Computer Vision and Image Understanding 199 (2020) 103026

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

Low-level multiscale image segmentation and a benchmark for its evaluation

Emre Akbas [a,*], Narendra Ahuja [b]

[a] Department of Computer Engineering, Middle East Technical University, Ankara 06800, Turkey
[b] Department of Electrical and Computer Engineering, Coordinated Science Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA

ARTICLE INFO

Communicated by Nikos Paragios

MSC:
41A05
41A10
65D05
65D17

Keywords:
Image segmentation
Benchmark
Low-level vision

ABSTRACT

In this paper, we present a segmentation algorithm to detect low-level structure present in images. The algorithm is designed to partition a given image into regions, corresponding to image structures, regardless of their shapes, sizes, and levels of interior homogeneity. We model a region as a connected set of pixels that is surrounded by ramp edge discontinuities where the magnitude of these discontinuities is large compared to the variation inside the region. Each region is associated with a scale that depends upon the fraction of the strong and weak parts of its boundary. Traversing through the range of all possible scales, we obtain all regions present in the image. Regions strictly merge as the scale increases; hence a tree is formed where the root node corresponds to the whole image, nodes closer to the root along a path correspond to larger regions and those further from the root capture smaller regions representing embedded details. To evaluate the accuracy and precision of our algorithm, as well as to compare it to the existing algorithms, we present a new benchmark dataset for low-level image segmentation. We provide evaluation methods for both boundary-based and region-based performance of algorithms. We also annotate different parts of the images that are difficult to segment with the types of segmentation challenges they pose. This enables our benchmark to give an account of which algorithm fails where. We show that our proposed algorithm performs better than the widely used low-level segmentation algorithms on this benchmark.

1. Introduction

This paper is concerned with the problem of low-level image segmentation, that is, partitioning a given image into regions that represent low-level image structure. A low-level image segment is characterized by a set of connected pixels having a certain degree of intensity homogeneity in the interior and a relative discontinuity at the border with their surrounding, where the magnitude of discontinuity, or the contrast, is large compared to the interior variation. While this definition of segment does not restrict the criteria that the notion of homogeneity and contrast are based on, the term "low-level " restricts them to local and intrinsic properties of pixels such as brightness, color, gradient, optical flow, etc. This distinguishes a low level segment from those constituting higher, e.g. object level segmentation, where the homogeneity (e.g., inside an animal body segment) and contrast (discontinuity across the animal body segment) may be defined in terms of somewhat higher, object level properties.

The sizes, shapes, homogeneity and intensity contrast values of regions in an image are *a priori* unknown. The homogeneity and contrast parameters associated with the population of regions in an image form the set of photometric scales present in the image, while the region sizes are said to define the geometric scales present. The size of a region relevant to low level segmentation is a local measure; it refers to the extent of the region in the vicinity of the pixel, as captured by, e.g., the distance to the nearest point on the region border, and is called the region's local geometric scale (Ahuja, 1996). The set of scales associated with a region is obtained from all the pixels in the region. Clearly, the number and scale values possessed by a region of a given total size depend on the region's shape.

The goal of low-level image segmentation is to detect all valid image regions regardless of their sizes, shapes, level of homogeneities and contrast values, while identifying the scales associated with them. Thus, an image in general contains multiple low-level segmentations, associated with different natural scales occurring in the image. The set of all valid regions can be organized as a hierarchical structure where the hierarchy relates two regions by the containment relation – a given region may be embedded within a larger region of higher contrast. This hierarchical structure is a tree, called the segmentation tree (Ahuja, 1996), and it captures all the segmentations of the image and thus represents the multiscale, low level structure of the image. To obtain the segmentation tree of an arbitrary gray-scale (or color) image is the objective of image segmentation pursued in this paper.

Evaluating and characterizing the performance of low level segmentation algorithms is the second major focus of this paper. While there have been efforts to develop evaluation methods for high-level

* Corresponding author.
E-mail address: emre@ceng.metu.edu.tr (E. Akbas).

https://doi.org/10.1016/j.cviu.2020.103026
Received 19 June 2020; Accepted 21 June 2020
Available online 27 June 2020

segmentation (Martin et al., 2001; Everingham et al., 2010), the same cannot be said about low-level image segmentation. To this end, we develop a benchmark dataset for evaluating the quality of low-level image segmentation algorithms by comparing their results with the known ground truth. The benchmark dataset consists of a large number of image patches that are small, and therefore, their contents are not always recognizable. Their ground-truth segmentations are identified by human subjects. These segmentations are low-level because, in the absence of any cues about their high level content, humans may only segment them by the spatial variations in the pixel values. We use this dataset to characterize the performance of a given segmentation algorithm in terms of three types of differences between the algorithm results and ground truth regions, namely differences between: region boundaries, region interiors, and both together.

The relationship between our objective of low-level segmentation and semantic segmentation can be viewed as follows. The separation of regions of different contrasts achieved by low-level segmentation is useful since regions of interest in an image often correspond to regions on physical objects, having similarities within, and differences across, in physical properties. For example, these differences may be in the materials or lighting present in these regions. Since, boundaries between physical objects also are in general characterized by such physical differences, detection of low-level regions is useful as primitives whose connected sets may constitute semantic segments. The use of our low-level segmentation as inputs, instead of raw images, can thus potentially improve the complexity and performance of semantic segmentation algorithms.

1.1. Related work

There is a large body of past work on segmentation. The earliest techniques were based on thresholding where a histogram of color values of all pixels is computed and then peaks and valleys of this histogram are located. A given image is segmented by thresholding it using the values where valleys occur in the histogram. These kind of methods are extremely efficient and work well for simple figure-background images. In practice, the valleys of the histogram cannot be located easily and reliably. A survey of these methods is given by Sahoo et al. (1988). Thresholding is still being used in segmentation related tasks. For example, (Matas et al., 2002) use thresholding and look for stable regions which do not change much as the threshold is changed.

A significant body of work is based on region-growing. In this technique, a region is expanded outward from a seed pixel, recursively examining the pixels across the region boundary and absorbing them in to the region if they are sufficiently similar to the region. Designing this similarity test is the most important part of the technique because it determines when the region stops to grow, i.e. where the boundary of the segmented region lies. Related to region growing are split-and-merge techniques where contiguous regions are recursively merged, or a region is recursively split, until the resulting region is as large as possible and its pixel values satisfy some predefined similarity rule. The reader is referred to Stockman and Shapiro (2001) for a review of these techniques.

Another popular technique for segmentation is clustering (Carson et al., 2002; Comaniciu et al., 2001; Comaniciu and Meer, 2002; Ren and Shakhnarovich, 2013; Yu et al., 2015; Kim et al., 2014). In this approach, each pixel or superpixel (small group of pixels) (Achanta et al., 2012) is associated with a feature vector containing some attributes of the pixel (such as intensity or color) and/or the pixel's neighborhood (such as texture). To account for the location of the pixel (or superpixel) in the image plane, either the x, y positions of the pixel are added to the feature vector or location information is implicitly represented using pixel neighborhoods (for example, as in Kim et al., 2014). Then, a clustering (e.g. Comaniciu and Meer, 2002; Ren and Shakhnarovich, 2013) or embedding (e.g. Yu et al., 2015; Shi and Malik, 2000) method is used to find the clusters in the spatio-attribute feature space. Clusters detected correspond to image regions. Although any clustering method can be used for this purpose, agglomerative methods, k-means, expectation–maximization and mean-shift clustering are widely used. The main shortcoming of these approaches is that one does not *a priori* know the relative importance of the parameters or how to select their values for the clustering algorithm, e.g. the number of regions, kernel bandwidth for mean-shift, number of agglomerative stages to run (e.g., in Ren and Shakhnarovich, 2013), etc. A good review of segmentation using clustering is given by Forsyth and Ponce (2002).

Graph based methods have also been adopted for image segmentation (Zabih and Kolmogorov, 2004; Shi and Malik, 2000; Felzenszwalb and Huttenlocher, 2004; Yu et al., 2015; Kim et al., 2014). In these approaches, initially a graph is constructed where each node represents a pixel or a group of pixels, and the edges between nodes encode a measure of inter-node affinity. Affinity can be expressed in terms of spatial proximity, intensity/color and texture similarity (possible when a node represents a group of pixels) between nodes. In the work of Zabih and Kolmogorov (2004) and Shi and Malik (2000), the goal is to cut the graph into a predefined number of connected components in such a way that the sum of weights of the edges that are cut is minimal. In practice, finding these minimal cuts boils down to computing the eigenvectors of a matrix called "the affinity matrix" which contains the weights of edges in the graph. The most popular segmentation algorithm in this category is the normalized-cuts (Shi and Malik, 2000) (known as N-cuts). Recently, normalized-cuts was improved by Yu et al. (2015) by solving a L_1 regularized version of the objective function in order to produce more definite transitions between regions. These approaches also have the shortcoming of clustering based segmentation methods wherein the user has to specify the number of regions in advance. Another graph based method worth noting is Felzenswalb's algorithm (Felzenszwalb and Huttenlocher, 2004) where segmentation is formulated as finding multiple disjunct minimum-spanning-trees (MST) that cover the entire image. Each tree corresponds to a distinct region, and existence of an inter-region boundary is decided based on a heuristic predicate that compares the contrast across the boundary (minimum weight edge between two MSTs) and the interior contrast (maximum weight edge) within each region (MST). This formulation has similarities with our region model in that it compares the contrast across and within the region but does so based only on the maximum contrast edge in each case, instead of estimating it from the many edges that occur across and within the regions. Further, the algorithm computes a single-level partitioning of the image; that is, it ignores multiscale aspect of segmentation. Finally, it implicitly uses the step-edge model (hence ignores the ramp nature of the real world edges) which causes the algorithm to produce many thin regions within the ramp areas between regions.

