From Ramp Discontinuities to Segmentation Tree

Emre Akbas and Narendra Ahuja

Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign

Abstract. This paper presents a new algorithm for low-level multiscale segmentation of images. The algorithm is designed to detect image regions regardless of their shapes, sizes, and levels of interior homogeneity, by doing a multiscale analysis without assuming any prior models of region geometry. As in previous work, a region is modeled as a homogeneous set of connected pixels surrounded by ramp discontinuities. A new transform, called the ramp transform, is described, which is used to detect ramp discontinuities and seeds for all regions in an image. Region seeds are grown towards the ramp discontinuity areas by utilizing a relaxation labeling procedure. Segmentation is achieved by analyzing the output of this procedure at multiple photometric scales. Finally, all detected regions are organized into a tree data structure based on their recursive containment relations. Experiments on real and synthetic images verify the desired properties of the proposed algorithm.

1 Introduction

Low-level image segmentation partitions a given image into regions which are characterized by some low-level properties of their constituent pixels, where the term "low-level" refers to local and intrinsic properties of pixels such as gray-level intensity (or color), contrast, gradient, etc. For a set of connected pixels to form a region, they should have a certain degree of interior homogeneity and a discontinuity with their surroundings, where the magnitude of discontinuity is large compared to the interior variation.

Our goal is to detect image regions regardless of their shapes, sizes and levels of interior homogeneity. These goals preclude the use of any prior model about region shape or geometry, and the fact that a region can have any level of homogeneity requires us to do multiscale analysis. Furthermore, we want the algorithm to work without requiring any image-dependent parameter tuning. Achieving these goals are challenging because of the nature of discontinuities that separate regions. A sharp edge in 3D world might be mapped to a wide ramp discontinuity in the image due to defocussing and penumbral blur in the image acquisition process. Hence, region boundaries in images, which can have arbitrary shapes, are surrounded by ramp discontinuities of various widths and heights.

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 [8], region-growing [8] and watersheds [11] ignore the multiscale aspect of the problem. Energy minimization based approaches such as Markov Random Field modeling [7] and active contours [14] enforce constraints on the

local shape of regions; therefore, they are not capable of detecting arbitrarily shaped regions. Graph theoretical methods such as Normalized Cuts [13] and graph cuts [15] requires the number of regions to be given as input, which does not guarantee the detection of regions 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 density estimation parameters [3,4], as input. As discussed in [2], 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 have been developed by aiming to maximize performance on the Berkeley Segmentation Benchmark Dataset [9] which contains images segmented by humans. We note that these are object-level segmentations and many regions in the images are not marked (*i.e.* many edges are not marked even if there is strong visual evidence for an edge, or some edges are marked where there is too little or no visual evidence). It is not our goal, in this study, to segment objects out, 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 higher level vision algorithms.

To this end, we follow the line of research in [1] and [2], and develop a new algorithm to achieve the aforementioned goals. As in [1], we use gray-level intensity and contrast as the low-level properties to define and detect low-level image structures. We define an image region as a set of connected pixels surrounded by ramp discontinuities, as done in [2]. We model ramp discontinuities with strictly increasing (or decreasing) intensity profiles. Each ramp discontinuity has a magnitude, or contrast, which allows us to associate a photometric scale with each boundary fragment surrounding regions. We achieve a multiscale segmentation over a range of scales, by progressively removing boundary fragments whose photometric scales are less than the current scale of analysis. Finally, all regions detected at all photometric scales are organized into a tree data structure according to their recursive containment relations.

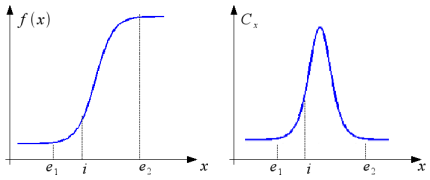
In this paper, we propose a new method, called the ramp transform, for detection of ramps and estimation of ramp parameters. At a given pixel, we analyze multiple intensity profiles passing through the pixel in different directions, and estimate the magnitude of the ramp discontinuity at that pixel by minimizing a weighted error measure. After applying the ramp transform, seeds for all image regions are detected and these seeds grow to become complete regions.

Our contributions are: (1) A new segmentation algorithm which detects image regions of all shapes, sizes and levels of intensity homogeneity. It arranges regions into a tree data structure which can be used in high level vision problems. The algorithm gives better results and is less sensitive to image-dependent parameter tuning, compared to the existing popular segmentation algorithms. (2) A new transform, called the ramp transform, which takes an image and outputs another image where the value at each pixel gives a measure of ramp discontinuity magnitude at that pixel in the original image, is proposed. (3) A new ground-truth dataset for low-level image segmentation. To the best of our knowledge, this is the first dataset of its kind for such purpose. This dataset could also be used as a benchmark for edge detection algorithms.

