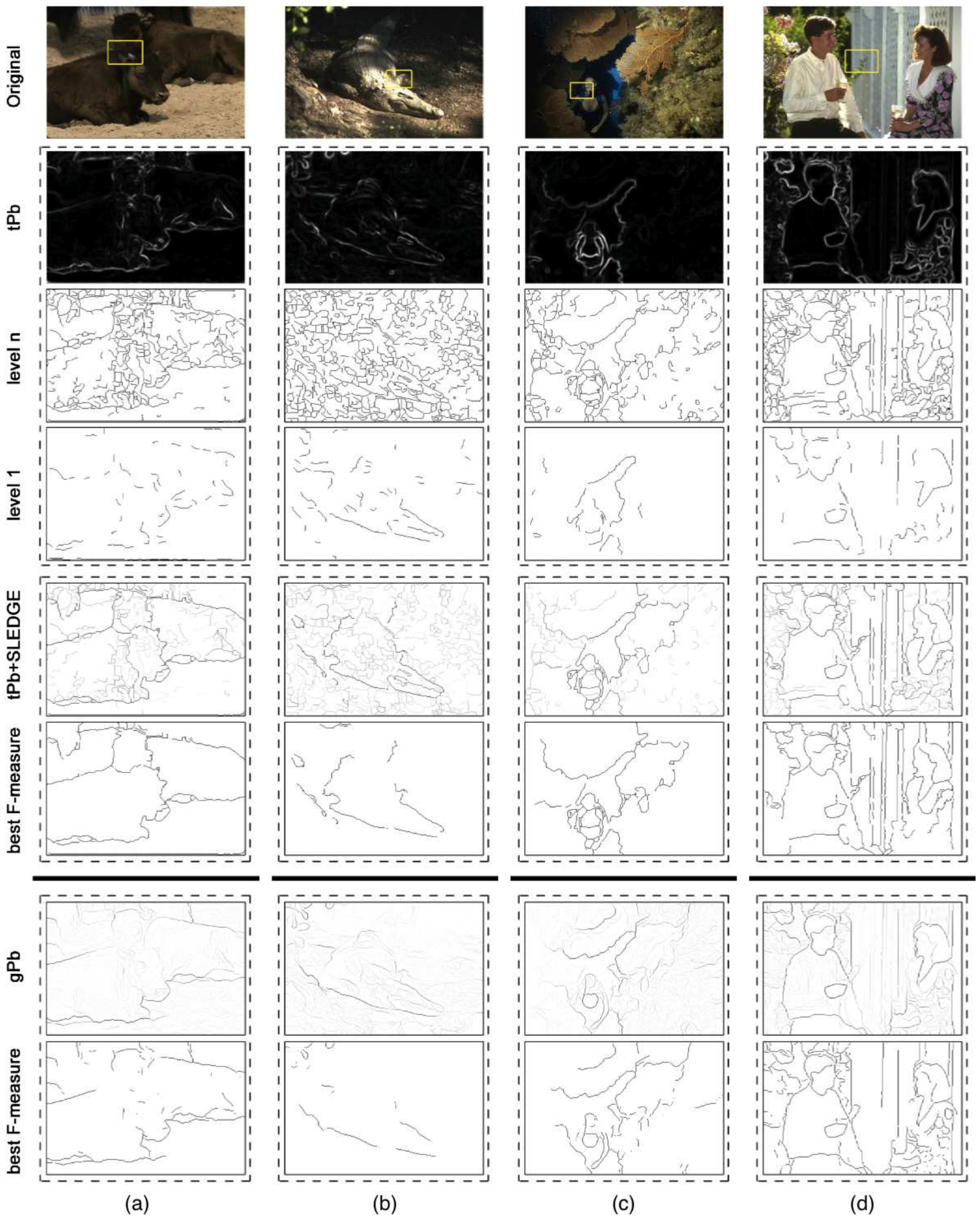


Figure 15. **Boundary detection on BSD.** From top to bottom: Original image, Edge probability map tPb, Edges corresponding to the leaves of the edge graph (the finest-scale), Edges corresponding to the roots of the edge graph (the coarsest scale), Boundary map output by SLEDGE, Boundaries detected by SLEDGE for best F-measure, Boundary map output by gPb [49], Boundaries detected by gPb for best F-measure. A zoom-in of the windows highlighted in the original images