In the context of our study, we can classify the previous work as either not being multiscale, or imposing models on the geometry of edges. The earliest approaches to segmentation such as thresholding (Haralick and Shapiro, 1985), region-growing (Haralick and Shapiro, 1985) and watersheds (Meyer and Beucher, 1990; Bleau and Leon, 2000) ignore the multiscale aspect of the problem. Energy minimization based approaches such as Markov random field (MRF) modeling (Geman and Geman, 1984) and active contours (Xu and Prince, 1998) enforce constraints on the local shape of regions; therefore, they are not capable of detecting arbitrarily shaped regions. Graph based methods such as normalized cuts (Shi and Malik, 2000) and graph cuts (Zabih and Kolmogorov, 2004) require the number of regions to be given as input, which yields a subset of all the regions present at all scales. Clustering methods attempt to find regions as clusters in the joint space of pixel positions and some features, (e.g. intensity, texture, etc.) extracted from pixels. For this, they either need the number of regions or some parameter specific to the method (such as density estimation parameters, number of stages in agglomerative clustering,

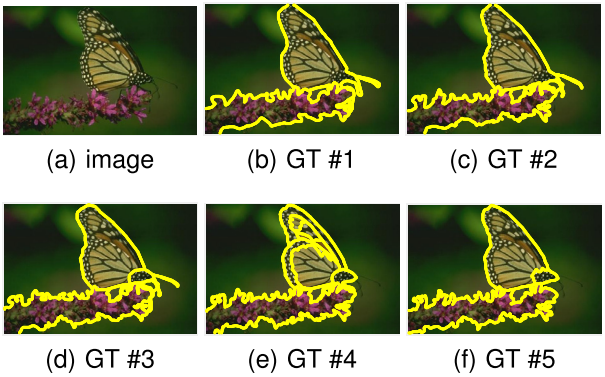

Fig. 1. An example illustrating high-level (i.e. object-level or semantic) segmentation from the BSDS500 dataset (Arbelaez et al., 2011). Five different ground-truth (GT) segmentations (done manually by humans) are shown (b–f) for the image in (a). Note that many regions – especially within the butterfly's wing and on the flower – are missing in the GT segmentations.

etc.) (Comaniciu and Meer, 2002; Comaniciu et al., 2001; Kim et al., 2014; Ren and Shakhnarovich, 2013) as input. As discussed by Arora and Ahuja (2006), mean-shift based segmentation cannot detect steep corners due to averaging and tends to create multiple boundaries, hence many small regions, for the blurred edges in the image.

In recent years, many segmentation algorithms (including Arbelaez et al., 2014; Yu et al., 2015; Ren and Shakhnarovich, 2013; Kim et al., 2014; Arbelaez et al., 2011; Donoser and Schmalstieg, 2014 as notable examples) have been developed aimed at maximizing performance on the Berkeley Segmentation Benchmark Dataset (BSDS) (Martin et al., 2001; Arbelaez et al., 2011) which contains images segmented by humans. We note that these are object-level, i.e. semantic or high-level, segmentations and many intensity-wise pronounced but semantically weak regions in the images are not marked. Also, some regions are marked even if there is too little or no visual appearance (low-level) evidence. An example image from this dataset and its ground truth segmentations are given in Fig. 1. As one can observe, in 4 out of the 5 ground truth segmentations, the boundaries mark only the semantic objects, namely the butterfly and the flower. Only in "GT #4", details within the butterfly's wing are partly marked. From the viewpoint of low-level segmentation, these segmentations are unsuccessful due to the fact that many regions in the wing of the butterfly and on the flower are missing. Note that the butterfly example given in Fig. 1 is not an isolated but a typical one. One can find many such images (see also Fig. 12) in the BSDS dataset. It is not our goal, in this study, to segment out (semantic) objects; instead, we aim to detect low-level image structures, i.e. regions, as accurately and completely as possible so as to provide a reliable and usable input to other vision algorithms (as we have demonstrated in Cheng and Ahuja, 2012; Moaveni et al., 2013; Akbas and Ahuja, 2010, 2014; Singh and Ahuja, 2013; Ghanem and Ahuja, 2013; Nam and Ahuja, 2012; Singh and Ahuja, 2012, 2015). The low-level segments (homogeneous regions) we extract are intended to serve as primitives; an object/semantic segment is often a union of such low level segments. The value of detecting the low level segments is somewhat like that of detecting strokes of characters and characters in text, which may then be composed to form high-level/semantic segments, corresponding to words and beyond. It is a different matter as to what type of control flow might be involved between the detections of low level and high level segments, but the detection of low level segments is by itself a longstanding problem in image analysis and computer vision. Segmentation algorithms that try to maximize performance on the BSDS dataset are mostly learning based algorithms, trained on high-level boundaries as in Fig. 1, and therefore tuned to high-level semantic segmentation. Nevertheless, we compare the performance of our algorithm with two such state-of-the-art algorithms (namely, gPb Arbelaez et al., 2011 and Multiscale Combinatorial Grouping (MCG) (Arbelaez et al., 2014)) from this category.

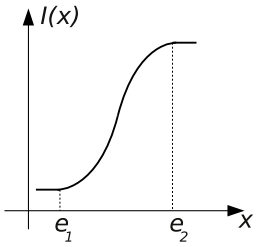

Fig. 2. Ramp model stipulates that any 1-D cross section of the boundary between two adjacent regions across (illustrated as a curve, or a 1-D image here) is a ramp characterized by an increasing or decreasing intensity profile.

1.2. Overview

We develop a segmentation algorithm which does not assume any prior models of the shape, size, contrast, number and boundary geometry of regions. We follow the basic formulation of Ahuja (1996) and Tabb and Ahuja (1997) but instead of viewing a region as being surrounded by a step edge, we model an image region as a set of connected pixels surrounded by ramp discontinuities, as done by Arora and Ahuja (2006), with strictly increasing (or decreasing) intensity profiles. The ramp discontinuity across a boundary point has an associated ramp height, or contrast, which allows us to associate a photometric scale the boundary point. From the contrast values occurring all along the boundary, we derive a single contrast value to associate with the entire boundary, or region, and call it the photometric scale of the region. We then analyze the discontinuities obtained for the entire range of possible contrast values, given by the complete range of intensity values, and obtain segmentations over all naturally occurring photometric scales in the image. Finally, all regions detected at all photometric scales are organized into a tree data structure, where the hierarchy is defined by recursive containment of regions.

To evaluate the performance of our algorithm, as well as to compare it to the existing algorithms, we develop a new benchmark dataset for low-level image segmentation. Our method for evaluating the performance of segmentation algorithms compares the ground truth with the algorithm-provided segmentation in terms of agreement between the boundary and interior properties as well as their combination. We also classify different types of segmentation challenges and annotate the benchmark dataset with their occurrences in individual test images. This enables our benchmark dataset to give an account of which algorithm fails where. Using this benchmark dataset, we show that our proposed algorithm performs better than other widely used low-level segmentation algorithms.

1.2.1. Contributions

In this paper, we make the following new contributions over our previous work (Akbas and Ahuja, 2009):

1. We bring a theoretical justification for the greedy agglomerative merging of regions by introducing the problem of "photometrically stable regions" (Section 2.3). We believe that this theoretical contribution will be useful for any work that uses greedy agglomerative merging.
2. We simplify the segmentation algorithm by replacing the costly relaxation labeling step by a basic watershed procedure.
3. We extend the benchmark dataset in several different ways:
    - Ground-truth segmentation was obtained from multiple human subjects instead of just one.
    - Ground-truth segmentations were collected on a horizontally laid tablet-pc by drawing on screen using a stylus pen, which increased the quality of segmentation.
    - We added two new performance measures, region-based and combined-boundary-and-region-based, in addition to the existing boundary-based measure.

- And, most importantly, we enriched the benchmark dataset by introducing a taxonomy of segmentation challenges, and locating them in images, which reveals specific weaknesses and strengths of any given segmentation algorithm in terms of performance on the various challenges.

The rest of the paper is organized as follows. In Section 2, we describe our segmentation algorithm, including the ramp discontinuity and region models, using them to detect regions at all photometric scales, and their organization into a segmentation tree. Experimental results are presented in Section 3 in two parts. First, we give sample segmentation tree results for qualitative inspection. Next, we present the benchmark dataset and compare the performance of our algorithm with those of existing, widely used segmentation algorithms, in terms of various segmentation challenges posed by the dataset. Finally, Section 4 presents conclusions.

2. The models and the algorithm

A set of connected pixels, R, is said to form a *region* if it is surrounded by ramp discontinuities, and the magnitudes of these discontinuities (contrasts) are larger than the contrasts (intensity variations) within R. To elaborate on this definition, we first describe the ramp discontinuity model.

2.1. Ramp model

We assume that the discontinuities that separate adjacent regions in an image are characterized by increasing or decreasing profiles of intensities along any path (e.g. a line segment) starting at one end of the ramp and monotonically approaching the other. Consider the one-dimensional image I given in Fig. 2. The part of the image between e_1 and e_2 is a ramp. The width of the ramp is $|e_1 - e_2|$, and its magnitude, i.e. contrast, is $|I(e_2) - I(e_1)|$. The ramp endpoints can be detected by inspecting the sign of the derivative of the image. Within a ramp, the sign of the derivative does not change and its magnitude is non-zero; at a ramp endpoint, the derivative becomes zero, or may even change sign when overshoots and undershoots are present at region boundaries. To identify ramps, we consider all such paths, if any, associated with each pixel $\boldsymbol{p}$, and their profiles. In the next section, we describe how these ramp profiles are processed in order to produce a contrast map of the image.

2.2. Ramp transform

The transform converts the input image I to a scalar height field C. The height at pixel $\boldsymbol{p}$, $C(\boldsymbol{p})$, is computed as follows. Suppose I is the 1-D image given in Fig. 2. Then, for a pixel p between e_1 and e_2,

$$C(p) = \begin{cases} |I(e_2) - I(e_1)| & \text{, if } I''(p) = 0 \\ 0 & \text{, otherwise.} \end{cases} \tag{1}$$

That is, the contrast of the ramp is assigned to the pixel(s) where the rate of change of the first derivative of the image, I', is maximum (zero-crossing of $I''(\cdot)$, i.e. $I''(\cdot) = 0$).

When I is a 2-D image, an infinite number of lines pass through pixel $\boldsymbol{p}$, each having its own intensity profile, passing through multiple ramp discontinuities of the regions lying along. Each discontinuity has its own local maximum. Let θ, the angle that a line makes with the horizontal axis of the image, parametrize these lines, and hence their corresponding intensity profiles. For a pixel $\boldsymbol{p}$ and an angle θ, we take the intensity profile passing through $\boldsymbol{p}$ at θ as a 1-D image and compute the associated directional contrast, $C(\boldsymbol{p}, \theta)$, as described above. This is done for a finite set of angles in $[0, 2\pi)$. Then, the final result is obtained by taking the maximal contrast over angles at each pixel,

$$C(\boldsymbol{p}) = \max_{\theta} C(\boldsymbol{p}, \theta). \tag{2}$$

2.3. Region model and detecting "photometrically stable regions"

As stated earlier, we model a region, R, as a set of connected pixels, having ramp discontinuities along its border, such that the contrasts (2D local variations) within the ramp are larger than those at pixels within R. The region shape is assumed to be a priori unknown, and is determined by the geometry of the surrounding ramp discontinuities. We detect region boundary without making any restrictive assumptions about its geometry.

To obtain regions that conform with the above model, note that the output, C, of the ramp transform is a scalar height field wherein height is proportional to contrasts of the ramp discontinuities. Neighboring local maxima pixels lie along the axis of a ramp discontinuity and serve as the edges of the associated region. To locate these maxima, we use the watershed approach on the topography defined by C (Vincent and Soille, 1991). We start filling it with water, which starts rising starting at the minima, and filling the landscape until neighboring pools start merging at their common separating local maxima (Vincent and Soille, 1991), starting with low lying pools progressing towards the high. Each initially disconnected pool of water forms a unique region, with maxima of C serving as watershed ridges surrounding basins of lower contrasts. A region's contrast is determined by when it merges with a neighboring pool. From the partitioning of I, we produce a boundary map B where adjacent regions are separated by boundary pixels (i.e. pixels on watershed ridges). The ridge edges constitute region boundary map with the contrast at each boundary pixel known from C.

