Erschienen in: Image Analysis and Processing – ICIAP 2013 / Petrosino, Alfredo (Hrsg.). - Berlin, Heidelberg : Springer Berlin Heidelberg, 2013. - (Lecture Notes in Computer Science ; 8156). - S. 131-140. - ISBN 978-3-642-41180-9
https://dx.doi.org/10.1007/978-3-642-41181-6_14

Learning Precise Local Boundaries in Images from Human Tracings

Martin Horn and Michael R. Berthold

Nycomed Chair for Bioinformatics and Information Mining AND
Konstanz Research School Chemical Biology
University of Konstanz, Box 712, 78457 Konstanz, Germany
{martin.horn,michael.berthold}@uni-konstanz.de

Abstract. Boundaries are the key cue to differentiate objects from each other and the background. However whether boundaries can be regarded as such cannot be determined generally as this highly depends on specific questions that need to be answered. As humans are best able to answer these questions and provide the required knowledge, it is often necessary to learn task-specific boundary properties from user-provided examples. However, current approaches to learning boundaries from examples completely ignore the inherent inaccuracy of human boundary tracings and, hence, derive an imprecise boundary description. We therefore provide an alternative view on supervised boundary learning and propose an efficient and robust algorithm to derive a precise boundary model for boundary detection.

Keywords: boundary detection, supervised learning.

1 Introduction

A key cue in differentiating objects from each other and their background lies in the discovery of their boundaries. Hence, the detection of object boundaries is one of the most studied problems in computer vision and finds application in many different tasks like object detection/recognition, segmentation and tracking.

But characterizing boundaries can be quite a complex task and methods range from measuring abrupt changes in some low-level image features such as brightness or color (e.g. detected by looking for positions with high derivative) to considering texture gradients or other properties distinguishing the interior and exterior of a region (e.g. detected by comparing the distribution of some property of two halves of a disc [1]). Even more difficult, a high response of an unsupervised boundary detector alone does not tell us much in many real-world applications as high boundary strength and high boundary importance are considerably different concepts [2].

A popular way to deal with these difficulties is to combine different boundary cues by learning from user provided examples. To apply supervised learning to boundary detection, a set of features (boundary cues) is commonly calculated for an image patch of a certain size and orientation and an image patch classifier is

trained by minimizing its pixel-level disagreement with the given human boundary tracing. Applied to new images it results in a boundary map estimating the probability of a pixel to be a boundary pixel.

2 Learning Boundaries - Previous Research

The methods for supervised boundary detection [1,3,4,5,6,7,8] mainly differ in the boundary features they calculate and classification models they use. Martin et al. [1] for instance proposed a small, carefully selected, hand-designed feature set and tried various classifiers. They conclude that the choice of the classifier is of minor importance. Dollar et al. [5] in turn learned boosted trees on a huge set of generic features. A recent and very successful approach is suggested by Ren and Bo [3] in which features are automatically learned from the data using sparse coding before being classified with a linear SVM. Most of the methods are ranked and compared by means of the popular Berkeley Segmentation Benchmark [9], a set of natural images with a human-marked groundtruth.

3 An Alternative View on Boundary Learning

Amongst others Martin et al. [1] diagnose that the boundary learning problem is characterized by a high degree of class overlap in any low-level feature if tackled as the straightforward minimization of the pixel-level disagreement with the human boundary tracings. This is mainly caused by the inherent and unavoidable inaccuracy of human tracings. Requiring the human to provide a perfect and consistent boundary tracing, even for a single region, is ambitious and not actually feasible. But due to the mere amount of available training samples given by the Berkeley Benchmark [9] the effect is negligible if evaluated on that basis, and has therefore not been considered problematic so far.

But if boundaries are learned from a sufficient *small* set of training samples, the problem gets severe and unacceptable for many tasks and has at least two serious implications:

First, the boundaries detected are become broader, less precise and locally ambiguous the more inaccuracy the user introduces. But respecting the fact that a boundary in the continuous space is actually infinitely thin, a detected boundary in digital images should preferably not exceed the width of one pixel. This problem is illustrated in Fig. 1.

The second implication, closely related to the first, is that the complexity of the model learned (complexity for instance in terms of the *minimum description length*) increases with the users inaccuracy. And it is common sense that less complex models or hypotheses should be preferred (*Occam's razor*).