is available in Fig. 16. SLEDGE outperforms gPb on challenging object boundaries amidst texture or low-contrast ramps, for example: the cow's head in (a); the tip of the alligator's mouth in (b); the green coral in (c); and the details in the dress of the woman in (d).

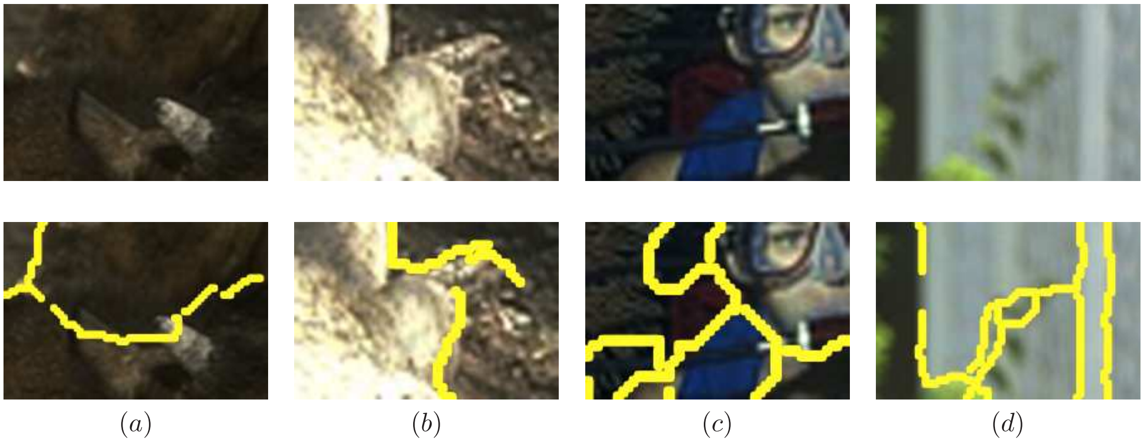


Figure 16. **Zoomed-in parts.** Top row: Zoom-in of the windows that are marked in Fig. 15. Bottom row: Boundaries detected by SLEDGE for best F-measure are overlaid onto the original images.

