A Min-Cover Approach for Finding Salient Curves

Pedro Felzenszwalb
University of Chicago
pff@cs.uchicago.edu

David McAllester
TTI at Chicago
mcallester@tti-c.org

Abstract

We consider the problem of deriving a global interpretation of an image in terms of a small set of smooth curves. The problem is posed using a statistical model for images with multiple curves. Besides having important applications to edge detection and grouping the curve finding task is a special case of a more general problem, where we want to explain the whole image in terms of a small set of objects. We describe a novel approach for estimating the content of scenes with multiple objects using a min-cover framework that is simple and powerful. The min-cover problem is NP-hard but there is a good approximation algorithm that sequentially selects objects minimizing a "cost per pixel" measure. In the case of curve detection we use a type of best-first search to quickly find good curves for the covering algorithm. The method integrates image data over long curves without relying on binary feature detection. We have applied the curve detection method for finding object boundaries in natural scenes and measured its performance using the Berkeley segmentation dataset.[1]

1. Introduction

The problem of finding salient curves in grayscale or color images is an important question with applications to boundary detection, perceptual grouping and object recognition. This paper describes a new algorithm for estimating the set of curves in an image. The method is based on a statistical model for scenes with multiple curves and a novel approach for estimating the content of a scene with multiple objects.

We see the curve detection task as a instance of a more general problem, where we want to simultaneously detect multiple objects in one image. The general problem of finding an optimal interpretation of an image in terms of multiple objects poses an incredible algorithmic challenge. Our paper describes a novel approach for tackling this problem that is both simple and powerful.

We formulate multiple object detection as a weighted minimum-cover problem. The basic idea is that each object in the scene covers a part of the image. An interpretation of the scene is given by a set of objects which cover the whole image, where one of the objects is typically a generic background model. The minimum-cover problem is NP-hard but approximation algorithms exist. We apply the general framework for finding curves in images and give empirical results on the Berkeley segmentation dataset. We show that our curve finding method improves boundary detection results relative to the local filters described in [10] and the conditional random field model in [13]. [2]

One of our main goals is to reliably detect meaningful curves in natural images. Intuitively, a good curve is a path in the image which has a single underlying cause. For example, a curve can be caused by an occlusion (a discontinuity in depth), a sharp change in albedo (paint on a surface) or a sudden change in lighting (a shadow). We construct a simple model for scenes with multiple curves where the notion that each curve should have a single cause is represented implicitly in the statistical properties of the model. For example, we stipulate that curves tend to be smooth.

As mentioned above, we pose multiple object detection as a minimum-cover problem. In general each object in a covering has a cost which depends on the prior probability of seeing that object (some objects are more common than others) and the probability of observing the part of the image covered by the object, assuming that the object is in the scene. We propose to use a classical greedy approximation algorithm for computing a good interpretation of an image. The algorithm sequentially selects objects so as to minimize the ratio of the object cost per area of the image it explains. This can be seen as a "cost per pixel" or cost density measure. As explained in Section 5, sequentially finding objects with minimum cost density can lead to significantly different results from sequentially finding the best single object interpretation of the remaining image.

One of our main contributions is an efficient method for

[1]This material is based upon work supported by the National Science Foundation under Grant No. 0534820 and 0535174.

[2]Our results are inferior to [13] on a horse dataset. An interpretation of the differences between the two datasets is discussed in Section 7.

implementing the inner loop of the minimum-cover algorithm in the case where the objects we are looking for are smooth curves. We use a type of best-first search to quickly find a curve with minimum cost density. Our method works by integrating image measurements over long curves without relying on intermediate decisions such as binary edge detection. This makes our curve finding system a practical alternative to classical methods that work by performing feature detection followed by a linking procedure.

Of course we are not the first to consider the problem of finding curves in images. Our model for images with multiple curves is a generalization of the model by Geman and Jedynak in [6]. They described a statistical model for scenes with one curve and a fast inference algorithm for the case where the starting point of the curve is known in advance. In contrast we consider a model for scenes with multiple curves without known starting points.

There has been considerable earlier work on using curvilinear continuity to compute a *saliency* measure for each pixel or edge fragment in an image [1, 7, 14, 15, 16, 17]. Typically this saliency measure is related to the likelihood that a curve in the scene goes through a particular location in the image. The methods in [8] and [9] look for multiple salient curves by sequentially selecting the most salient curve and taking it out of the image.

Our curve finding algorithm searches for optimal curves by starting from short curves and iteratively expanding curves that look promising. This is related to the methods in [6] and [4]. It is also related to parsing algorithms that use figures of merit to order computation [3].

2. Scene Model

We use a simple model for discrete curves where each curve is represented by a sequence of short oriented segments (similar to [15] and [6]). Let P be a set of points in the plane. We assume that there is a fixed number of possible segments connecting each point $p \in P$ to nearby points as illustrated in Figure 1. The set of segments coming out of a point p corresponds to different orientations that a curve can follow as it passes through the point. The set of all possible oriented segments is denoted by S. A curve is represented by a sequence of adjacent segments $(s_1, \ldots, s_n)$ as shown in Figure 2.