Consequently, a human tracing cannot be considered to cover the true desired boundary and only roughly indicates that the true boundary is located somewhere near the boundary labeling. Instead of receiving a set of instances that are labeled either positive (boundary) or negative (non-boundary), the learner should receives a set of *bags* containing *potentially* positive instances. During the

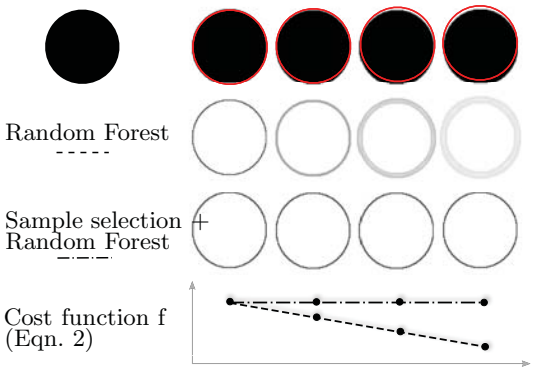

Fig. 1. Illustration of the problem when a boundary is learned by minimizing the pixel-level disagreement with the human boundary tracings. First row: An artificial image object and a labeling, which is increasingly disturbed (i.e. moved upwards). For the naive creation of the boundary model, the pixels (or rather the associated feature vector) underneath the red labels are the positive training samples, the negatives are randomly selected. Second row: If classifiers are naively trained on the training set, they will only perform well, if the label perfectly covers the right boundary positions. If not, the detected boundary will be broader and less precise. Third row: This is the boundary model that was learned by the alternative view on the supervised learning problem, by using the method proposed. Last row: The cost function indicating the homogeneity of the positive training samples used for classification.

inference of the model a maximum of one of these is selected. The non-boundary instances remain as single instances.

In the remainder of this section we will formally define the learning problem (Sect. 3.1) and present an approach to tackle it efficiently (Sect. 3.2).

3.1 Problem Definition

In order to learn boundaries from images a local boundary description (i.e. a feature vector for each pixel for a certain angle) needs to be derived and a set of closed contours (indicating where the boundaries to model are located, i.e. the human tracing) is required. Formally we define an oriented feature image and a closed contour as:

Definition 1 (Oriented Feature Image). *Let $\Omega \subset \mathbb{R}^2$ be the image domain (e.g. a pixel grid). Then the function I (oriented feature image) assigns to each pixel position (x, y) in the image and an angle α a feature vector (sample) $\boldsymbol{s} \in \mathbb{R}^d$, i.e. $I : \Omega \times [0, 2\pi] \mapsto \mathbb{R}^d$.*

Definition 2 (Closed Contour). *A contour $\gamma : [0, 1] \mapsto \mathbb{R}^2$ is a differentiable function from the closed interval $[0, 1]$ to the plane, also known as Jordan Curves. A contour is closed, if $\gamma(0) = \gamma(1)$.*

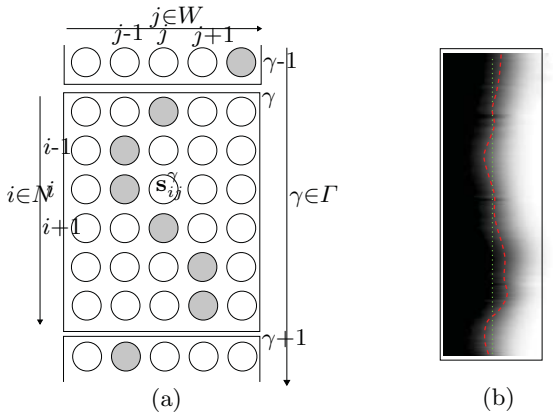

Fig. 2. (a) The grid, constructed curve-linear to the human-provided contours, with an exemplary boundary sample selection (shaded) which satisfies the constraints given in (3); (b) The grid curve-linear to the human trace (here as green dotted line) and a constraint-satisfying optimal solution (red dashed line).

Now let $\Omega \subset \mathbb{R}^2$ again be the image domain we are working on and Γ a set of closed contours (the given human tracings). Furthermore assume a feature image I according to Definition 1. Please note that for notational convenience the contour set as well as the feature image are continuous functions.