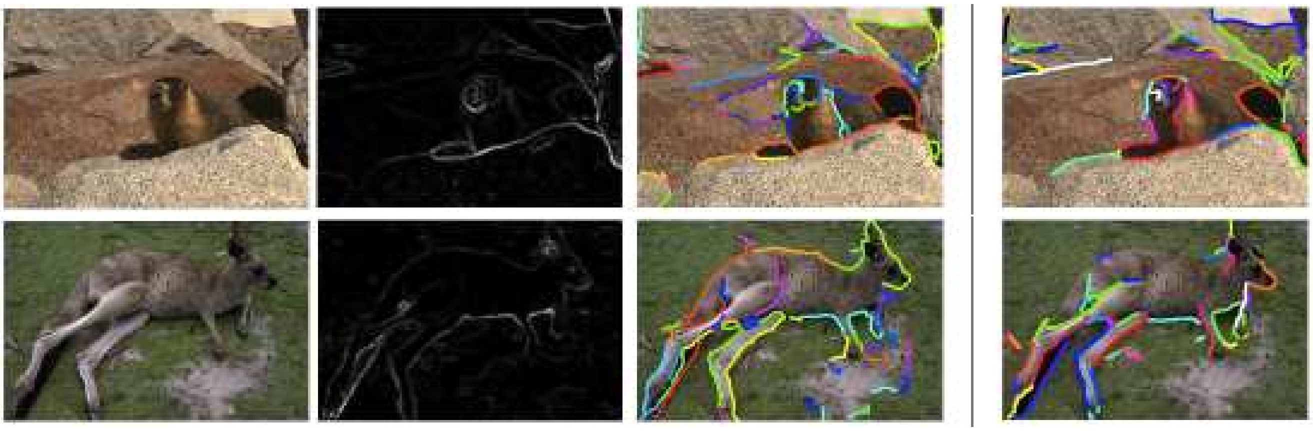


Figure 17. **Strength of boundaries.** From left to right: Original image, Edge probability map tPb, 50 most confident boundaries output by SLEDGE, 50 most confident boundaries output by the approach of [40]. The confidence is color-coded in HSV space, where red means very confident, and purple and magenta mean less confident. Best viewed in color.

(not used in this paper). Note that the results in Table 2 depend on our particular formulations of the Gestalt rules, which are common in the literature.

7. Conclusion

We have presented an approach to boundary detection in arbitrary images. The approach takes as input salient edges, and classifies these edges using a new sequential labeling algorithm, called SLEDGE. SLEDGE learns how to optimally combine Gestalt grouping cues and intrinsic edge properties for boundary detection. In addition to the common pairwise grouping cues, we have also formalized the Helmholtz global principle of grouping, as the entropy of the edge layout.

Our empirical evaluation demonstrates that the approach outperforms the state-of-the-art boundary detectors regardless of the input set of salient edges, as long as the set contains a high recall of true boundaries. We have observed that SLEDGE tends to favor good continuation of strong edges, which works well in most cases, but fails when there is an accidental alignment of object and background edges (e.g., shadows). Our results demonstrate that proximity and collinearity of edges are typically scheduled before the other Gestalt principles of grouping for boundary detection.

Two key novel aspects of our approach lead to good performance—namely, reasoning about boundaries: (1) investigates a data-driven sequence of distinct image locations, rather than scans uniformly all image locations, and (2) takes into account both local and global perceptual organization of salient edges, rather than small image patches.

References

[1] N. Ahuja and S. Todorovic. Learning the taxonomy and models of categories present in arbitrary images. In *ICCV*, 2007. 2