In the simplest case the set of points P correspond to pixels of the input image. However, we have found it useful to consider subpixel curves where the segments go between subpixel locations. In the general model we avoid specifying the precise nature of the sets P or S.

Let F_s denote the output of a local image filter associated with a segment $s \in S$. The filter response F_s gives a noisy indication as to whether or not s is part of a curve in the scene. For example, F_s could measure the image derivative perpendicular to s. Our algorithm looks for a

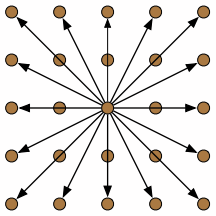

Figure 1. Example where there are 16 oriented segments leaving each point. The set S is the union of all oriented segments.

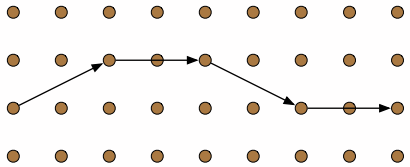

Figure 2. Curves are sequences of adjacent segments. The picture shows a curve formed by 4 segments.

global explanation of the filter responses in terms of a set of curves $X = \{C_1, \ldots, C_k\}$. Each curve explains the filter responses for the segments in it, while the filter responses that are not under a curve are explained by a background model. Each curve in the hypothesis X should be smooth and the number of curves should be small.

We use a statistical model for images with multiple curves which is a generalization of the model for images with one curve in [6]. We assume that each curve in the scene is drawn independently from a Markov process which favors smooth curves,

$$P(C) = \frac{1}{Z} \prod_{i=1}^{n-1} \Phi(s_i, s_{i+1}). \tag{1}$$

The number of curves in the image is chosen according to a distribution $P(k) \propto a^k$ with $0 < a < 1$ such that scenes tend to have few curves. This defines a prior distribution over scenes,

$$P(X) \propto \prod_{C \in X} aP(C). \tag{2}$$

We assume that the filter responses are conditionally independent given the true scene, and that the value of F_s depends only on whether or not a curve passes through s. Let $p_f(F_s)$ denote the distribution of filter responses for segments along a curve while $p_b(F_s)$ denotes the distribution of filter responses in the background. As long as the intersections among curves in the scene are small we can approximate the data model as,

$$P(F_s,\ s \in \mathcal{S} \,|\, X) = \prod_{C \in X} \prod_{s \in C} p_f(F_s) \prod_{s \in X_B} p_b(F_s), \tag{3}$$

where X_B denotes the segments that are not in any curve.

Given the output of the filter responses F_s, the most likely hypothesis for X can be expressed as,

$$X^* = \operatorname*{argmax}_X P(X)P(F_s,\ s \in S \mid X) \tag{4}$$

$$= \operatorname*{argmax}_X \prod_{C \in X} \left(aP(C) \prod_{s \in C} p_f(F_s) \right) \prod_{s \in X_B} p_b(F_s). \tag{5}$$

3. Reduction to Weighted Min-Cover

We can reformulate the optimization problem by taking the negative logarithm of equation (5). First define,

$$cost(C) = -\log \left(aP(C) \prod_{s \in C} p_f(F_s) \right) \tag{6}$$

$$cost(s) = -\log p_b(F_s) \tag{7}$$

Now the optimal X^* can be expressed as,

$$X^* = \operatorname*{argmin}_X \sum_{C \in X} \text{cost}(C) + \sum_{s \in X_B} \text{cost}(s). \tag{8}$$

This is an instance of the general weighted min-cover problem. In the general case we have a set of elements U and a set of objects O. Each object covers some subset of U and has a cost. The problem is to find a subset of O with minimum sum of costs such that every element of U is covered. In our case we can take U to be the segments S and take O to be the set of all possible curves plus the set of all possible "background segments". Each background segment is an object that covers a single segment $s \in S$ with the background model. Since the costs are non-negative a minimum cost cover will always contain a set of curves and a set of background segments that are not in any curve, corresponding to X and X_B.

4. Greedy Approximation Algorithm

Here we think of an object $o \in O$ as the subset of U that it covers. There is a simple approximation algorithm for weighted min-cover that is guaranteed to find a solution within a factor of $\log(\max_{o \in O} |o|)$ of the optimal cover. From a practical perspective this factor does not seem to be very good — perhaps around 5 in a realistic setting where curves can have up to 200 segments. However, approximation algorithms often perform significantly better than their worst case guarantees and the method seems to perform well for the curve finding problem.