The rest of the paper is organized as follows. Section 2 describes the ramp and region models, the ramp transform and the segmentation algorithm. In Section 3, we present and discuss experimental results, and the paper is concluded in Section 4.

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 are larger than both the local intensity variation and the magnitudes of discontinuities, within R. To elaborate this definition, we first describe the ramp discontinuity model and the ramp transform.



(a) (b)

Fig. 1. (a) Ramp model. (b) Ramp transform of $f(x)$. C_i is equal to $|f(i+a) - f(i-a)|$ where $a = \min\{|i-e_1|, |i-e_2|\}$.

Ramp model. Consider the 1-dimensional image f given in Fig.1(a). A ramp is characterized by its strictly increasing (or decreasing) profile. So, the part of the curve between e_1 and e_2 is a ramp. The width of the ramp is $|e_1 - e_2|$, and its magnitude is $|f(e_2) - f(e_1)|$. Additionally, we define two measures, "ramp quality" and "point-magnitude", which will help us to generalize the ramp model to 2D functions.

We define the "ramp quality" as the ratio between its magnitude and width, namely: $\frac{|f(e_2)-f(e_1)|}{|e_1-e_2|}$. The "point-magnitude" at location i is defined as:

$$C_i = |f(i+a) - f(i-a)| \tag{1}$$

where $a = \min\{|i-e_1|, |i-e_2|\}$. Note that the ramp magnitude and point-magnitude are different measurements. Both are equal only when the point i is located at equal distances to the endpoints, e_1 and e_2, and $f(x-i)$ is an odd function.

In 2D images, computing the point-magnitude of the ramp discontinuity at $\boldsymbol{i}$, *i.e.* $C(\boldsymbol{i})$, is not a trivial task as it is in the 1D case. This is because an infinite number of lines, hence ramp intensity profiles, pass through the pixel $\boldsymbol{i}$. We assume that a pixel $\boldsymbol{i}$ is within a ramp discontinuity if it has at least one strictly increasing (or decreasing) intensity profile passing through it.

2.1 Ramp transform

The transform converts the input image I to a scalar height field C. The height at location $\boldsymbol{i}$ in C corresponds to the point-magnitude of the ramp discontinuity at $\boldsymbol{i}$ in the original image I.

If I is a 1D image, computing the ramp transform amounts to computing C_i for all i by setting $f = I$ in eq.(1). The ramp transform of the ramp of Fig.1(a) is given in Fig.4.

If I is a 2D image, an infinite number of lines passes through $\boldsymbol{i}$, and each of these lines has its own intensity profile. To parametrize these lines –and their corresponding

intensity profiles– let us define an angle θ which is the angle that the line makes with a horizontal row of the image. For a finite set of angles in $[0, 2\pi)$, we analyze the intensity profiles and measure the corresponding ramp parameters. For the ramp discontinuity at angle θ, let $q_{\boldsymbol{i}}^{\theta}$ be its ramp quality and $c_{\boldsymbol{i}}^{\theta}$ its point-magnitude at $\boldsymbol{i}$.

It is tempting to set $C_{\boldsymbol{i}} = \max_{\theta} c_{\boldsymbol{i}}^{\theta}$ but it gives us noisy estimates since ramp end points are affected by the noise present in images. To be robust to this noise, we estimate $C_{\boldsymbol{i}}$ in a weighted least-squares setting using the ramp quality measures as weights:

$$\hat{C}_{\boldsymbol{i}} = \min \textstyle\sum_{\theta} q_{\boldsymbol{i}}^{\theta} (C_{\boldsymbol{i}} - c_{\boldsymbol{i}}^{\theta})^2 \implies \hat{C}_{\boldsymbol{i}} = \frac{\sum_{\theta} q_{\boldsymbol{i}}^{\theta} c_{\boldsymbol{i}}^{\theta}}{\sum_{\theta} q_{\boldsymbol{i}}^{\theta}} \tag{2}$$

In the following, we drop the hat of $\hat{C}_{\boldsymbol{i}}$, and use $C_{\boldsymbol{i}}$.

2.2 Obtaining seeds for regions

The output, C, of the ramp transform contains point-magnitudes of the ramp discontinuities found in I. In this section, we first describe our region model and then elaborate on how seeds for all image regions are detected from C.