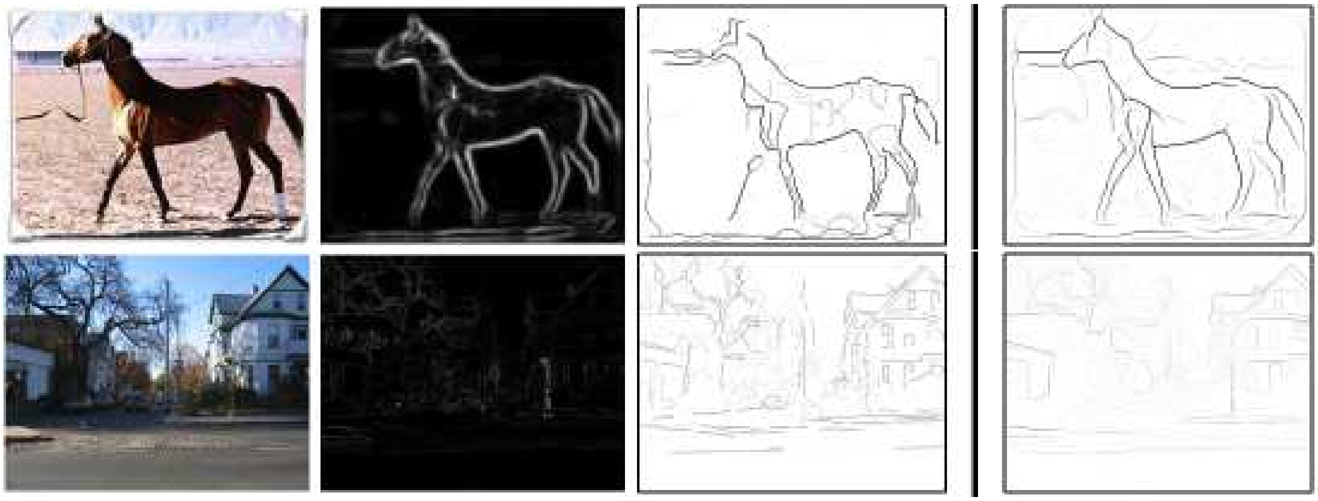


Figure 18. **Boundary detection on the Weizmann and LabelMe datasets.** From left to right: Original image, Edge probability map tPb, Object boundary map output by SLEDGE, Object boundary map output by gPb [49].

Rank	Proximity	Collinearity	Co-Circularity	Parallelism	Symmetry	Repetition
1	0.44	0.29	0.09	0.13	0.02	0.03
2	0.28	0.35	0.11	0.14	0.01	0.11
3	0.09	0.13	0.14	0.22	0.19	0.23
4	0.12	0.11	0.20	0.27	0.10	0.20
5	0.07	0.12	0.14	0.31	0.16	0.20

Table 2. **Scheduling of Gestalt rules.** We record how many times each Gestalt rule was used first, second, etc., in the scheduling of the Gestalt rules in the decision trees which form the final classifier f in SLEDGE. The rule that is used the most as the first splitting rule is the Proximity criterion, then it is the Collinearity criterion, and later on there is a shift toward more global rules, such as Parallelism, Symmetry and Repetition. There is no clear distinction as to which rule among Parallelism, Symmetry and Repetition ranks earlier.

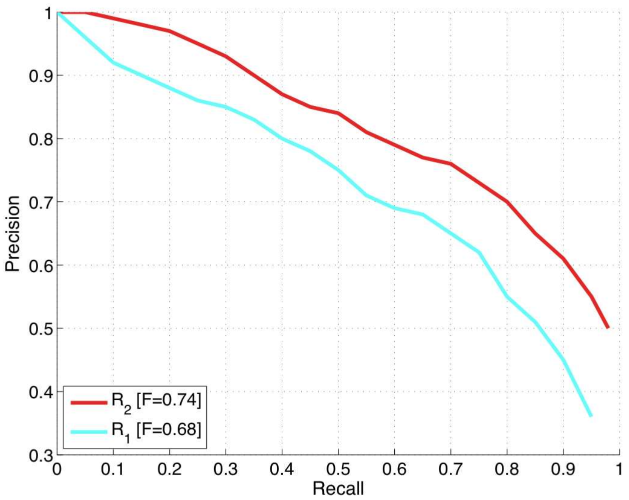


Figure 19. **Ranking functions.** We evaluate the edges labeled by SLEDGE on BSD when using ranking function R_1 or R_2. Ordering the edges according to the classifier confidence (R_2) produces better results than when the ordering is fixed (R_1).

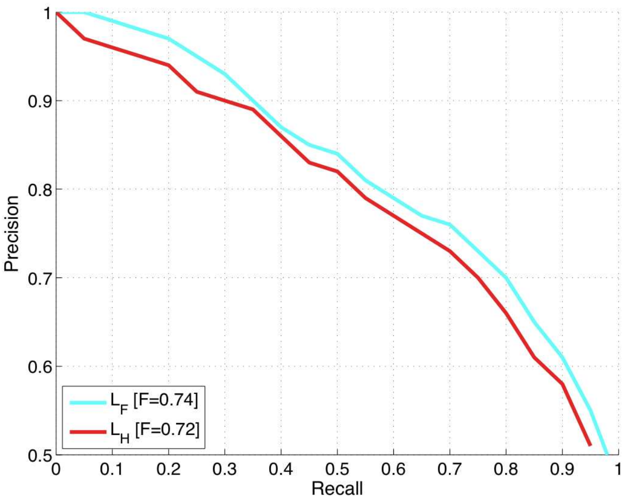


Figure 20. **Loss functions.** Performance of SLEDGE on BSD when using the loss function L_H or L_F, given by (5) and (6). L_H coarsely counts errors at the edge level, while L_F is a loss estimated finely at the pixel level. As expected, SLEDGEwith L_F improves the performance of SLEDGEwith L_H.