Boundary contrasts and homogeneity levels of regions present in an image are a priori unknown. A region might have faint boundaries, hence a relatively quite homogeneous interior due to the fact that the contrasts within a valid region cannot be larger than its boundary contrasts. Or, it could have high contrast boundaries and a higher level of inhomogeneity inside, still lower than the boundary contrasts. To account for this appearance variability, we associate each region with a photometric scale that describes its homogeneity and contrast. Accordingly, segmentation at a given photometric scale σ is defined as the set of all valid regions (i.e. regions with closed boundaries, without any leakages) having a photometric scale greater than σ. Photometric scale helps separate the different physically meaningful regions, as discussed in the paragraph just preceding Section 1.1

In previous work, the photometric scale of a region was defined to be the minimal contrast along its boundary (Ahuja, 1996; Tabb and Ahuja, 1997; Arora and Ahuja, 2006). Based on this definition, detecting all regions at a given photometric scale σ is a trivial task: simply threshold C at σ. However, this (old) definition causes premature leakage. To illustrate, suppose a high contrast region boundary has a few low contrast pixels. Because the region's photometric scale is determined by the lowest contrast along its boundary, this is not a valid region at high photometric scales, even though most of its boundary has high contrasts. In this paper, we relax and generalize the definition of the photometric scale of a region by not ruling out low contrast boundary pixels to be part of region boundaries at higher photometric scales. The old definition then becomes a special case of this new definition.

Given a photometric scale σ, we categorize the boundary pixels (in B) into two: *strong* and *weak*, depending on their contrasts being larger than σ or not, respectively. To obtain the segmentation at scale σ, we want to

- minimize the number of *weak* pixels that are part of region boundaries, and
- maximize the number of *strong* pixels that are part of regions boundaries, or, equivalently, minimize the number of *strong* pixels that are dangling (i.e. not forming a closed contour).

Formally, S_σ, the segmentation at scale σ, is the solution of the following optimization problem:

$$S_\sigma = \arg\min_S \sum_{p\in B} \mathbf{1}_{\{C(p)<\sigma, p\in S\}} + \sum_{p\in B} \mathbf{1}_{\{C(p)>\sigma, p\notin S\}}, \quad (3)$$

where $C(p)$ denotes the contrast of the boundary pixel p, S is a valid segmentation, "$p \in S$" means that pixel p is part of the boundary of a region in S, and $\mathbf{1}_{\{\cdot\}}$ is the indicator function. The first term in Eq. (3) counts the number of weak pixels that are part of a region boundary in S_σ, and the second term counts the number of strong pixels that are **not** part of a region boundary in S_σ.

Without loss of generality, this minimization problem can be formulated in terms of *boundary fragments* in B (each associated with the closed contour of a separate region) instead of pixels in B. Doing so would make processing more efficient because the entire fragment must either be accepted or rejected, as partial acceptance will lead to a dangling segment which is not desirable.

Let $G = (V, E)$ be the region adjacency graph corresponding to the boundary map B. Each vertex in V corresponds to a region, and each edge in E corresponds to a boundary-fragment separating two adjacent regions. We assign weights to the edges in E in terms of the given photometric scale σ as follows:

$$w(e) = \sum_{p\in e} \mathbf{1}_{\{C(p)\geq\sigma\}} - \mathbf{1}_{\{C(p)<\sigma\}} \quad \forall e \in E, \quad (4)$$

where $w(e)$ is the weight of the edge e. The first term counts the number of *strong* boundary pixels in e, and the second term counts the *weak* ones. The larger is $w(e)$, the stronger is the fragment e, hence, more desirable it is to be part of a region boundary in the segmentation result S_σ. As the optimization objective given in Eq. (3) dictates, we want a minimal number of weak pixels and a maximal number of strong pixels to be part of the final solution, S_σ. We formally state this problem, which we call as the "photometrically stable regions", in terms of boundary fragments as follows.

Problem "PSRσ": PHOTOMETRICALLY STABLE REGIONS AT σ

Given σ and graph $G = (V, E)$ with edges weighted according to Eq. (4), color the vertices in such a way that the sum of the weights of the monochromatic edges (i.e. those between same color vertices) is minimized.

In the final graph G', each unique color represents a region. The solution S_σ is given by the coloring in G'.

This problem is NP-hard (see our proof in the Appendix). For its solution, we consent to a greedy heuristic algorithm which we sketch in the following. First, we construct the region adjacency graph $G = (V, E)$ from the boundary map B. We assign a unique color to each vertex in V and compute the weights of the edges in E using Eq. (4). Thus, initially, each edge is dichromatic, i.e., by design, it separates two different regions. Now we identify the dichromatic edge with the lowest weight. Let this edge be e and the vertices that e connects to each other be v_1 and v_2 (i.e. e is incident to v_1 and v_2). Making v_1 and v_2 the same color means that e becomes a monochromatic edge. This action is equivalent to merging the two regions represented by v_1 and v_2. Since the goal of the optimization is to minimize the sum of the weights of the monochromatic edges, we make e a monochromatic edge only if $w(e)$ is negative. Note that $w(e) < 0$ means that e is a *weak* edge at the given photometric scale σ. If $w(e) > 0$, then we do not make it monochromatic, since doing so would increase the objective value (sum of the weights of the monochromatic edges). This re-coloring rule can be applied repeatedly until there are no dichromatic edges with negative weights left. Note that as the algorithm progresses, making two vertices the same color may result in more than one edge in the graph to become monochromatic.

In practice, instead of keeping track of which edges are monochromatic and which ones are dichromatic, when $w(e)$ is negative, we apply the following steps to make sure all edges in G are dichromatic (i.e. separating two different regions) at all times: (1) we remove e from G, (2) we create a new vertex v_3 to represent the new region R which is the union of the regions corresponding to v_1 and v_2, (3) we create a new edge between v_3 and the vertex of each region that is spatially adjacent to R (if any), (4) recompute the weights for the new edges created in the previous step, (5) remove v_1 and v_2 from G. These steps lead to the algorithm outlined in Algorithm 1.

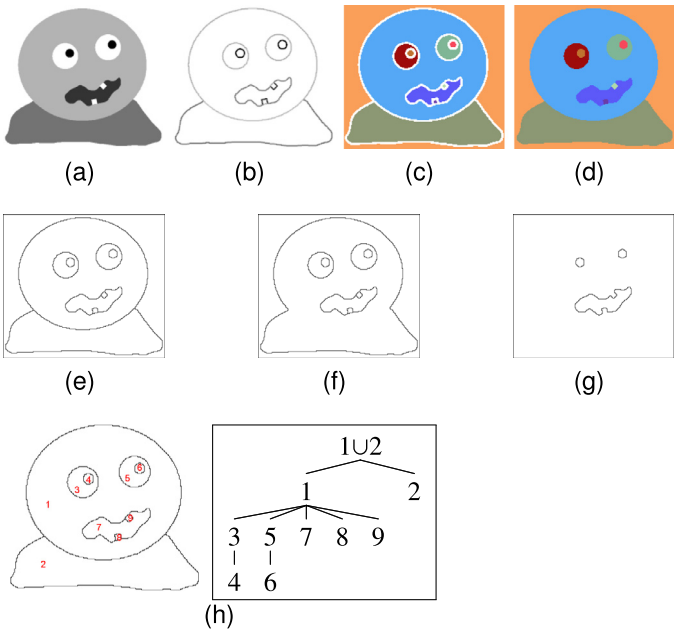

Fig. 3. Illustration of steps in the algorithm. (a) Input image I. (b) Output of ramp transform, C. Here, the darker the pixel, the higher the contrast of the underlying ramp discontinuity. (c) Basins of C. Each basin, associated with a unique local minimum of C, is represented with a different color. (d) Final labeling obtained by watershed. (e, f, g) Results of multiscale segmentation: (e) Segmentation result at photometric scale $\sigma = 5$. (f) Segmentation at $\sigma = 65$. Two regions (head and the body) merged, which means the photometric scales of these regions are less than 65. (g) Segmentation for $\sigma = 80$. More regions have merged. The remaining regions are of photometric scale larger than $\sigma = 80$. (h) Segmentation tree. On the left, each region is labeled by a number. Using the containment relations of regions, our algorithm computes the tree given on the right-hand side.

Algorithm 1 PSRσ: Computing photometrically stable regions at photometric scale σ.

Input: Photometric scale σ, boundary map B, ramp discontinuity contrasts C.
Output: S_σ, segmentation at σ.

1: Construct region adjacency graph $G = (V, E)$ from B.
2: Compute edge weights using Eq. (4).
3: **while** there exists an edge in E with negative weight **do**
4: Identify the edge, $e \in E$, with smallest weight $w(e)$.
5: Let v_1 and v_2 be the vertices that e connects to each other. Let R_1, R_2 represent the regions corresponding to vertices v_1, v_2.
6: Remove e from G.
7: Add a new vertex v_3 to G, representing the newly formed region $R_3 = R_1 \cup R_2$.
8: Create new edges in G between v_3 and the vertices representing the spatially adjacent neighbors of R_3 (if any).
9: Compute the weights of the edges created in the previous step (if any) using Eq. (4).
10: **end while**
11: S_σ is the segmentation defined by the final graph G.

The procedure given in Algorithm 1 guarantees a local minimum solution for PSRσ. To see this, consider the output, S_σ, of the algorithm.

Fig. 4. Sample segmentation result for qualitative inspection. From left to right: input image, segmentations at scales $\sigma = 25$, 45 and 70. Region boundaries are drawn with red color in between pixels (best viewed in color and when zoomed-in). Results show regions where the boundary is not simple. In particular, steep corners (at the left corner of the roof in the lower-right part of the image), thin regions (on the roof ends) and jagged boundaries (in the middle-right where the roof meets the wall) can be observed. A partial segmentation tree for this image is given in Fig. 11.

Fig. 5. Sample segmentation result for qualitative inspection, analogous to Fig. 4. From left to right: input image, segmentations at scales $\sigma = 45$ and 80. Many thin regions (e.g. the spokes on the wheels), steep corners (e.g. at the front fork) can be observed. This example illustrates well the intuition behind σ (as a contrast threshold). For example, at $\sigma = 45$ we see both the wheels and the spokes; however, at $\sigma = 80$, the spokes have disappeared, implying that the contrast of the spoke-regions is less than 80.