Region model. A set of connected pixels, R, is said to form a *region* if: 1) it is surrounded by ramp discontinuities, 2) the magnitudes of these discontinuities are larger than both the local intensity variation and the magnitudes of discontinuities, within R.

To obtain regions that conform with the above definition, we look for the basins of the height field C. To find all basins, all photometric scales, *i.e.* contrast levels, are traversed from lower to higher and new basins are marked as they occur. Then, the set of basin pixels, $\mathcal{S}$, contains the seeds for image regions. The remaining set of pixels, $\mathcal{D}$, correspond to the ramp discontinuity areas, and we call these pixels as ramp pixels.

We find the connected-components in the set $\mathcal{S}$, and label each component, which corresponds to a distinct basin of C, with a unique label. If there are N connected-components, then each pixel in the set S takes a label from the set $\mathcal{L} = \{1, 2, 3, \ldots, N\}$.

2.3 Region growing by relaxation labeling

Having obtained the regions seeds ($\mathcal{S}$), we want to grow them by propagating labels towards the ramp discontinuity areas ($\mathcal{D}$) which are unlabeled. For this purpose, we use a relaxation labeling [12] procedure. Although the classical watershed transform might be utilized here, we choose not to use it because it does not give good edge-location accuracy at corners and junctions.

Let $\boldsymbol{i} \leftarrow \ell$ denote the event of assigning label ℓ to pixel $\boldsymbol{i}$, and $P^{(t)}(\boldsymbol{i} \leftarrow \ell)$ denote the probability of this event happening at iteration t. Relaxation labeling iteratively updates these probabilities so that the labeling of pixels get more consistent as the method iterates. Next, we describe how we compute the initial probability values.

Computing Priors. For a pixel that is part of any detected region seed, we define the prior probabilities as $P^{(0)}(\boldsymbol{i} \leftarrow \ell) = 1$ if $\boldsymbol{i} \in R_{\ell}$ (0, else) $\forall \boldsymbol{i} \in \mathcal{S}, \forall \ell \in \mathcal{L}$.

On the other hand, the prior probabilities of the pixels which are within the ramp discontinuities, *i.e.* those $\boldsymbol{i} \in \mathcal{D}$, are not trivial. To compute these priors, we design a cost function for assigning label ℓ to pixel $\boldsymbol{i}$, *i.e.* the event $\boldsymbol{i} \leftarrow \ell$, as follows.

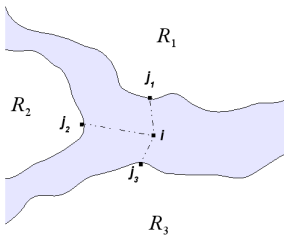


Fig. 2. Computing the initial probabilities for relaxation labeling.

Consider the scenario given in Fig.2. Pixel $\boldsymbol{i}$ is within the ramp discontinuity area among regions R_1, R_2, and R_3. The point $\boldsymbol{j}_1$ is the closest point to $\boldsymbol{i}$, in region R_1 (similarly $\boldsymbol{j}_2$ for R_2 and $\boldsymbol{j}_3$ for R_3). Let $\overline{ij_1}$ denote the line segment connecting $\boldsymbol{i}$ and $\boldsymbol{j}_1$. We compute the cost of assigning label 1 to ramp pixel $\boldsymbol{i}$ by analyzing the intensity profile along the line segment $\overline{ij_1}$. The cost function is designed in such a way that the flatter the profile is, the lesser the cost, and vice versa. We achieve this by summing up finite differences of the intensity profile at regular intervals. Formally, the cost of assigning label ℓ to a ramp pixel $\boldsymbol{i}$ is given by:

$$G(\boldsymbol{i} \leftarrow \ell) = \sum_{n=1}^{\|\boldsymbol{i}-\boldsymbol{j}\|/h} |I_{\boldsymbol{i}+nh\boldsymbol{u}} - I_{\boldsymbol{i}+(n-1)h\boldsymbol{u}}| \tag{3}$$

where $\boldsymbol{j}$ is the closest pixel to $\boldsymbol{i}$ such that $\boldsymbol{j} \in R_\ell$, h is a small stepsize, $\|\boldsymbol{i}-\boldsymbol{j}\|$ is the distance between $\boldsymbol{i}$ and $\boldsymbol{j}$, and $\boldsymbol{u} = \frac{\boldsymbol{j}-\boldsymbol{i}}{\|\boldsymbol{i}-\boldsymbol{j}\|}$, a unit vector.