[2] N. Ahuja and S. Todorovic. Connected segmentation tree – a joint representation of region layout and hierarchy. In *CVPR*,

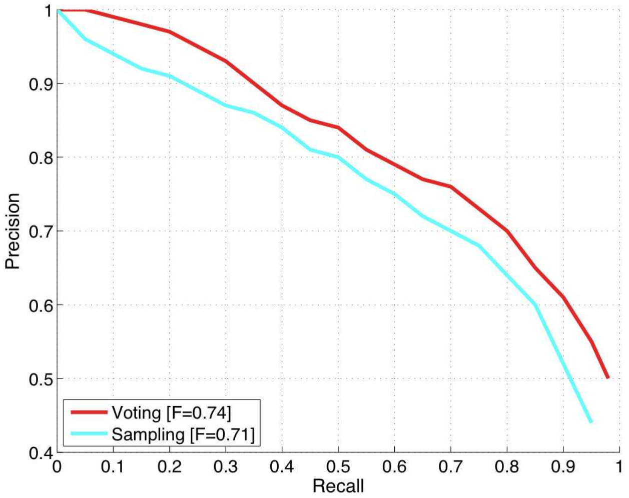


Figure 21. **Interpolation schemes.** Performance of SLEDGE on BSD when using sampling or voting. Precision significantly increases when voting is used vs. sampling.

2008. 8

[3] P. Arbelaez. Boundary extraction in natural images using ultrametric contour maps. In *POCV*, page 182, 2006. 3

[4] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *TPAMI*, 99(RapidPosts), 2010. 1, 2, 13, 17

[5] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *TPAMI*, 24(4):509–522, 2002. 7

[6] I. Biederman. Surface versus edge-based determinants of visual recognition. *Cognitive Psychology*, 20(1):38–64, 1988. 1

[7] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *ECCV*, volume 2, pages 109–124, 2002. 2, 13, 17

[8] G. Borgefors. Hierarchical Chamfer matching: A parametric edge matching algorithm. *TPAMI*, 10(6):849–865, 1988. 1

[9] C. R. Brice and C. L. Fennema. Scene analysis using regions. *Artificial Intelligence*, 1:205–226, 1970. 1

[10] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Seventh International World-Wide Web Conference (WWW 1998)*, 1998. 7

[11] J. Canny. A computational approach to edge detection. *TPAMI*, 8(6):679–698, 1986. 3, 12, 13, 14, 16

[12] J. M. Coughlan and A. L. Yuille. Bayesian A* tree search with expected o(n) node expansions: applications to road tracking. *Neural Comput.*, 14(8):1929–1958, 2002. 4

[13] Y. Deng and B. S. Manjunath. Unsupervised segmentation of color-texture regions in images and videos. *TPAMI*, 23(8):800–810, 2001. 12

[14] A. Desolneux, L. Moisan, and J. Morel. Edge detection by Helmholtz principle. *J. Mathematical Imaging and Vision*, 14(3):271–284, 2001. 5

[15] A. Desolneux, L. Moisan, and J.-M. Morel. Meaningful alignments. *IJCV*, 40(1):7–23, 2000. 5

[16] A. Desolneux, L. Moisan, and J.-M. Morel. A grouping principle and four applications. *TPAMI*, 25(4):508–513, 2003. 5

[17] T. G. Dietterich. Ensemble methods in machine learning. In *Lecture Notes in Computer Science*, pages 1–15, 2000. 11

[18] P. Dollar, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. In *CVPR*, pages 1964–1971, 2006. 3