There are two possible actions on S_σ; one can further remove an edge, or one of the already removed edges (in line 6) can be brought back. Taking the first action would remove a boundary fragment which consists mostly of strong pixels since in the final graph G, it is guaranteed that $w(e) > 0$ for all e; and this action consequently would increase objective value due to the second term in Eq. (3). On the other hand, taking the second action would bring in a weak boundary fragment which will increase the first term in (3). Therefore, S_σ is a local minimum of PSRσ.

2.4. Multiscale segmentation

To obtain multiscale segmentation, we run the Algorithm 1 for a range of photometric scales starting from σ_{min} to σ_{max} at $\Delta\sigma$ increments (lines 4–6 in Algorithm 2). This process ensures that remaining regions always conform with the region model (that a region must have a close contour) and successive merges continue to form a strict hierarchy.

Algorithm 2 The complete segmentation algorithm

Input: Image I.
Output: Segmentation tree, T, of I.
1: $C \leftarrow$ RampTransform(I)
2: Find local minima of C. These are seeds for regions.
3: Grow seeds using watershed which gives a labeling of all pixels, or equivalently, a boundary map B.
4: **for** $\sigma = \sigma_{min}$ to σ_{max} with $\Delta\sigma$ increments **do**
5: $S_\sigma \leftarrow$ PSR$\sigma(\sigma, B, C)$ {call Algorithm 1}
6: **end for**
7: Collect and make a list, L, of all regions from all S_σ.
8: Construct the segmentation tree T from the regions in L based on geometric containment relations.

2.5. Constructing the segmentation tree

As a part of the multiscale segmentation process described in the previous section, regions merge as the scale of analysis, σ, is increased. This allows us to arrange the regions into a tree data structure according to their geometric containment relationships. Suppose that regions R_1 and R_2 at photometric scale σ_n have merged and become R_3 at scale σ_{n+1}. Then, in the segmentation tree R_1 and R_2 are made the children of R_3. Applying this rule recursively for all regions, we obtain a tree of regions called the segmentation tree, where the root node corresponds to the entire image itself. We present the overall segmentation algorithm, where input is an image I, and the output is the segmentation tree of I, in Algorithm 2. We illustrate all steps of the algorithm on a simple synthetic image, in Fig. 3.

3. Experiments

This section describes how we evaluate the performance of our algorithm, both qualitatively and quantitatively. We first give sample segmentation and segmentation tree results for qualitative inspection. Next, we describe the benchmark dataset we developed to evaluate the performance of any low-level segmentation algorithm, ours included. We use the benchmark to quantitatively compare the performance of our algorithm with those of existing, popular approaches. Finally, we evaluate the performances of the various algorithms on specific types of image features such as blurred edges, corners, junctions and thin regions. We believe that the fidelity of the segmentation delivered by an algorithm at such image features is a good test of its performance.

3.1. Sample segmentations and segmentation trees

In Figs. 4–8, we present sample results of our segmentation algorithm for qualitative evaluation. In these figures, region boundaries

Fig. 6. Sample segmentation result for qualitative inspection, analogous to Fig. 4. From left to right: input image, segmentations at scales $\sigma = 25$, 40 and 60. This example, in addition to non-trivial segmentations such as thin regions (e.g. the microphones cable on the man's shirt) and low-contrast boundaries (e.g. the man's arms against the background), illustrates the fact that high-level/semantic objects (e.g. the upper torso of the man) are mostly composed (union) of our low level segments. A partial segmentation tree for this image is given in Fig. 10.

Fig. 7. Sample segmentation result for qualitative inspection, analogous to Fig. 4. From left to right: input image, segmentations at scales $\sigma = 45$ and 80.

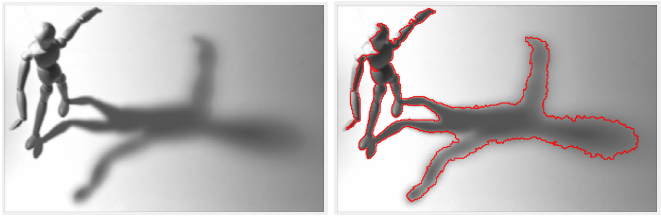

Fig. 8. Sample segmentation result for qualitative inspection, analogous to Fig. 4. From left to right: input image, segmentation at scale $\sigma = 140$. The input mannequin image is from (Elder and Zucker, 1998) (used with permission). This example, in particular, shows sharp and blurred (diffuse) edges (*cf.* the edges on the mannequin itself and its shadow) in the same image. Also, the boundary of the shadow of the mannequin's head has very low contrast. Our algorithm handles this challenging segmentation example thanks to our ramp transform which computes contrast by taking ramp width into consideration (further illustrated in Fig. 9).

Fig. 9. Gradient magnitude (left) and ramp transform output (right) of the mannequin image. The maximum contrast in each output is normalized to 1 for comparison. Images have the same colormap range. While the gradient magnitude quickly vanishes as the edges get more diffuse, ramp transform correctly computes contrasts regardless of the ramp width.

are drawn with red color in-between the pixels. To be specific, if two horizontally-neighbor pixels belong to different regions, we draw a red, vertical line in between them. For vertically neighboring pixels, we draw a horizontal line in between them. We used vector graphics to draw these lines, so that the viewer can zoom in to see both the detailed boundaries and the image itself. These figures are best viewed in color.

The results in Figs. 4–8 are for image parts where the boundary is not simple. In particular, high curvature boundaries, e.g. steep corners, can be found in the house image (Fig. 4, at the left corner of the roof in the lower-right part of the image) and in the bicycle image (Fig. 5, at the front fork of the bicycle.) Thin regions can be found in the bicycle image at the spokes and other parts, in the house image in the roof area, in the building image (Fig. 7) at the window frames, and in the "man with his bicycle image " (Fig. 6) at the microphone on man's shirt, at the pole in the background, etc. Jagged boundary structure can be found in the house image (Fig. 4) in the middle-right of the image in the roof area. A good example for blurred edges, i.e. wide ramps, is in the mannequin image (Fig. 8) where are very wide, diffuse edges (in the shadow area) with varying edge width along with very sharp edges (on the mannequin itself). The results of our algorithm can be seen to qualitatively correctly follow the perceptual edges in all these challenging parts of the images.

The mannequin image is also a good example for demonstrating the ramp transform. In Fig. 9, we present the output of the ramp transform together with the gradient magnitude of the image for comparison. The boundary contrasts of the shadow, i.e. the difference between the interior and the background intensities across the shadow boundary, are better estimated by the ramp transform. Due to the changing lighting, the width of the ramp discontinuities increases from the left to the right side of the image. Despite this variability in ramp width, ramp transform estimates the ramp contrasts correctly whereas the gradient magnitude fails to do so because it employs fixed-size finite difference filters thus ignoring the changing ramp width.

We finally present two automatically generated sample segmentation trees in Figs. 10 and 11. As we noted earlier in Section 2.5, the hierarchy is completely determined by containment relations of regions and not by their photometric scales. In a segmentation tree, the root node corresponds to the whole image, nodes closer to the root along a path represent larger regions, while child nodes represent smaller, embedded details of the parent node's region.

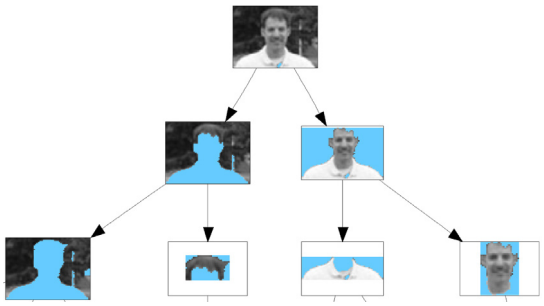

Fig. 10. Part of an automatically generated segmentation tree. The root node corresponds to the whole image, and nodes close to the root along a path are large, while their children nodes are smaller and capture embedded details. Each region is drawn on a light-blue background for better visualization of its boundaries. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

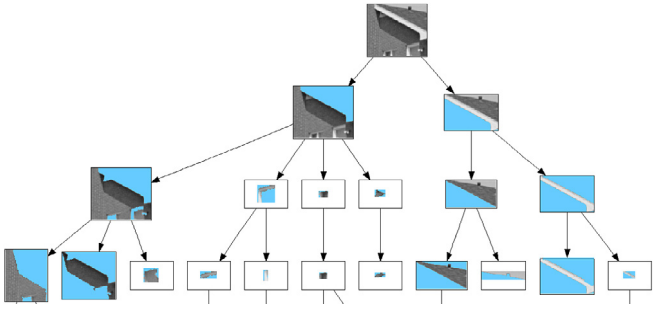

Fig. 11. Part of an automatically generated segmentation tree, analogous to Fig. 10.

Fig. 12. An example illustrating the disconnect between low-level segmentation and the ground-truth segmentation given in semantic segmentation datasets. The image on the left is from the Segmentation Benchmark Dataset (Martin et al., 2001; Arbelaez et al., 2011). On the right is its ground-truth segmentation. Many regions in the area marked with the red bounding box (on the left) are not marked in the ground-truth. See also Fig. 1 for another example.

3.2. Benchmark dataset

Creating a ground truth dataset for low-level image segmentation is a challenging task because (1) segmenting an image by hand is a laborious and tedious process, which adversely affects the quality of the result, and (2) more importantly, humans tend to draw the boundaries of only the (semantic) objects they see, and skip many other boundaries even if they are visually strong (but perhaps do not play an important role with respect to representing the object/semantics.) Indeed, in Berkeley Segmentation Benchmark Dataset (BSDS) (Martin et al., 2001; Arbelaez et al., 2011), human subjects segment out objects, not regions, and that is why many details in terms of low-level segmentation are missing. One example, among many, is given in Fig. 12. If the ground-truth given in Fig. 12 is used to test the results of a low-level segmentation algorithm, then the algorithm would be penalized for detecting the details missing in the ground truth. This high-level semantic bias towards objects must be eliminated in order to obtain a ground-truth for low-level segmentation. We do not consider BSDS as a suitable benchmark for low-level image segmentation. BSDS is good for benchmarking high-level or object-level segmentation. Our goal in this study is not object-level segmentation; instead, we want to detect all regions as accurately and completely as possible. As stated in the last paragraph of Section 1.1, through the example of text understanding, the motivation here is to extract general purpose image syntax, which could serve as an input to algorithms extracting semantics.

Fig. 13. The set of images used to create the benchmark dataset.

We address these challenges by having human subjects segment small image patches instead of whole images. Obviously, hand segmenting a small image patch is much less work than segmenting a whole image. Further, high-level knowledge bias is reduced to the extent the small image patch lacks evidence of large objects. To further reduce the bias, we also randomly rotate the image patch (at multiples of 90 degrees) which causes reduction/loss of object perception.