To compute prior probabilities for a ramp pixel $\boldsymbol{i}$, we use:

$$P^{(0)}(\boldsymbol{i} \leftarrow \ell) = \frac{G^{-1}(\boldsymbol{i} \leftarrow \ell)}{\sum_{k \in \mathcal{L}} G^{-1}(\boldsymbol{i} \leftarrow k)}, \text{for } \forall \boldsymbol{i} \in \mathcal{D}, \forall \ell \in \mathcal{L}. \tag{4}$$

Relaxation labeling. Once the probabilities are initialized by $P^{(0)}(\cdot)$,we iteratively update them by the following relaxation labeling update rule:

$$P^{(t+1)}(\boldsymbol{i} \leftarrow \ell) = \frac{P^{(t)}(\boldsymbol{i} \leftarrow \ell)(1+Q^{(t)}(\boldsymbol{i} \leftarrow \ell))}{\sum_{k \in L} P^{(t)}(\boldsymbol{i} \leftarrow k)(1+Q^{(t)}(\boldsymbol{i} \leftarrow k))} \tag{5}$$

where $Q(\cdot)$ is defined as:

$$Q^{(t)}(\boldsymbol{i} \leftarrow \ell) = \frac{1}{|\mathcal{N}_{\boldsymbol{i}}|} \sum_{\boldsymbol{j} \in \mathcal{N}_{\boldsymbol{i}}} \sum_{k \in L} R_{\boldsymbol{ij}}(\ell, k) P^{(t)}(\boldsymbol{j} \leftarrow k). \tag{6}$$

Here $\mathcal{N}_{\boldsymbol{i}}$ denotes the neighbors of pixel $\boldsymbol{i}$ and $R_{\boldsymbol{ij}}(\ell, k)$, called the compatibility function, gives a measure of how compatible the assignments "$\boldsymbol{i} \leftarrow \ell$" and "$\boldsymbol{j} \leftarrow k$" are. The constraint on $R(\cdot)$ is that it should return values in the interval $[-1, +1]$: 1 meaning that the two events are highly compatible, -1 meaning just the opposite. We choose the following form:

$$R_{\boldsymbol{ij}}(\ell, k) = \begin{cases} e^{-\frac{|I_{\boldsymbol{i}} - I_{\boldsymbol{j}}|}{s}} & , \ell = k \\ e^{-\frac{\mathcal{M} - |I_{\boldsymbol{i}} - I_{\boldsymbol{j}}|}{s}} & , \ell \neq k \end{cases} \tag{7}$$

where $\mathcal{M}$ is the maximum value that $|I_{\boldsymbol{i}} - I_{\boldsymbol{j}}|$ can take for any I, $\boldsymbol{i}$, $\boldsymbol{j}$. It is 255 for standard 8-bit images. This compatibility function forces the neighboring pixels with similar intensities to have the same labels.

Final labeling of ramp pixels. When the highest change in any $P^{(t)}(\cdot)$ becomes very small, we stop the iterations and label the ramp pixels with the labels having the maximum probabilities: $\boldsymbol{i} \leftarrow \arg\max_{\ell} P^{(t)}(i \leftarrow \ell)$.

2.4 Multiscale segmentation

After relaxation labeling, every pixel in the image has a label, hence every pixel has joined one of the seeds detected after the ramp transformation step. Now we analyze these regions at a finite set of photometric scales, *i.e.* contrast levels, and produce multiscale segmentation of the image.

Segmentation of I at a given contrast level σ is defined as the partitioning of I into regions where all boundary fragments have a photometric scale larger than or equal to σ. A boundary fragment is defined to be a connected set of boundary pixels separating two neighboring regions. For a given contrast level σ, a fragment f is said to have lower photometric scale than σ, if it satisfies: $\frac{1}{|f|}\sum_{\boldsymbol{p}\in f}\mathbb{I}_{\{C(\boldsymbol{p})<\sigma\}} < \alpha$ where $|f|$ denotes the length of the fragment and α is a small constant (in all our experiments we set it to 5%). This criteria allows boundary fragments to have some amount of weak edge pixels, *i.e.* those having contrast less than σ.

Regions at all photometric scales are obtained by starting from the lowest level photometric scale and removing boundary fragments having contrasts less than σ, progressively as σ increases. This process, which is an agglomerative clustering of regions according to the inequality above, ensures that remaining regions always conform with the region model and successive merges create a strict hierarchy.