[19] M. Donoser, H. Riemenschneider, and H. Bischof. Linked edges as stable region boundaries. In *CVPR*, 2010. 1, 3

[20] C. Drummond and R. C. Holte. C4.5, class imbalance, and cost sensitivity: why under-sampling beats over-sampling. In *Workshop on Learning from Imbalanced Datasets II*, 2003. 11

[21] P. Felzenszwalb and D. McAllester. A min-cover approach for finding salient curves. In *POCV*, 2006. 4, 13

[22] V. Ferrari, F. Jurie, , and C. Schmid. From images to shape models for object detection. *IJCV*, 87(3):284–303, 2010. 1

[23] Y. Freund, Y. Mansour, and R. E. Schapire. Why averaging classifiers can protect against overfitting. In *Proceedings of the 8th International Workshop on Artificial Intelligence and Statistics*, 2001. 11

[24] A. Fridman. Mixed markov models. *Proceedings of the National Academy of Sciences*, 100(14):8092–8096, 2003. 4

[25] M. Galun, R. Basri, and A. Brandt. Multiscale edge detection and fiber enhancement using differences of oriented means. In *ICCV*, pages 1–8, 2007. 3

[26] D. Geman and B. Jedynak. An active testing model for tracking roads in satellite images. *TPAMI*, 18(1):1–14, 1996. 4

[27] M. A. Greminger and B. J. Nelson. A deformable object tracking algorithm based on the boundary element method that is robust to occlusions and spurious edges. *IJCV*, 78(1):29–45, 2008. 1

[28] G. Guy and G. Medioni. Inferring global perceptual contours from local features. *IJCV*, 20(1-2):113–133, 1996. 4

[29] H. Helmholtz. *Treatise on physiological optics*. New York: Dover, 1962 (first published in 1867). 2, 7

[30] J. E. Hochberg. Effects of the Gestalt revolution: the Cornell symposium on perception. *Psychol. Rev.*, 64(2):73–84, 1957. 2

[31] H. D. III, J. Langford, and D. Marcu. Search-based structured prediction. *Machine Learning Journal*, 2009. 4, 5, 9

[32] L. Itti and C. Koch. Computational modeling of visual attention. *Nature Reviews Neuroscience*, 2(3):194–203, 2001. 2

[33] A. Jain, A. Gupta, and L. S. Davis. Learning what and how of contextual models for scene labeling. In *ECCV*, 2010. 2

[34] I. H. Jermyn and H. Ishikawa. Globally optimal regions and boundaries as minimum ratio weight cycles. *TPAMI*, 23(10):1075–1088, 2001. 4

[35] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *JAIR*, 4:237–285, 1996. 9

[36] G. Kim, C. Faloutsos, and M. Hebert. Unsupervised modeling of object categories using link analysis techniques. In *CVPR*, June 2008. 7

[37] J. Kittler, M. Hatef, R. P. W. Duin, and J. Matas. On combining classifiers. *TPAMI*, 20:226–239, 1998. 11

[38] K. Koffka. *Principles of Gestalt Psychology*. Routledge, London, UK, 1935. 1, 2, 3

[39] I. Kokkinos. Boundary detection using F-measure-, Filter- and Feature- (F^3) boost. In *ECCV*, 2010. 3, 17

[40] I. Kokkinos. Highly accurate boundary detection and grouping. In *CVPR*, 2010. 3, 14, 19

[41] S. Konishi, A. Yuille, J. Coughlan, and S.-C. Zhu. Fundamental bounds on edge detection: An information theoretic evaluation of different edge cues. In *CVPR*, 1999. 3

[42] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu. Statistical edge detection: learning and evaluating edge cues. *TPAMI*, 25:57–74, 2003. 3

[43] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, pages 282–289, 2001. 4

[44] Y. Lee and K. Grauman. Shape discovery from unlabeled image collections. In *CVPR*, 2009. 7