Fig. 13 shows a mosaic of the 15 images from which we collected patches to create the benchmark dataset. 15 images may appear too few to create a benchmark dataset, and, this would certainly be true if we were targeting high-level semantic segmentation. However, our need is an adequately large number of challenges for low-level segmentation, i.e., image patches containing different challenges, which maybe contained in much smaller number of images, particularly if the images are suitably chosen to be rich in such challenges (see Section 3.2.7). Therefore, we believe that for low-level segmentation, diversity of low-level content of images rather than their number is more important. Our set contains images of indoor and outdoor scenes, containing natural and man-made objects at different scales, and a biomedical image.

The decision about what patch size to use came from the original motivation, namely, creating a dataset in which the patches are small enough to be devoid of easy hints about the object from which the patch is derived. This means that the patch should be small enough not to contain the smallest semantically meaningful part of the smallest object contained in the images. For our images, the size of 50×50 pixels meets this criterion. We extracted 25 randomly located patches from each image, thus obtaining 375 patches in total. Each patch was segmented by 3 to 5 different subjects, resulting in a total of 1479 ground-truth patches. To ensure consistency across subjects, we developed a graphical user interface which let the subjects segment patches by drawing boundaries using a stylus pen on a tablet-pc. Note that this follows a dataset we introduced earlier in Akbas and Ahuja (2009). The dataset obtained in this work incorporated three improvements over (Akbas and Ahuja, 2009): (1) Ground-truth is obtained from multiple subjects (same patch is segmented by multiple subjects) instead of one as earlier. (2) Ground-truth is collected by asking the subjects to pen-trace the segments' borders on a horizontally laid tablet-pc, which is a more natural setting than drawing on a standard vertical (desktop) screen using a mouse as done by Akbas and Ahuja (2009). (3) Additional performance measures, based on regions alone and regions

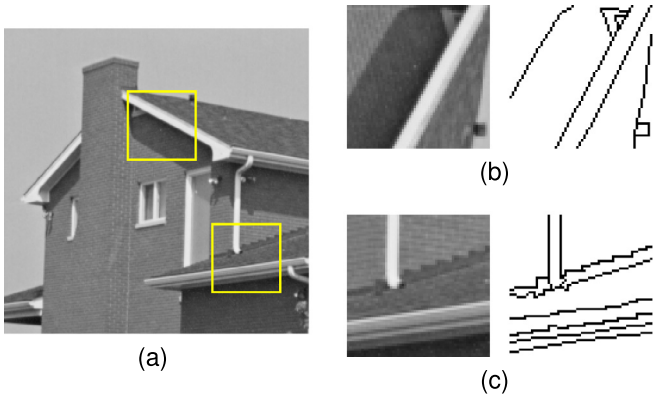

Fig. 14. (a) An image from our benchmark dataset. **(b)** The patch represented by the upper yellow square on (a) and its ground-truth segmentation. Note that the patch is rotated 90 degrees clockwise. **(c)** The other patch and its ground-truth segmentation.

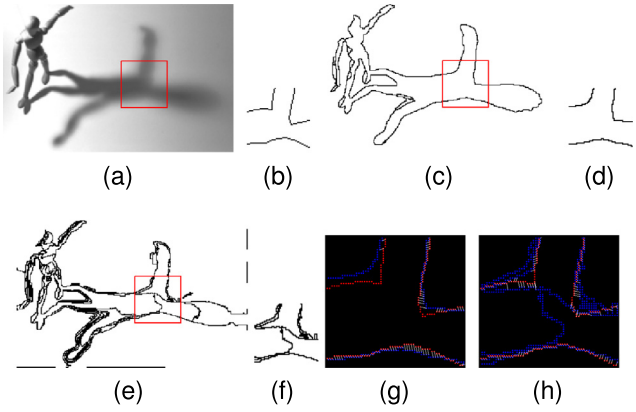

Fig. 15. Illustration of the boundary-based evaluation method for a single patch. (a) An image from our dataset. The patch of interest is marked by the red square. (b) Provided human segmentation for the patch. (c) Segmentation of (a) obtained by our algorithm. (d) Our result at the location of the patch. (e) Segmentation of (a) obtained by the mean-shift based algorithm. (f) Mean-shift's result at the location of the patch. **(g)** Matching result between the ground-truth (b) and our result (d). Red pixels represent the ground-truth, blue pixel represent algorithm's output (our result, in this case). White lines denote matching pixels. (h) Matching result between (b) and (f). See (g) for explanation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

combined with boundaries, are obtained and included as a part of the dataset.

Fig. 14 shows an image from the dataset and two example patches randomly extracted from it, along with their subject-obtained segmentations.

3.2.1. Performance measures

We evaluate the performance of a segmentation algorithm in terms of the agreement of the segmentation it provides with the ground-truth segmentations provided by human subjects. We measure the accuracy of a segmentation in terms of: region boundaries, region interiors and a combination of the two. Boundary-based evaluation measures how precisely and completely the ground-truth boundaries are detected, and the region-based evaluation measures how similar the algorithm's segmentation layout is to the ground-truth segmentation layout. The combined measure involves both of these measures.

Since each patch is segmented manually by multiple different humans, we can measure "human performance" by comparing these manual segmentations against each other using leave-one-out cross-validation. In particular, for each patch, we leave out one of the human segmentations as the test example, use the others as ground-truth

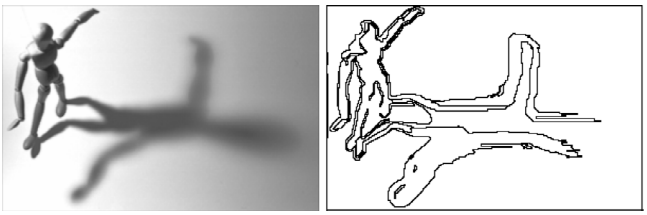

Fig. 16. Illustration of why a region-based evaluation method is needed. On the right is the segmentation result obtained by Felzenszwalb's algorithm (Felzenszwalb and Huttenlocher, 2004) (with parameters $\sigma = 0.5$, $k = 750$, and $a = 10$; see Section 3.2.5 for an explanation of parameters) for the mannequin image on the left. The region that corresponds to the shadow of the mannequin is not correctly segmented: it is merging with the background at the head's shadow (compare this result with the one given in Fig. 8). Boundary-based method does not penalize this error much because most of the shadow boundary is present, whereas the region-based method would assign a high penalty because the error eliminates an entire, large region, i.e. the shadow of the mannequin.

and measure performance. This process is repeated over all human segmentations pertaining to that patch. "Median human" refers to the median of leave-one-out cross-validation performances.

3.2.2. Boundary-based evaluation

We adopt the usual boundary-based evaluation method used in BSDS (Martin et al., 2001; Arbelaez et al., 2011) as in Martin et al. (2004). In this method, an optimal bipartite matching is computed between the pixels in the ground-truth boundary map and the segmentation boundary map, first. Then, the goodness of this match is expressed in terms of recall and precision. Recall measures the proportion of the ground-truth pixels that are matched by the segmentation, and the precision measures the proportion of the segmentation pixels that are matched. We illustrate this evaluation for a single patch in Fig. 15. To account for the multiple ground-truths for a single patch, a separate bipartite matching is computed between the segmentation patch and each of its ground-truth patches, and overall precision, recall are computed as done by Martin et al. (2004). When needed, we combine precision and recall values into the F-measure defined as: $f = \frac{2pr}{p+r}$.

3.2.3. Region-based evaluation

The boundary-based method falls short of penalizing a serious segmentation error: leakage. When there is a leakage on a boundary, the entire segment is lost, and yet the evaluation by the boundary-based method may yield a continuous, intermediate value, proportional to the fraction of the boundary matched, i.e. inversely proportional to the amount of leakage (Fig. 16). The region-based method, on the other hand, directly compares the complete partitions, not the corresponding boundaries alone.

We use the variation of information (VI) (Meila, 2003) as our region-based evaluation measure. VI is commonly used in the clustering literature and it measures the "distance " between two partitionings of the same set. It has also been used to evaluate segmentation performance (Arbelaez et al., 2009, 2011).

Given two segmentations X, Y of the same patch, VI is defined as:

$$\mathrm{VI}(X,Y) = H(X) + H(Y) - 2I(X,Y) \tag{5}$$

where H is the entropy,

$$H(X) = -\sum_{x} p(x)\log(p(x)), \tag{6}$$

and I is the mutual information function,

$$I(X,Y) = \sum_{y}\sum_{x} p(x,y)\log\left(\frac{p(x,y)}{p(x)p(y)}\right). \tag{7}$$

Here, lower-case letters represent regions; that is, x is a region in X. $p(x)$ and $p(x, y)$ are computed as:

$$p(x) = \frac{\#\text{ pixels in } x}{\#\text{ pixels in } X}, \quad p(x, y) = \frac{\#\text{ pixels in } (x \cap y)}{\#\text{ pixels in } X}. \tag{8}$$

$p(y)$ is computed in a similar way to $p(x)$ and $p(x, y) = p(y, x)$ since (# pixels in X) = (# pixels in Y) as X and Y are two different segmentations of the same patch, and they contain the same number of pixels.

3.2.4. Boundary and region combined evaluation

As illustrated in Fig. 16, there are cases where the boundary-based evaluation gives a good result while the region-based evaluation score is poor. To retain the quality of both, we combine both evaluations by simply making a convex combination of them. For the boundary-based evaluation part, we use the F-measure, so larger F-measure values indicate better segmentation. For the region-based part, we use VI wherein lower values represent better segmentation. To combine both, we subtract the F-measure from a positive constant c and then convexly combine it with VI, with α as the trade-off parameter:

$$\text{Combined} = \alpha(c - f) + (1 - \alpha)\text{VI} \tag{9}$$

The lower the value of the combined measure, the better the segmentation is. We describe how to select values for c and α in Section 3.2.6.

3.2.5. Algorithms

We compare our algorithm[1] with five available algorithms which are widely used in the literature: Felzenszwalb's graph-based algorithm[2] (Felzenszwalb and Huttenlocher, 2004), Multiscale NCuts[3] (Cour et al., 2005), the mean-shift algorithm[4] (Comaniciu and Meer, 2002), the gPB algorithm[5] (Martin et al., 2004; Arbelaez et al., 2009, 2011), and the MCG (Multiscale Combinatorial Grouping) algorithm.[6] (Arbelaez et al., 2014) Among these five algorithms, gPB and MCG are not considered low-level segmentation algorithms since their contour detector was trained on the BSDS dataset. Nonetheless, we include them here due to their popularity and to highlight the contrast between the results obtained by low and high level segmentation algorithms.