We now construct a finite set of feature vectors (samples) S by sampling along the contours in Γ as follows: Let $N = \{1, ..., n\}$ be the sampling points along a contour (n maybe chosen for each contour individually according to its length $l(\gamma) = \int_0^1 \|\gamma'(t)\| dt$, but w.l.o.g we choose n equal for each contour) and $W = \{0, ..., 2w\}$ the sampling points along the normal vectors of a contour with w being the number of points in one direction (width). Then we define a sampling function $s : N \times W \times \Gamma \mapsto \mathbb{R}^d$ for all contours

$$s(i, j, \gamma) = I(\gamma(i/n) + (j - w) * \gamma'(i/n), \measuredangle\gamma'(i/n)) \tag{1}$$

where $\gamma'(i)$ denotes the normal vector of γ at position i and $\measuredangle\gamma'(i)$ its angle. For notational convenience we will use $\boldsymbol{s}_{ij}$ instead of $s(i, j, \gamma)$ in the remainder and, therewith, ignore the presence of multiple contours. The extension of the process to multiple contours is straightforward.

Essentially we sampled within the feature image I along stripes, curve-linear to the given contour, and have constructed a grid with n rows and $(2w + 1)$ columns consisting of $n(2w + 1)$ feature vectors/samples. S is the union of all samples, i.e. $S = \{\boldsymbol{s}_{ij} | \forall i \in N; \forall j \in W\}$. See Fig. 2 for illustration.

Now, we want to determine a subset $\hat{S} \subset S$ such that (i) in almost each row exactly one sample is selected (*thinness*); (ii) the selected samples in successive rows are close together with respect to their column coordinates, if they belong to the same closed contour (*continuity*); (iii) the selected samples are located around the middle column, where the true contour is expected to be, i.e. their column positions average to 0 (*centrality*); and (iv) the selected samples

form dense clusters in the feature space $\mathbb{R}^d$ across all contours, i.e. maximizing $\sum_{\boldsymbol{s},\boldsymbol{s}'\in S'} \exp\left(-\frac{\|\boldsymbol{s}-\boldsymbol{s}'\|^2}{2\theta^2}\right)$ (*homogeneity*), following the intuition that positions *on* the contour should look similar. A fifth condition, enforcing a large difference between samples of the *same* row (related to feature selection), could possibly be regarded as well, but is out of the scope of this paper and subject to future work.

More formally, it is a constraint discrete optimization problem (also known as a general nonlinear integer programming problem) which is generally defined as the maximization of a function $f(\mathbf{x})$ (soft constraint) subject to certain (hard) equality $h_i(\mathbf{x}) = 0$ and inequality constraints $g_i(\mathbf{x}) \leq 0$.

For the problem formulated above we want to maximize

$$f(\mathbf{x}) = \sum_{i\in N} \sum_{i'\in N} \exp\left(-\frac{\|\boldsymbol{s}_{ix_i} - \boldsymbol{s}_{i'x_{i'}}\|^2}{2\theta^2}\right) \tag{2}$$

subject to the constraints

$$\begin{aligned} h(\mathbf{x}) &= \frac{1}{n}\sum_{i\in N} x_i \;\; = 0 \\ g(\mathbf{x}) &= \max_{i\in N} |x_i - x_{i'}| \leq 1 \\ &\quad \text{with } i' = (i+1) \mod n \end{aligned} \tag{3}$$

whereas $\mathbf{x} = [x_1, ..., x_n] \in W^n$ represents a certain solution (samples selection) of the optimization problem such that each entry of the vector $\mathbf{x}$ is the column index in the grid for the respective row within a contour. Hence, the actual sample set to be determined is then $\hat{S}_{\mathbf{x}} = \{\boldsymbol{s}_{ix_i} | \forall i \in N\}$. Please note that if multiple contours are considered, the cost function f has to be evaluated across all contours whereas the constraints (at least g) need to be satisfied within each contour individually.

The resulting sample subset $\hat{S}_{\mathbf{x}}$ serves as a positive training set and is used in conjunction with a negative training set (e.g. $S \backslash \hat{S}_{\mathbf{x}}$) to train an arbitrary classifier $c : \mathbb{R}^d \mapsto [0, 1]$ (e.g. random forest). The boundary detector, assigning each pixel a likelihood being a boundary pixel, is then defined as $P_b(x, y, \alpha) = c(I(x, y, \alpha))$.

As the number of feasible solutions is exponential $O(x^n)$ in the number of rows n, i.e. the total "length" of the given contour, it is almost impossible to enumerate all possible solutions to find the best one within a reasonable time (exhaustive search). Already the exact evaluation of the cost function (2) has a complexity of $O(n^2)$. Hence, approximations are inevitable. In the next section we will propose an algorithm which allows us to determine an approximate solution deterministically and very efficiently.