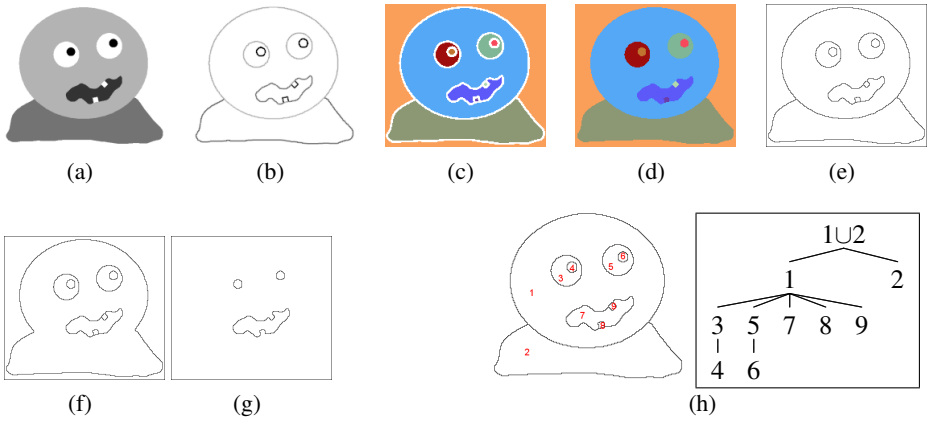
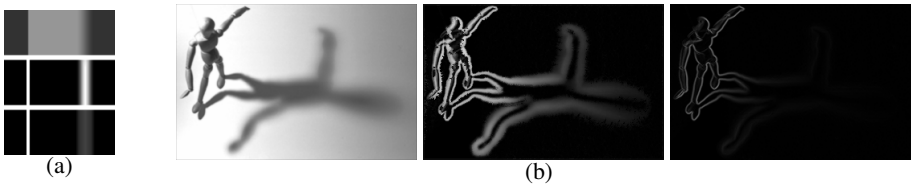


Fig. 3. Illustration of steps in the algorithm. (a) Input image I. (b) Output of ramp transform, C, applied to I. Here, the darker the pixel, the higher the contrast of the underlying ramp. (c) Basins of C. Each basin is represented with a different color. These basins correspond to region seeds and the remaining pixels are ramp pixels (white color). (d) Final labeling obtained by growing the region seeds towards the ramp pixels using relaxation labeling. (e,f,g) Results of multiscale segmentation. (e) Segmentation result for photometric scale $\sigma = 5$. All regions are included. (f) Segmentation for $\sigma = 65$. Two regions (head and the body) merged. This means that the photometric scale of the boundary fragment in between the two merging regions is less than 65. (g) Segmentation for $\sigma = 80$. More regions have disappeared. The remaining regions are of photometric scale larger than $\sigma = 80$, ensured by the region model and the algorithm. (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.

2.5 Constructing the segmentation tree

Due to the nature of multiscale segmentation process described above, regions merge as the scale of analysis, σ, is increased. This allows us to arrange the regions into a tree data structure. 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 should be 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 illustrate all steps of the algorithm on a simple synthetic image, in Fig.3.

3 Experiments and results



(a) (b)

Fig. 4. Illustration of the ramp transform. (a) Top: A synthetic image containing a sharp step edge, on the left, and a wide ramp edge on the right, which was obtained by blurring a step edge by a Gaussian kernel of $\sigma = 4$. The contrasts of both edges are 100. Middle: Ramp transform of the synthetic image. For both edges the peak value of the response of the transform is 100, which is the contrast of the edges. Bottom: Gradient magnitude, obtained by horizontal and vertical $[-1\ +1]$ filters. The responses for two edges are not the same. (b) Ramp transform of a real image. Left: An image containing ramp discontinuities of varying widths. Middle: Ramp transform of the image. Right: Gradient magnitude of the image.

We first demonstrate the ramp transform. To show that it can correctly measure the contrast of the underlying ramp edge, we created a synthetic image containing two edges of the same intensity contrast (Fig.4(a)). One of the edges is a step-edge, and the other is a ramp edge. Although they have different widths, we expect that the ramp transform gives similar values for these two edges. Fig.4(a) illustrates this property.