Both Felzenszwalb's algorithm and the mean-shift algorithm require three input parameters from the user and there is a large variability in the results depending on the values of these parameters. We sample a large number of input parameters for both algorithms. Mean-shift based segmentation method (Comaniciu and Meer, 2002) requires a spatial bandwidth σ_s, a range bandwidth σ_r, and a minimum region area a. We select the following input parameter space: $\{\sigma_s, \sigma_r, a\} \in \{5, 7, 9, 11, 15, 20, 25\} \times \{3, 6, 9, 12, 15, 18, 21, 24, 27\} \times \{5, 10\}$. The graph-based algorithm (Felzenszwalb and Huttenlocher, 2004) expects a smoothing scale σ, a threshold k which is the scale of observation (equation (5) in Felzenszwalb and Huttenlocher (2004)), and a minimum region size a, as input. We use the following parameter space: $\{\sigma, k, a\} \in \{0.5, 1, 1.5, 2\} \times \{250, 500, 750, 1000, 1250, 1500, 1750, 2000, 2250, 2500\} \times \{5, 10\}$. Multiscale NCuts takes a single input parameter which is the number of regions, n. We use $n \in \{20, 30, \ldots, 300\}$. MCG algorithm produces an ultrametric contour map for a given image. Then, one has to apply a threshold to it to obtain segmentations. We used 100 threshold values sampled from the range $[0, 1]$. gPB algorithm does not require any input parameters from the user and outputs segmentations at different scales ($\{0, 0.025, 0.05, 0.075, \ldots, 0.975, 1\}$). Like gPB, our algorithm does not require any input parameters and outputs a hierarchy of regions, the segmentation tree, whose regions are coming from segmentations at all photometric scales. That is, for 8-bit grayscale images the photometric scales used are $\{1, 2, \ldots, 255\}$.

[1] http://vision.ai.uiuc.edu/segmentation

[2] http://www.cs.brown.edu/~pff/segment/

[3] http://www.cis.upenn.edu/~jshi/software/

[4] http://coewww.rutgers.edu/riul/research/code/EDISON/index.html

[5] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

[6] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/mcg/#code

Table 1
Best possible performance (BPP) for the boundary-based evaluation. For each patch, the largest F-measure obtained is the best possible performance. Corresponding recall and precision values are also shown. Higher scores mean better segmentation. Best performances are shown in bold.

	Recall	Precision	F-measure
gPB	0.855	0.860	0.857
Graph-based	0.903	0.836	0.868
MCG	0.881	0.857	0.869
Mean-shift	0.912	0.878	0.895
NCuts	0.878	0.872	0.875
Segmentation tree (ours)	**0.937**	**0.891**	**0.913**
Median human	0.937	0.951	0.944

Table 2
Best possible performance (BPP) for the region-based evaluation as measured by variation of information (VI). Lower scores mean better segmentation. Best performance is shown in bold.

	Variation of information
gPB	0.639
Graph-based	0.718
MCG	0.672
Mean-shift	0.658
NCuts	0.735
Segmentation tree	**0.621**
Median human	0.421

3.2.6. Comparison of algorithms

For a given patch in the benchmark dataset, each algorithm has a set of different segmentation results depending on its input parameter values. In the case of our algorithm, different segmentation results corresponding to that patch can be obtained from the segmentation tree. To account for all these different segmentations, we report the boundary, region and combined-boundary-and-region performances in two different ways. In the first, the overall performance of an algorithm is computed by taking its *best result per patch*. We will refer to this measurement as the *best possible performance*, or BPP for short. The use of BPP is analogous to the multiple-segmentations (or the soup-of-segments) approaches (Russell et al., 2006; Malisiewicz and Efros, 2007; Endres and Hoiem, 2010; Carreira and Sminchisescu, 2012) where one or more segmentation algorithms are run with a large set of input parameter values, and a higher level algorithm picks the segmentation whichever it thinks is the best to solve a specific task at hand. In the second way, the overall performance is computed by using a fixed input parameter value set for all the patches in the dataset. We refer to this as the *performance per scale*, or PPS for short. PPS would thus suit commonly used segmentation algorithms which produce only a single image segmentation (not a set of them, or a hierarchy).

We give the boundary-based BPP results of algorithms along with the median human performance in Table 1. Our algorithm outperforms the other five algorithms in terms of recall, precision and F-measure. Mean-shift is the second best, followed, in order, by NCuts, MCG, Graph-based and gPB.

Table 2 presents the region-based BPP results. Again, our algorithm performs the best, however, the rest of the ranks (of gPB, Mean-shift, MCG, Graph-based, NCuts) here are different from those for the boundary-based evaluation above. This means that for the other five algorithms, best performing segmentation for the boundary-based evaluation is not always the best in terms of the region-based evaluation. This was illustrated in Fig. 16.

Next, we look at the combined-boundary-and-region performance evaluation (Eq. (9)). It has two free parameters c and α. We let α free

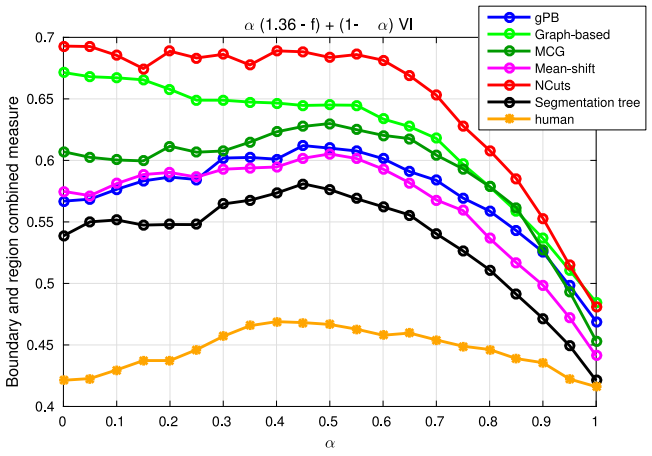

Fig. 17. Best possible performances (BPP) of algorithms for the boundary-and-region-combined evaluation (Section 3.2.4). The parameter c of the combined evaluation (Eq. (9)) is adjusted so that the human performance curve is as flat as possible. Lower scores (on the y-axis) mean better segmentation.

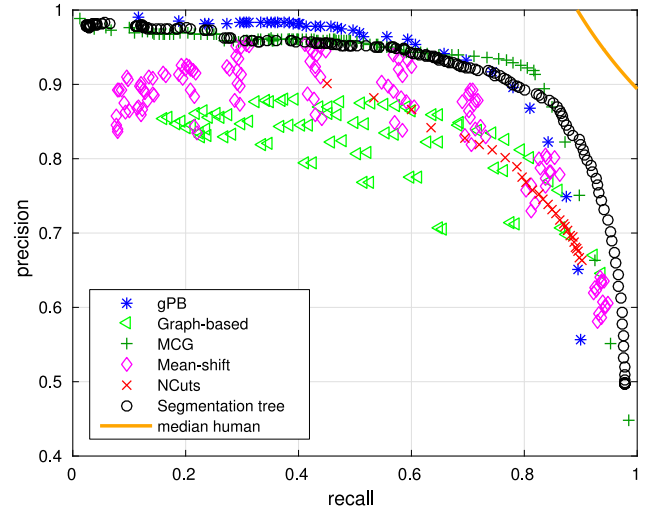

Fig. 18. Recall–precision plots. Each point corresponds to a fixed input parameter value set for its corresponding algorithm. Human performance is represented by the $f = 0.943$ curve shown in orange color. Closer points to the top-right corner mean better segmentation. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

and plot the combined score for all values of it. We pick a value for c so that median human performance is maximally close to a flat line, which effectively makes it independent of α. This value turns out to be 1.36. We present the plots of the combined scores in Fig. 17.

A few comments can be made about the results in Fig. 17. First, the ranking of algorithms from the best to worst is as follows: ours, Mean-shift, gPB, MCG, Graph-based, NCuts. This ranking stays the same in general except for $\alpha \in [0, 0.25]$ where gPB seems to be slightly better than Mean-shift, and for $\alpha \in [0.9, 1]$ where MCG performs slightly better than both gPB and Graph-based. Second, our algorithm consistently maintains a margin with its closest competitor, for all values of α. Finally, boundary-based performances (the rightmost part, where i.e. $\alpha = 1$) are confined to a smaller interval, and more scattered across it, compared to the region-based performances (the leftmost part, where i.e. $\alpha = 0$) where they cluster into two identifiable groups: (i) our algorithm, Mean-shift, gPB and MCG, and (ii) Graph-based, NCuts.

Next, we give the boundary-based and region-based performance per scale (PPS) results. In Fig. 18, Recall–precision points for different input parameter values can be found. Each point corresponds to a unique input parameter value set for its algorithm. Median human performance is marked by the curve where F-measure is equal to 0.943 (see Table 1). In this plot, the closer a point to the top-right corner, i.e. the perfect Recall–precision point, the better it is. The points that are closest to the human performance, and to the top-right corner, tend to be from our algorithm. However, in medium-low recall regime (recall < 0.7) gPB has better precision than ours. Region-based PPS evaluations are presented in Table 3, which also show that our algorithm outperforms others.

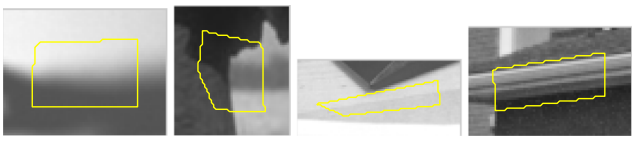

Fig. 19. Sample segmentation challenges as annotated in our benchmark dataset. Regions with yellow boundaries indicate challenging areas for segmentation algorithms. From left to right, the challenge types are blurred boundary, steep corner and junction, low contrast boundary, and thin region.

Table 3
Variation of information scores per input parameter value set. Lower scores mean better segmentation.

	Variation of information
gPB	0.738
Graph-based	0.872
MCG	0.756
Mean-shift	0.824
NCuts	0.846
Segmentation tree	**0.729**
Median human	0.421

Table 4
Performances of algorithms on *blurred* boundaries.

	Recall	Precision	F	VI	Combined
gPB	0.834	**0.910**	0.870	0.218	0.354
Graph-based	0.807	0.802	0.805	0.267	0.411
MCG	0.806	0.807	0.806	0.250	0.402
Mean-shift	0.882	0.849	0.865	0.233	0.364
NCuts	0.843	0.846	0.844	0.246	0.381
Seg. tree	**0.906**	0.839	**0.871**	**0.213**	**0.351**

Table 5
Performances of algorithms on *high curvature* boundaries which include steep corners and jagged edges.

	Recall	Precision	F	VI	Combined
gPB	0.846	**0.912**	0.878	0.185	0.334
Graph-based	0.914	0.857	0.885	0.198	0.337
MCG	0.881	0.855	0.868	0.198	0.345
Mean-shift	0.921	0.895	**0.908**	0.167	0.309
NCuts	0.872	0.872	0.872	0.192	0.340
Seg. tree	**0.925**	0.884	0.904	**0.157**	**0.306**

3.2.7. Performance evaluation on segmentation challenges

In order to provide some insights about which algorithm fails where, we manually annotated regions of interests for typical segmentation challenges (Ahuja, 1996; Tabb and Ahuja, 1997; Arora and Ahuja, 2006). We look at four challenge types: (i) blurred boundaries, (ii) high curvature boundaries which include steep corners, jagged edges, etc., (iii) low contrast boundaries, and (iv) thin regions. For each type, we collected 25 samples (See Fig. 19). Ground-truth for these regions of interest is obtained from the already discussed human segmentations. We use the same boundary and region based evaluations as in previous sections to analyze the performance of segmentation algorithms on selected challenges. We believe that such an analysis would provide valuable insights into how to improve the algorithms.