3.2 Boundary Sample Selection

Let Q_{ij} be the likelihood of the respective sample $\boldsymbol{s}_{ij}$ being selected as a boundary sample. Initially we consider all samples in the middle of the grid (the original

human selection) as most likely to be selected, hence, all $Q_{iw} = 1$ and the rest is set to 0. Then we iteratively update each $\tilde{Q}_{ij}$ with

$$\tilde{Q}_{ij} = \sum_{i'j'} \exp\left(\frac{-\|\boldsymbol{s}_{ij} - \boldsymbol{s}_{i'j'}\|^2}{2\theta^2}\right) Q_{i'j'} \quad . \tag{4}$$

That is, the samples with a likelihood of nonzero spread their values to nearby samples in the feature space and samples originally deemed as unlikely are increasingly likely to be selected if they had similar samples in their neighborhood, which, in turn, were defined as having high likelihoods in the previous iteration. After this for each row the sample with the maximum likelihood is determined and reset to 1, the rest of the same row again to 0. This process is repeated till convergence, i.e. the most likely sample for each row stops changing (see Algorithm 1).

Algorithm 1. *Basic* Sample Selection Algorithm

1: $Q_{ij} = \begin{cases} 1 \text{ if } j = w \\ 0 \text{ else} \end{cases}$
2: **while** not converged **do**
3: $\quad \tilde{Q}_{ij} \leftarrow \sum_{i'j'} \exp(\frac{-\|\boldsymbol{s}_{ij} - \boldsymbol{s}_{i'j'}\|^2}{2\theta^2}) Q_{i'j'}$
4: $\quad \hat{Q}_{ij} \leftarrow \begin{cases} 1 \text{ if } j = \operatorname{argmax}_{j'} \tilde{Q}_{ij'} \\ 0 \text{ else} \end{cases}$
5: **end while**

Algorithm 2. *Extended* Sample Selection Algorithm

1: $Q_{ij} = \begin{cases} 1 \text{ if } j = w \\ 0 \text{ else} \end{cases}$
2: $k(\boldsymbol{s}) = \sum_{\boldsymbol{s}' \in F} \exp(-\frac{\|\boldsymbol{s} - \boldsymbol{s}'\|^2}{2\theta^2})$
3: **while** not converged / not oscillating **do**
4: $\quad \tilde{Q}_{ij} \leftarrow (\sum_{i'j'} \exp(\frac{-\|\boldsymbol{s}_{ij} - \boldsymbol{s}_{i'j'}\|^2}{2\theta^2}) Q_{i'j'}) / k(\boldsymbol{s}_{ij})$
5: $\quad \hat{Q}_{ij} \leftarrow \begin{cases} 1 \text{ if } j = \operatorname{argmax}_{j'} \tilde{Q}_{ij'} \\ 0 \text{ else} \end{cases}$
6: $\quad$ Ensure $h(\mathbf{x}) = 0$ and $g(\mathbf{x}) \leq 1$ (3) with $x_i = \operatorname{argmax}_{j'} \tilde{Q}_{ij'}$
7: **end while**

If we consider

$$f = \sum_{ij} \sum_{i'j'} \exp\left(\frac{-\|\boldsymbol{s}_{ij} - \boldsymbol{s}_{i'j'}\|^2}{2\theta^2}\right) Q_{ij} Q_{i'j'} \tag{5}$$

then it is apparent that $f^1 \leq f^2 \leq ...$ holds true after each iteration $1, 2, ...$. Hence, it is converging towards a local maximum of the same function as given in (2).

In fact, the final solution we derive in this manner does not necessarily comply with the constraints (3) defining a feasible solution. Hence, after each iteration it might be necessary to alter the likelihoods of some dedicated samples such that the constraints are not violated anymore (e.g. by identifying the most probable line via dynamic programming). But changing some Q's might result in a temporary decrease of f (as the row-wise maxima are no longer considered for each row) and one has to be aware of that this can possibly lead to an oscillation

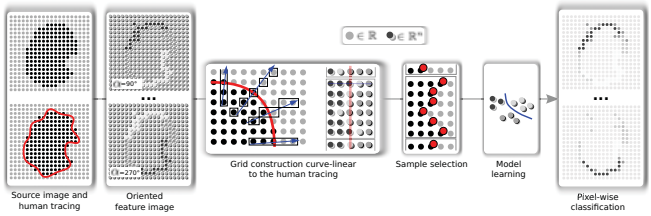

Fig. 3. The whole proposed process of boundary learning and detection: A feature image is derived from the source image by calculating boundary properties (e.g. gradients) for local image patches for different orientations. A grid, curve-linear to the (inconsistent) human tracing, is created (see Sect. 3.1 and Fig. 2) where each grid point is a feature vector. With the approach proposed in Sect. 3.2 the "true" boundary pixels can be identified efficiently. Taking these as positive training samples for a conventional classifier (e.g. random forest), a robust boundary model is derived and used to detect boundaries via pixel-wise classification.

rather than a convergence. Alternatively one can also disregard those samples that break ranks and consider only the constraint-satisfying samples in the next iteration or generally ignore the rows lacking a certain heterogeneity (which are likely to be non-boundary positions) completely.