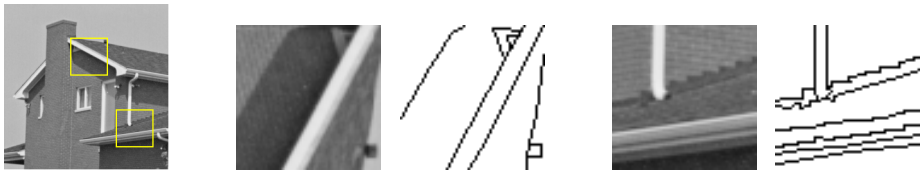
Fig.4(b) illustrates the ramp transform on a real image (taken from [5] with permission). This image contains ramp edges of various widths. A fixed-length gradient filter is incapable of measuring the ramp magnitudes correctly (see Fig.4(b), bottom). The ramp transform successfully estimates the pointwise ramp magnitudes as expected. Next, we describe how we quantitatively compared our segmentation algorithm with available algorithms.

3.1 Quantitative comparison with other algorithms

Creating a ground truth dataset for low-level image segmentation is a challenging task because 1) Segmenting images by hand is a laborious process. 2) Humans use their

high-level semantic knowledge while segmenting images, and it is difficult to eliminate this bias. This is why we do not use the Berkeley segmentation dataset [9], where human subjects were asked to segment out objects (see paragraph 4 in Section 1). Our goal in this study is not object-level segmentation, instead we want to detect all regions as accurately and completely as possible.

The dataset. We address the challenges stated above by having human subjects segment small image patches instead of whole images. This makes segmentation-by-hand a much easier process and removes the high-level knowledge bias to a large extent because a small image patch is unlikely to contain objects. We also randomly rotate the image patch (at multiples of 90 degrees) to further reduce this bias. We developed a GUI which, given an image, displays a random patch and allows the user to segment it by drawing polygonal lines.



(a) (b) (c)

Fig. 5. (a) An image from the 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.

We created such a dataset using a set of 15 images. We fixed the size of the patches at 50×50 pixels and randomly extracted 50 patches per image, thus obtaining 750 patches in total. Image sizes range approximately from 250×250 to 500×500 pixels. We used 5 human subjects to hand-segment these patches. Each patch was segmented by a single human subject. Since patch extraction was random, patches might overlap.

Fig.5 shows an image from our dataset and two example patches randomly extracted from it. Human segmentations for these patches are also given.

Performance measure. We evaluate the performance of a segmentation algorithm by looking at its segmentation accuracy over the patches where ground-truth segmentations are provided by human subjects. The segmentation accuracy for a single patch is measured by precision-recall values obtained by comparing the ground-truth and machine's output. We describe this process with an example. Consider the image in Fig. 6(a) and its patch in Fig. 6(b) with its ground-truth in Fig. 6(c) (call this as G). Suppose that our segmentation algorithm produced the result given in Fig. 6(d), thus the segmentation corresponding to the patch is Fig. 6(f) (let this be R). Now, we compare G and R and find correspondences between the boundary pixels in R and G. Each boundary pixel in G either matches with a single boundary pixel in R, or does not match with anything at all. To find the optimal matching between G and R, we use the method described in Appendix B of [10], which casts the problem as a minimum cost bipartite assignment problem. The result of matching is given in Fig. 6(g) where red pixels denote the

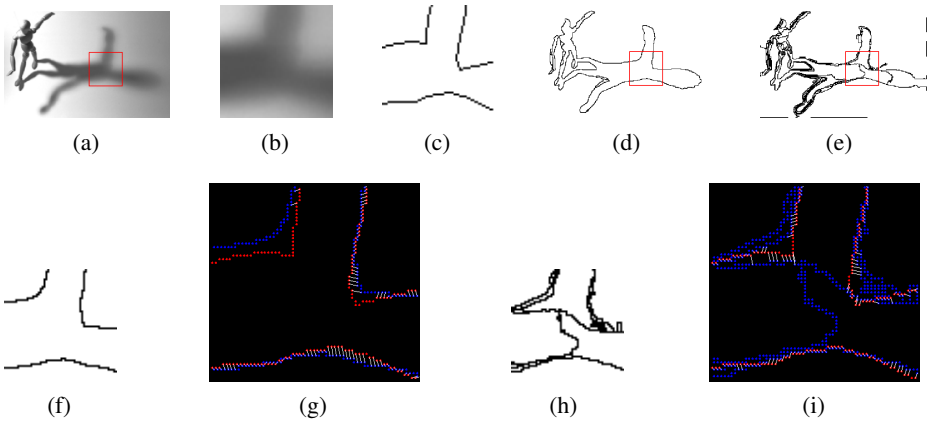


Fig. 6. Illustration of the performance measure. **(a)** An image from our dataset. A patch is marked by the red square. **(b)** Magnified version of the patch. **(c)** Provided human segmentation for the patch. **(d)** Segmentation of (a) obtained by our algorithm. **(e)** Segmentation of (a) obtained by the mean-shift based algorithm. **(f)** Our result at the location of the patch (red square in (d)). **(g)** Matching result between the ground-truth (c) and our result (f). Red-pixels represent the ground-truth, blue pixel represent algorithm's output (our result, in this case). White lines denote matching pixels. **(h)** Mean-shift based method's result at the location of the patch (red square in (e)). **(i)** Matching result between (c) and (h). See (g) for explanation.