Table 6
Performances of algorithms on *low contrast* boundaries.

	Recall	Precision	F	VI	Combined
gPB	0.800	**0.832**	0.815	0.196	0.370
Graph-based	0.818	0.810	0.814	0.171	0.359
MCG	0.832	0.759	0.794	0.208	0.387
Mean-shift	0.826	0.817	0.822	0.208	0.373
NCuts	0.790	0.800	0.795	0.191	0.378
Seg. tree	**0.851**	**0.832**	**0.841**	**0.164**	**0.341**

Table 7
Performances of algorithms on *thin regions*.

	Recall	Precision	F	VI	Combined
gPB	0.774	0.895	0.830	0.219	0.374
Graph-based	0.884	0.867	0.876	0.191	0.338
MCG	0.882	0.870	0.876	0.221	0.352
Mean-shift	0.889	**0.904**	0.896	**0.171**	0.317
NCuts	0.851	0.889	0.869	0.212	0.351
Seg. tree	**0.941**	0.888	**0.914**	**0.171**	**0.309**

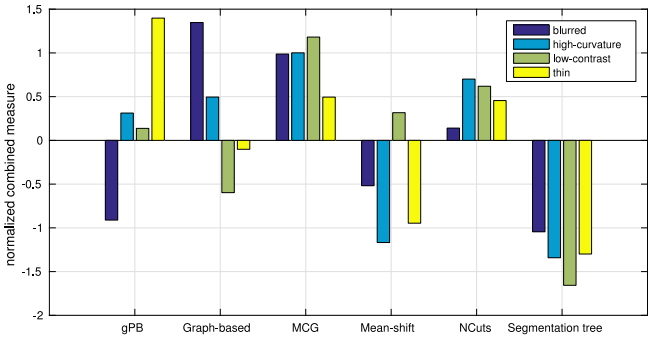

Fig. 20. Performance comparison on the segmentation challenge types. Normalized combined-boundary-and-region performance is used for comparison.

Tables 4–7 give boundary, region and combined evaluation results for the four challenge types. Best results are marked in bold. To compute the combined measure, we used $c = 1.36$ for maximal independence from α, as discussed in Section 3.2.6, and therefore an arbitrary value for α, 0.5. In all four categories, our algorithm performs the best. Rest of the rankings vary according to the type of challenge.

In order to compare algorithms across different challenges and to see how failure is distributed on different types of challenges for a specific algorithm, we do the following post-processing on the combined evaluation scores. For each challenge type, we take all the combined scores and transform them so that they have zero mean and unit standard deviation, to obtain the "normalized combined scores ". Fig. 20 shows these normalized scores. Here, the baseline, i.e. the 0-line, can be considered as the performance of an average segmentation algorithm; and the lower the score, the better the performance. Our algorithm clearly outperforms the others. In fact, it is the only algorithm which does better than the average segmentation algorithm in all the four challenge types. The next best algorithm appears to be Mean-shift, since it does better than average in 3 of the challenge types. Next, in order, are Graph-based, gPB, NCuts and MCG.

Fig. 20 allows us to make comments on how failure is distributed over different challenge types for a specific algorithm. This is possible because combined scores are normalized per challenge type over all segmentation algorithms. gPB algorithm seems to be suffering from thin regions the most. Blurred boundaries is the least concern for gPB. For the Graph-based algorithm, the performance bottleneck appears to be the blurred boundaries and the high-curvature boundaries. Mean-shift is significantly poorer at low-contrast boundaries, while it is particularly good at high-curvature boundaries and thin regions. NCuts' and MCG's performances are more or less uniformly poorer across all the categories. The failures of our algorithm are also more or less uniformly distributed across all challenge categories.

Finally, we provide segmentation results from different algorithms for qualitative evaluation and comparison in Fig. 21. As each algorithm produces a multi-scale segmentation, presenting them in a single figure is a challenge. As a solution, we present a single segmentation result per algorithm, which corresponds to the best result (i.e. the one with the lowest error) based on the boundary-and-region-combined measure described in Section 3.2.4. We chose the best four algorithms based on the ranking given by the same measure (in Fig. 17). These are: our algorithm, gPB, mean-shift and MCG. Examples of under and over segmentation errors can be easily seen on the segmentation outputs. Qualitatively, our segmentation results appear to have fewer errors compared to the outputs of other algorithms (see Fig. 21).

4. Conclusion

We have presented a new multiscale algorithm for low-level image segmentation. The algorithm is capable of detecting low-level image segments, each defined by its own, arbitrary level of photometric homogeneity, and arbitrary shape. A low-level image structure, or region, is defined as a connected set of pixels surrounded by ramp discontinuities. To detect regions, the image is converted to a ramp magnitude map. Then we find the basins of the ramp magnitude map and use them as region seeds. These seeds are grown by a watershed procedure to get the final segmentation. After this, we obtain multi-photometric scale segmentation by doing a multiscale analysis over a range of scales where, at each scale, boundary fragments having less contrast than the scale value are removed. This process guarantees a strict hierarchy. Using this property, we arrange the regions into a tree data structure which we call the segmentation tree.

We have also presented a new benchmark dataset to evaluate the performance of low-level segmentation algorithms. The dataset consists of a number of image patches along with their ground-truth segmentations done by human subjects. Our evaluations are in terms of accuracies of the detected segment boundaries, entire segments, and their best combinations. Boundary-based evaluation measures how precisely and completely the ground-truth boundaries are detected, while the region-based evaluation measures how similar the algorithm's partitioning of the image, i.e. segmentation, is to the ground-truth partitioning. We have used the benchmark to quantitatively compare the performance of our algorithm to those of existing, popular approaches, and shown that our algorithm outperformed others in all our evaluations. In addition, our benchmark dataset contains a set of annotated segmentation challenges such as low-contrast boundaries, corners, junctions, etc. This enables us to evaluate the performances of algorithms on these challenge types and give a breakdown of errors across them which could be used to improve the segmentation algorithms.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The support of the Office of Naval Research, USA under grant N00014-06-1-0101 is gratefully acknowledged. We also acknowledge Dennis Kraft's help (Kraft, 2015) in proving the NP-hardness of PSRσ. We thank Akash Sonth for helping with some of the experiments and figures.

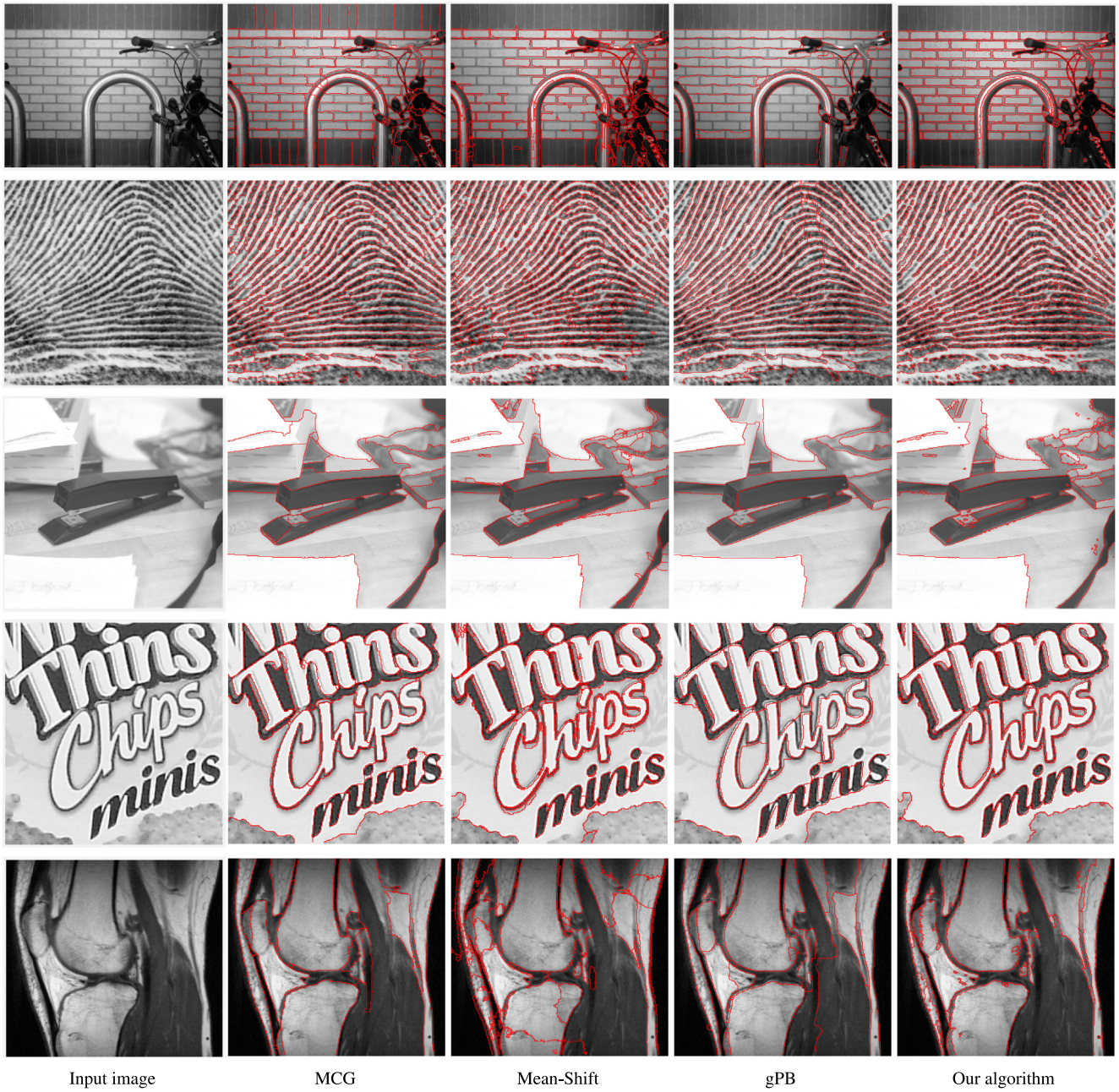

Fig. 21. Best single-scale segmentation result generated by different algorithms, presented for qualitative comparison. For each algorithm, we picked the best result, that is the one with the lowest error with respect to the ground-truth based on the boundary-and-region-combined measure (Section 3.2.4). Easily seen under-segmentation errors can be seen in the first row, on the bricks of the wall; in the second row, on the fingerprint patterns; and in the last row, around the light-colored regions. Examples of over-segmentation can be easily observed in the fourth row, around and within the letters; in the third row, on the black region in the bottom-right corner. Qualitatively, our segmentation results (last column) appear to have fewer errors compared to the outputs of other algorithms. (Best viewed in color and when zoomed-in). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Appendix. NP-hardness of PSRσ

NP-hardness of the problem PSRσ, namely the problem of computing photometrically stable regions, can be shown by a reduction from the graph k-coloring problem. Whether the nodes of a given graph can be colored using exactly k unique colors in such a way that no adjacent nodes are colored with the same color is called the graph k-coloring problem, and is known to be an NP-complete problem (Garey et al., 1976; Havet, 2015). Here we present a polynomial time reduction from an instance of the graph k-coloring problem to an instance of the PSRσ problem.