Moreover, we observed that if true background samples accidentally have high likelihoods (e.g. due to the initial assignment) they negatively affect the convergence behavior and lead to undesired results (i.e. other background samples becoming successively most probable). That is due to the fact that background samples are usually more frequent and are therefore more likely to have more samples in the local neighborhood that contribute to the sum. But assuming that boundary samples are less frequent than background samples we simply convert the weighted sum into a weighted average, i.e. each sum is additionally normalized by the sample density. Very rare background samples are not selected either, as it is highly likely that they will not be similar to already-selected boundary samples. This little modification leads to much more robust results.

These insights are incorporated into the extended version of the proposed algorithm listed in 2.

3.3 Implementation

Both the maxima determination (line 5 in Algorithm 2) and assurance of the grid position constraints (line 6) run in linear time. The bottleneck is the update of the likelihoods (line 4). For each position in the grid the likelihoods have to be updated requiring the evaluation of the sum over all other variables and the estimation of the local density for normalization. A naive implementation would yield a quadratic complexity in the number of all samples, i.e. $O((2wn)^2)$, for each iteration. But looking more closely at the update equation in line 4 of the

Algorithm 2, one may recognize that this is actually a so-called Gauss transform, which is generally defined as

$$\hat{\boldsymbol{v_i}} = \sum_j \exp\left(\frac{-|\boldsymbol{p}_i - \boldsymbol{p}_j|^2}{2}\right) \boldsymbol{v}_j \ . \tag{6}$$

Various approaches have been proposed to overcome the computational complexity of Gauss transforms by approximate algorithms (e.g. [10]). They use the fact that the vector distances are faded out by a Gaussian kernel and only the distances of spatially close vectors have to be evaluated (those with big distances contribute very little to the overall sum and, hence, can be disregarded). This can be achieved by embedding the feature space into a special downsampled space (e.g. a Permutohedral lattice [10]) and reduces the complexity of the update step from quadratic to linear(!). Moreover, if we use a homogeneous vector representation the weighted sum in (6) can also perform a weighted average as required in Algorithm 2 (line 4). Then the input value is $[x, 1]$ and the output value of $[x, y]$ should be understood as $[\frac{x}{y}]$.

4 Results and Discussion

In our experiments we used two different feature sets to characterize boundary pixels. To describe simple edges (step or delta edges) we convolved the source images with various Gabor filters of different scales, each filter response representing an individual feature. To approximately capture the objects' interior and exterior we additionally regarded each pixel intensity along a line of certain length for different orientations as a feature (i.e. a line profile). These simplistic features have shown to be very powerful. None of the features we used measure more complex properties as those computational expensive texture features suggested by Martin et al. [1] or the learned feature set based on sparse codes introduced by Ren and Bo [3], because the aim is to understand and demonstrate the methodical need and success of the proposed boundary learning approach and not to drive an extensive evaluation of different hand designed feature sets, which, in fact, would have a strong impact on the results for a certain data set.

The examples in Fig. 4 compare the results obtained if the boundaries are learned naively or with the proposed boundary samples selection. The results clearly show that the alternative view on boundary learning significantly improves the outcome of the according boundary detector, i.e. the actual boundaries have almost the width of one pixel and they receive a significantly higher confidence than the background. A considerable increase of the cost function (2) and, hence, the altered curve-linear grid (Fig. 4 (d)) confirm this perception.

The generated angle-dependent probability map can subsequently be used to determine a segmentation by, for instance, performing an over-segmentation (e.g. with a watershed transform) followed by a region merging process, or, if the objects of interest are roundish (e.g. cells), by using dynamic programming applied on derived polar images at dedicated seeding points (see [11]).