boundary pixels of G, and blue pixels denote the boundary pixels of R. The white line between a red pixel and a blue pixel indicates a match.

As done in [10], we measure the *goodness* of the match by precision-recall. If the image patch is considered as a query, G as its relevant (ground-truth) result, and R as the retrieved result (by the algorithm), we could compute precision-recall as: $\text{Recall}\ \ r = \frac{|\text{relevant} \cap \text{retrieved}|}{|\text{relevant}|}$, and $\text{Precision}\ \ p = \frac{|\text{relevant} \cap \text{retrieved}|}{|\text{retrieved}|}$. We combine precision and recall using the F-measure defined as: $f = \frac{2pr}{p+r}$. For the matching result of Fig. 6(g), precision-recall and F-measure values are $p = 0.61$, $r = 0.66$, $f = 0.63$, whereas for Fig. 6(i) they are $p = 0.31$, $r = 0.95$, $f = 0.47$. Note that for the latter case, the recall is very high since only a few red pixels are unmatched, but the precision is low because there are plenty of blue pixels that are unmatched.

Comparison. We compared our algorithm with two available algorithms which are widely used in the literature: Felzenszwalb's graph-based algorithm [6] and the mean-shift algorithm [3]. We did not include N-Cuts [13] because it needs the number of regions as input, which is an explicit image-dependent parameter, and unknown apriori.

Both algorithms requires 3 input parameters from the user and we do not know which values to use. So, we sample a large number of input parameters for both algorithms. Mean-shift based segmentation method [3] requires a spatial bandwidth σ_s, a range bandwidth σ_r, and a minimum region area a. We selected 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 [6] expects a smoothing scale σ, a threshold k which is the scale of observation (equation (5) in [6]), and a minimum region size

a, as input. We used the following parameter space: $\{\sigma, k, a\} \in \{0.5, 1, 1.5, 2\} \times \{250, 500, 750, 1000, 1250, 1500, 1750, 2000, 2250, 2500\} \times \{5, 10\}$. Our algorithm outputs a hierarchy of regions. However, in order to compute its segmentation accuracy and compare it with other algorithms, we need single-layer segmentation results. For this purpose we used the photometric segmentation outputs at scales $\sigma = \{10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38\}$.

First, we report best average F-measure results per image (BAFPI). To get this for an image, we compute F-measures for all patches of that image and we average these results for each element in the input parameter space. Formally, BAFPI for image I and algorithm A is $\mathrm{BAFPI(I, A)} = \max_{\boldsymbol{p} \in S} \frac{1}{50} \sum_{i=1}^{50} F(I_i, A(I, \boldsymbol{p})_i)$ where S is the input parameter space of A, I_i is the ground-truth for the i^{th} patch of I, $A(I, \boldsymbol{p})$ is the segmentation result of A applied on I with parameters $\boldsymbol{p}$, $A(I, \boldsymbol{p})_i$ is the i^{th} patch of the segmentation result, and $F(ptch_1, ptch_2)$ gives the F-measure between the ground-truth $ptch_1$ and machine output $ptch_2$. BAFPI results are given in Table 1. Our algorithm outperforms the other two algorithms on all except three images.

Table 1. Best average F-measure per image (BAFPI) results.

Images →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Avg.
Graph-based	0.47	0.68	0.55	0.55	0.56	0.67	0.66	0.51	0.72	0.54	0.51	0.67	0.63	0.67	0.68	0.60
Mean-shift	0.62	0.80	0.60	0.58	0.70	0.78	0.79	0.63	0.72	**0.67**	0.65	**0.69**	0.77	0.76	**0.72**	0.70
Our method	**0.74**	**0.86**	**0.69**	**0.67**	**0.75**	**0.81**	**0.82**	**0.68**	**0.79**	0.65	**0.68**	0.65	**0.81**	**0.80**	0.71	**0.74**

The results in Table 1 are useful to show us the best these algorithms could do per image. On the other hand, these results are unrealistic because for each image, it assumes that we know the best input parameters to use, which is not true in practice. Therefore, we next look at what happens if we use the same input parameters for all images. We report the best average F-measures (BAF) per algorithm (along with the corresponding precision and recall values) in Table 2. BAF is defined as $\mathrm{BAF(A)} = \max_{\boldsymbol{p} \in S} \frac{1}{15\mathrm{x}50} \sum_{j=1}^{15} \sum_{i=1}^{50} F(I_i^j, A(I^j, \boldsymbol{p})_i)$ where I^j denotes the j^{th} image in the dataset.