Suppose we are given a graph $G = (V, E)$ and we want to decide if G is k-colorable. To solve this decision problem, we construct a new graph G' that consists of G as well as the complete graph $K_k = (W, F)$. For each pair of nodes (v, w) where $v \in V$ and $w \in W$, we add an edge between v and w with weight -1. To all other edges, i.e. $\forall e \in E \cup F$, we assign very large (positive) weights.

With this construction, we claim that G is k-colorable, if and only if PSRσ solution on G' has a cost of $-|V|$.

Proof. Suppose G is k-colorable. Consider the following coloring scheme on G':

1. K_k is colored with exactly k colors in such a way that each node gets a unique color,
2. Each node in G picks a color from the k colors in K_k,

3. Any two adjacent nodes in G do not share the same color.

Note that conditions (2) and (3) are possible because G is k-colorable.

The PSRσ cost, that is the sum of the weights of the monochromatic edges, for the coloring scheme given above is $-|V|$ because the only monochromatic edges are between G and K_k, there is one per node in G and each has -1 weight. We claim that this is the optimal, i.e. minimum cost, PSRσ solution. To see this, consider the cases where we depart from the given coloring scheme. If we used less than k colors, the cost would have been higher due to the large weights within G and K_k. On the other hand, if we used more than k colors, the cost would have increased due to the loss of monochromatic edges with -1 weights.

Next, suppose that G is not k-colorable. This implies that conditions (2) and (3) above are not possible, which results in a PSRσ cost higher than $-|V|$.

References

Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S., 2012. SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans. Pattern Anal. Mach. Intell. 34 (11), 2274–2282. http://dx.doi.org/10.1109/TPAMI.2012.120.

Ahuja, N., 1996. A transform for multiscale image segmentation by integrated edge and region detection. IEEE Trans. Pattern Anal. Mach. Intell. 18 (12), 1211–1235. http://dx.doi.org/10.1109/34.546258.

Akbas, E., Ahuja, N., 2009. From ramp discontinuities to segmentation tree. In: ACCV. Springer, Xi'an, China, pp. 123–134 (oral presentation).

Akbas, E., Ahuja, N., 2010. Low-level image segmentation based scene classification. In: ICPR. IEEE, pp. 3623–3626.

Akbas, E., Ahuja, N., 2014. Low-level hierarchical multiscale segmentation statistics of natural images. IEEE Trans. Pattern Anal. Mach. Intell. 36 (9), 1900–1906.

Arbelaez, P., Maire, M., Fowlkes, C., Malik, J., 2009. From contours to regions: An empirical evaluation. In: CVPR. http://dx.doi.org/10.1109/CVPRW.2009.5206707.

Arbelaez, P., Maire, M., Fowlkes, C., Malik, J., 2011. Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 33 (5), 898–916.

Arbelaez, P., Pont-Tuset, J., Barron, J.T., Marques, F., Malik, J., 2014. Multiscale combinatorial grouping. In: CVPR.

Arora, H., Ahuja, N., 2006. Analysis of ramp discontinuity model for multiscale image segmentation. In: ICPR. http://dx.doi.org/10.1109/ICPR.2006.270.

Bleau, A., Leon, L., 2000. Watershed-based segmentation and region merging. Comput. Vis. Image Underst. 77 (3), 317–370. http://dx.doi.org/10.1006/cviu.1999.0822, URL: http://www.sciencedirect.com/science/article/pii/S1077314299908226.

Carreira, J., Sminchisescu, C., 2012. CPMC: Automatic object segmentation using constrained parametric min-cuts. IEEE Trans. Pattern Anal. Mach. Intell.

Carson, C., Belongie, S., Greenspan, H., Malik, J., 2002. Blobworld: Image segmentation using expectation-maximization and its application to image querying. IEEE Trans. Pattern Anal. Mach. Intell. 24 (8), 1026–1038. http://dx.doi.org/10.1109/TPAMI.2002.1023800.

Cheng, H.-T., Ahuja, N., 2012. Exploiting nonlocal spatiotemporal structure for video segmentation. In: CVPR. IEEE, pp. 741–748.

Comaniciu, D., Meer, P., 2002. Mean shift: A robust approach toward feature space analysis. IEEE Trans. Pattern Anal. Mach. Intell. 24 (5), 603–619. http://dx.doi.org/10.1109/34.1000236.

Comaniciu, D., Ramesh, V., Meer, P., 2001. The variable bandwidth mean shift and data-driven scale selection. In: ICCV, pp. 438–445.

Cour, T., Benezit, F., Shi, J., 2005. Spectral segmentation with multiscale graph decomposition. In: CVPR, pp. 1124–1131. http://dx.doi.org/10.1109/CVPR.2005.332.

Donoser, M., Schmalstieg, D., 2014. Discrete-continuous gradient orientation estimation for faster image segmentation. In: CVPR, pp. 3158–3165.

Elder, J.H., Zucker, S.W., 1998. Local scale control for edge detection and blur estimation. IEEE Trans. Pattern Anal. Mach. Intell. 20 (7), 699–716. http://dx.doi.org/10.1109/34.689301.

Endres, I., Hoiem, D., 2010. Category independent object proposals. In: CVPR.

Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A., 2010. The PASCAL visual object classes challenge 2010 results. http://www.pascal-network.org/challenges/VOC/voc2010/workshop/index.html.

Felzenszwalb, P.F., Huttenlocher, D.P., 2004. Efficient graph-based image segmentation. Int. J. Comput. Vis. 59 (2), 167–181. http://dx.doi.org/10.1023/B:VISI.0000022288.19776.77.

Forsyth, D.A., Ponce, J., 2002. Computer Vision: A Modern Approach. Prentice Hall Professional Technical Reference.

Garey, M., Johnson, D., Stockmeyer, L., 1976. Some simplified NP-complete graph problems. Theoret. Comput. Sci. 1 (3), 237–267. http://dx.doi.org/10.1016/0304-3975(76)90059-1, URL: http://www.sciencedirect.com/science/article/pii/0304397576900591.

Geman, S., Geman, D., 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. 6 (11), 721–741.

Ghanem, B., Ahuja, N., 2013. Modeling dynamic swarms. Comput. Vis. Image Underst. 117 (1), 1–11.

Haralick, R.M., Shapiro, L.G., 1985. Survey- image segmentation techniques. Comput. Vis. Graph. Image Process. 29, 100–132.

Havet, F., 2015. Graph colouring. In: Combinatorial Optimization Lecture Notes. URL: http://www-sop.inria.fr/members/Frederic.Havet/Cours/coloration.pdf. (Last accessed November, 2015).

Kim, S., Yoo, C.D., Nowozin, S., Kohli, P., 2014. Image segmentation using higher-order correlation clustering. IEEE Trans. Pattern Anal. Mach. Intell. 36 (9), 1761–1774.

Kraft, D., 2015. Minimum edge deletion partitioning. Computer Science Stack Exchange. URL: http://cs.stackexchange.com/q/45610 (version: 2015-08-27).

Malisiewicz, T., Efros, A.A., 2007. Improving spatial support for objects via multiple segmentations. In: BMVC.

Martin, D.R., Fowlkes, C.C., Malik, J., 2004. Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Trans. Pattern Anal. Mach. Intell. 26 (5), 530–549. http://dx.doi.org/10.1109/TPAMI.2004.1273918.

Martin, D., Fowlkes, C., Tal, D., Malik, J., 2001. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: ICCV, Vol. 2, pp. 416–423.

Matas, J., Chum, O., Urban, M., Pajdla, T., 2002. Robust wide baseline stereo from maximally stable extremal regions. In: Rosin, P.L., Marshall, A.D. (Eds.), BMVC. British Machine Vision Association.

Meila, M., 2003. Comparing clusterings by the variation of information. In: Proc. Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory (COLT) and 7th Kernel Workshop, pp. 173–187.

Meyer, F., Beucher, S., 1990. Morphological segmentation. J. Vis. Commun. Image Represent. 1 (1), 21–46.

Moaveni, M., Wang, S., Hart, J., Tutumluer, E., Ahuja, N., 2013. Evaluation of aggregate size and shape by means of segmentation techniques and aggregate image processing algorithms. Transp. Res. Rec. J. Transp. Res. Board (2335), 50–59.

Nam, M., Ahuja, N., 2012. Learning human preferences to sharpen images. In: ICPR. IEEE, pp. 2173–2176.

Ren, Z., Shakhnarovich, G., 2013. Image segmentation by cascaded region agglomeration. In: CVPR, pp. 2011–2018.

Russell, B.C., Efros, A.A., Sivic, J., Freeman, W.T., Zisserman, A., 2006. Using multiple segmentations to discover objects and their extent in image collections. In: CVPR.

Sahoo, P.K., Soltani, S., Wong, A.K., Chen, Y.C., 1988. A survey of thresholding techniques. Comput. Vis. Graph. Image Process. 41 (2), 233–260. http://dx.doi.org/10.1016/0734-189X(88)90022-9.

Shi, J., Malik, J., 2000. Normalized cuts and image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. 22 (8), 888–905. http://dx.doi.org/10.1109/34.868688.

Singh, A., Ahuja, N., 2012. Exploiting ramp structures for improving optical flow estimation. In: ICPR.

Singh, A., Ahuja, N., 2013. Single image super-resolution using adaptive domain transformation. In: ICIP. IEEE, pp. 947–951.

Singh, A., Ahuja, N., 2015. Learning ramp transformation for single image super-resolution. Comput. Vis. Image Underst. 135, 109–125.

Stockman, G., Shapiro, L.G., 2001. Computer vision. Prentice Hall PTR, Upper Saddle River, NJ, USA.

Tabb, M., Ahuja, N., 1997. Multiscale image segmentation by integrated edge and region detection. IEEE Trans. Image Proc. 6 (5), 642–655.

Xu, C., Prince, J.L., 1998. Snakes, shapes, and gradient vector flow. IEEE Trans. Image Proc. 7 (3), 359–369.

Yu, Y., Fang, C., Liao, Z., 2015. Piecewise flat embedding for image segmentation. In: ICCV.

Zabih, R., Kolmogorov, V., 2004. Spatially coherent clustering using graph cuts. In: CVPR, pp. 437–444.