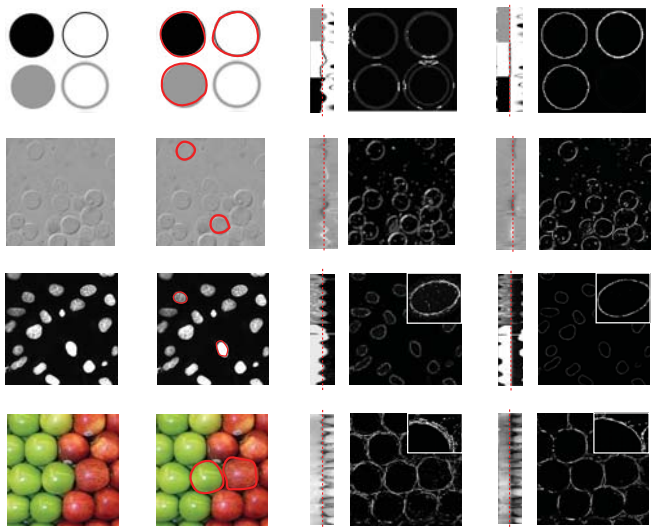

Fig. 4. Examples demonstrating the impact of the alternative boundary learning approach. (a) Source images: an artificial image, two cell images (taken from www.broadinstitute.org/bbbc) and a color image of apples; (b) Human selected examples; (c) The curve-linear grid without a correction via sample selection. The derived boundary detector is unsatisfactory; (d) The curve-linear grid corrected according to the selected instance identified with the proposed Algorithm 2. The boundary detection result is precise and as expected (the boundary orientation is known but not shown here).

These accurate segmentation results are, for instance, of major importance in an active segmentation framework where a segmentation is learned successively from as few examples as possible in a user-feedback-loop which is the ultimate goal of the work presented here.

5 Future Work

Essential for the suitability of the boundary model for a certain task is the choice of the right features calculated on the basis of the local pixel neighborhood. Although many combinations of available characteristics of a local image patch work quite well with our approach, it is still desirable to be able to determine the importance of individual features automatically for the supervised boundary learning problem at hand. It may, for instance, reduce the costs for the pixel-wise feature calculations and may lead to less complex boundary models.

Moreover, the choice of θ in (5) has a considerable impact on the quality of the results and an automatic determination would be desirable, possibly for subsection of the contours individually, which is essentially closely related to feature selection.

Finally, instead of selecting the samples after the boundary already has been traced by the human, a more consistent and homogeneous boundary might be suggested right during the actual manual delineation process.

6 Conclusions

We have proposed a simple and efficient algorithm to derive the "true" local boundary description indicated by an inconsistent and inaccurate human boundary tracing. The subsequently determined local boundaries via pixel-wise classification are invariant to disturbance and imprecision and examples have shown that the method significantly improves the accuracy and robustness of local boundary detection. Apart from the definition of the right local image patch features, which is subject to future work, the algorithm only requires one parameter θ (5).

References

1. Martin, D.R., Fowlkes, C.C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE PAMI 26(5), 530–549 (2004)
2. Köthe, U.: Reliable Low-Level Image Analysis. Professorial dissertation, Department Informatik. University of Hamburg (2008)
3. Ren, X., Liefeng, B.: Discriminatively trained sparse code gradients for contour detection. In: Bartlett, P., Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems 25, pp. 593–601 (2012)
4. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. Technical Report UCB/EECS-2010-17, EECS Department, University of California, Berkeley (February 2010)
5. Dollar, P., Tu, Z., Belongie, S.: Supervised learning of edges and object boundaries. In: CVPR, pp. 1964–1971 (2006)
6. Konishi, S., Yuille, A.L., Coughlan, J.M., Zhu, S.C.: Statistical edge detection: Learning and evaluating edge cues. IEEE PAMI 25, 57–74 (2003)
7. Meila, M., Shi, J.: Learning segmentation by random walks. In: Advances in Neural Information Processing Systems, pp. 873–879. MIT Press (2001)
8. Will, S., Hermes, L., Buhmann, J.M., Puzicha, J.: On learning texture edge detectors. In: Proc. Int. Conf. Image Processing, pp. 877–880 (2000)
9. Martin, D.R., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. Technical Report UCB/CSD-01-1133, EECS Department, University of California, Berkeley (Jan 2001)
10. Adams, A., Baek, J., Davis, M.A.: Fast high-dimensional filtering using the permutohedral lattice. Computer Graphics Forum 29(2), 753–762 (2010)
11. Horn, M., Berthold, M.: Towards active segmentation of cell images. In: Biomedical Imaging: From Nano to Macro, March 30 -April 2, pp. 177–181 (2011)