Results in Table 2 show that our algorithm outperforms the others even when the input parameter is fixed and the same for all images. In fact, the discrepancy between the performances of our method and the mean-shift's is larger in BAF results (0.70 to 0.62) than the BAFPI results (0.74 to 0.70), which suggests that mean-shift's input parameters are more image-dependent.

Table 2. Best average F-measure (BAF) results.

	Precision	Recall	F-measure (BAF)
Graph-based	0.56	0.81	0.57
Mean-shift	0.61	0.87	0.62
Our method	**0.67**	**0.87**	**0.70**

Finally, we give some of the segmentation results (in Fig.7) obtained by the best parameters found by the BAF measure. (The parameters are: graph-based algorithm: $\sigma = 0.5, k = 1000, a = 10$, mean-shift: $\sigma_s = 5, \sigma_r = 6, a = 10$, our method $\sigma = 18$).

Fig. 7. Segmentation results. First column: input image, second column: output of graph-based method, third column: output of mean-shift based method, fourth column: our method. See text for details on input parameters.

In the supplementary material of the paper, we provide a graphical user interface to browse the segmentation trees of images. Finally, we want to note that our algorithm is not designed for texture segmentation. If run on texture images, it would only segment the locally homogeneous regions, corresponding to various levels of detail of the texture.

4 Conclusions

We presented a new algorithm for low-level multiscale image segmentation. The algorithm is capable of detecting low-level image structures having arbitrary shapes, at

arbitrary homogeneity levels. 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. We name this conversion as the ramp transform. Then we find the basins of the ramp magnitude map, and consequently obtain region seeds. These seeds are grown by a relaxation labeling 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 current scale are removed. This process guarantees a strict hierarchy. Using this property, we arrange the regions into a tree data structure. For empirical study, we created a new low-level segmentation dataset and showed that our algorithm outperforms two widely used segmentation algorithms.

Acknowledgments The support of the Office of Naval Research under grant N00014-09-1-0017 and the National Science Foundation under grant IIS 08-12188 is gratefully acknowledged.

References

1. N. Ahuja. A transform for multiscale image segmentation by integrated edge and region detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(12):1211–1235, 1996. 2
2. H. Arora and N. Ahuja. Analysis of ramp discontinuity model for multiscale image segmentation. In *ICPR '06: Int'l Conf. on Pattern Recog.*, pages 99–103, 2006. 2
3. D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619, 2002. 2, 9
4. D. Comaniciu, V. Ramesh, and P. Meer. The variable bandwidth mean shift and data-driven scale selection. In *ICCV*, pages 438–445, 2001. 2
5. J. H. Elder and S. W. Zucker. Local scale control for edge detection and blur estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 20(7):699–716, 1998. 7
6. P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. J. Comput. Vision*, 59(2):167–181, 2004. 9
7. S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 6(11):721–741, 1984. 1
8. R. M. Haralick and L. G. Shapiro. Survey- image segmentation techniques. *Computer Vision Graphics and Image Processing*, 29:100–132, 1985. 1
9. 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 *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001. 2, 8
10. D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(5):530–549, 2004. 8, 9
11. F. Meyer and S. Beucher. Morphological segmentation. *J. Vis. Comm. Image Represent.*, pages 21–46, 1990. 1
12. A. Rosenfeld, R. A. Hummel, and S. W. Zucker. Scene labeling by relaxation operations. *Systems, Man and Cybernetics, IEEE Transactions on*, 6(6):420–433, 1976. 4
13. J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000. 2, 9
14. C. Xu and J. L. Prince. Snakes, shapes, and gradient vector flow. *IEEE Transactions on Image Processing*, 7(3):359–369, 1998. 1
15. R. Zabih and V. Kolmogorov. Spatially coherent clustering using graph cuts. In *CVPR '04: Int'l Conf. on Computer Vision and Pattern Recog.*, pages 437–444, 2004. 2