[45] T. Lindeberg. Edge detection and ridge detection with automatic scale selection. *IJCV*, 30(2):117–156, 1998. 3

[46] D. G. Lowe. *Perceptual Organization and Visual Recognition*. Kluwer Academic Publishers, Norwell, MA, USA, 1985. 2, 7

[47] S. Mahamud, L. R. Williams, K. K. Thornber, and K. Xu. Segmentation of multiple salient closed contours from real images. *TPAMI*, 25(4):433–444, 2003. 3, 4

[48] 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*, pages 43–56, 2008. 13

[49] M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. Using contours to detect and localize junctions in natural images. In *CVPR*, pages 1–8, 2008. 5, 6, 12, 13, 14, 16, 17, 18, 20

[50] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001. 2, 8, 13, 17

[51] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26:530–549, 2004. 1, 3, 11, 12, 14

[52] M. C. Morrone and R. A. Owens. Feature detection from local energy. *Pattern Recogn. Lett.*, 6(5):303–313, 1987. 3

[53] S. Palmer. *Vision Science: Photons to Phenomenology*. MIT Press, 1999. 1, 2, 3

[54] P. Perona and J. Malik. Detecting and localizing edges composed of steps, peaks and roofs. In *ICCV*, 1990. 3

[55] J. Porrill and S. Pollard. Curve matching and stereo calibration. *IVC*, 9(1):45–50, 1991. 1

[56] X. Ren. Multi-scale improves boundary detection in natural images. In *ECCV*, 2008. 3, 13

[57] X. Ren, C. Fowlkes, and J. Malik. Learning probabilistic models for contour completion in natural images. *IJCV*, 77(1-3):47–63, 2008. 1, 2, 3, 4, 6, 7

[58] Y. Rubner and C. Tomasi. Coalescing texture descriptors. In *ARPA Image Understanding Workshop*, pages 927–935, 1996. 3

[59] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *IJCV*, 77(1-3):157–173, 2008. 2, 13, 17

[60] E. Sharon, A. Br, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. In *CVPR*, pages 469–476, 2001. 3, 4

[61] A. Shashua and S. Ullman. Structural saliency: The detection of globally salient structures using a locally connected network. In *ICCV*, 1988. 4

[62] B. Taskar, C. Guestrin, and D. Koller. Max-margin Markov networks. In *NIPS*, 2004. 4

[63] C. H. Teh and R. T. Chin. On the detection of dominant points on digital curves. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(8):859–872, 1989. 7

[64] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *J. of Machine Learning Research*, 6:1453–1484, 2005. 4

[65] M. Varma and A. Zisserman. Texture classification: Are filter banks necessary? *CVPR*, 2:691, 2003. 12

[66] S. Wang, T. Kubota, J. M. Siskind, and J. Wang. Salient closed boundary extraction with ratio contour. *TPAMI*, 27(4):546–561, 2005. 3

[67] S. Will, L. Hermes, J. M. Buhmann, and J. Puzicha. On learning texture edge detectors. In *ICIP*, pages 877–880, 2000. 3

[68] L. Williams and D. Jacobs. Stochastic completion fields: A neural model of illusory contour shape and salience. In *ICCV*, pages 408–415, 1995. 3, 4

[69] L. R. Williams and K. K. Thornber. A comparison of measures for detecting natural shapes in cluttered backgrounds. *IJCV*, 34(2-3):81–96, 1999. 4

[70] W. Xiong and J. Jia. Stereo matching on objects with fractional boundary. In *CVPR*, 2007. 1

[71] S. Yu. Segmentation induced by scale invariance. In *CVPR*, 2005. 3

[72] Q. Zhu, G. Song, and J. Shi. Untangling cycles for contour grouping. In *ICCV*, pages 1–8, 2007. 3, 4, 13

[73] S.-C. Zhu. Embedding Gestalt laws in Markov random fields. *TPAMI*, 21(11):1170–1187, 1999. 2, 3, 4, 7