The general algorithm builds a cover of U using objects in O starting from the empty cover and selecting objects one at a time. We denote by $o_i \in O$ the object selected in the i-th iteration. At each step the algorithm selects an object with minimum cost per number of newly covered elements,

$$o_{i+1} = \operatorname*{argmin}_{o \in O} \frac{\text{cost}(o)}{|o - (o_1 \cup \ldots \cup o_i)|}. \tag{9}$$

Note that the right hand side is a type of cost density measure. The algorithm keeps selecting objects until all of U is covered. While this greedy algorithm is extremely simple, for the case of finding curves there is an exponential number of possible objects to consider in each step. In Section 6 we describe an efficient algorithm for finding curves with optimal cost density that can be used in the "inner-loop" of the approximation algorithm.

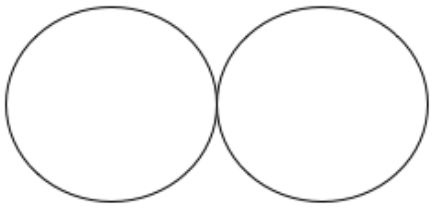

Figure 3. We can interpret this image as containing two zeros or an infinity sign. In the case of curves we can think of the image as containing two circles or a figure eight.

5. Min-Cover vs. Single Object Detection

The min-cover approximation algorithm finds one object at a time — the object that minimizes a cost density. It is natural to ask how this process differs from detecting one object at a time using a model for scenes with a single object. The difference is in the optimization criterion used. The min-cover approximation algorithm finds the object of minimum cost density ignoring the cost of parts of the image not covered by the object. Algorithms which assume that there is a single object in the image typically optimize the total cost of the object plus the background hypothesis. Examples include the method in [6] for curves and the method in [5] for deformable objects.

Optimizing the total cost (object plus background hypothesis) will tend to find larger objects than optimizing the object cost density. Consider the image in Figure 3. The image can be interpreted as having two zeros or one infinity sign. If we use an algorithm that assumes only one object is present then the infinity sign interpretation is strongly preferred because it explains more of the image with less cost than the background model. The method is biased because it assumes that any area of the image not explained by the object hypothesis will have to be explained as being part of the background. The min-cover greedy approximation algorithm will prefer the two zeros interpretation whenever one of the two zeros has lower cost density than the infinity

sign. Clearly neither algorithm is perfect. We would prefer an exact solution to the NP-hard min-cover problem. The problem of finding objects that are too small, or interpreting object fragments as whole objects is not so serious in curve finding because breaking a single curve into a few curves is not an important mistake for most applications. This can often be fixed by a postprocessing step.

6. Computing the Best Curve

In applying (9) to curve finding there are two cases to consider: o_{i+1} can be a curve or a background segment. It is easy to compute the best background segment. We focus here on computing the best curve. If the best background segment is better than the best curve, but it is not part of the best curve, then selecting the background segment does not change the best curve for the next iteration of the min-cover algorithm. This is because covering a segment can only increase the cost density of curves — if the best curve is not affected then it must remain optimal. Therefore, the best curve needs to be recomputed only when a curve is selected as the next object or when a background segment is selected that is part of the current best curve.

Consider the greedy selection defined by (9) where o_i is restricted to be a curve. For each segment $s \in S$ let $l(s) = 1$ if s has not been covered so far, while $l(s) = 0$ otherwise. Its useful to think of the length of C to be the number of uncovered segments in the curve,

$$l(C) = \sum_{i=1}^{n} l(s_i). \tag{10}$$

The greedy min-cover algorithm needs to repeatedly select a curve which minimizes $\mathrm{cost}(C)/l(C)$. To simplify notation we define the following quantities,

$$A = -\log a \tag{11}$$

$$w(s) = -\log p_f(F_s) \tag{12}$$

$$t(s,q) = -\log \Phi(s,q) \tag{13}$$

The constant A encourages the best curve to be long (larger values bias the model toward a smaller number of longer curves), $w(s)$ is the cost of explaining the filter response at s using the foreground model and $t(s,q)$ is a cost that encourages curves to be smooth. In practice we can define the constant A, the weights $w(s)$, and the transition costs $t(s,q)$ directly without reference to probabilities.

Now we express the cost of a curve in terms of a weight that depends on the data under C and the smoothness of C,

$$w(C) = \sum_{i=1}^{n} w(s_i) + \sum_{i=1}^{n-1} t(s_i, s_{i+1}) \tag{14}$$

$$\mathrm{cost}(C) = A + w(C) \tag{15}$$

One way to find a minimum density curve would be to fix a maximum length L and consider all curves of length up to L. We could use dynamic programming to compute the minimum cost curve of length 1 through L ending at each segment and then select the one with minimum cost density. Since the dynamic programming table has $|S|L$ entries this would be quite slow — in particular $|S|$ is on the order of the number of pixels in the image. Below we describe a simple algorithm which is able to find the optimal curve quickly by using a type of best-first search.

Intuitively the curve finding algorithm works as follows. We start by generating all curves of length one and inserting them in a queue. We repeatedly remove a curve from the queue and insert all possible one-segment extensions back in the queue. We stop when we can guarantee that no extension of a curve in the queue can have better cost density than one of the curves generated so far. Since every curve is an extension of its first segment, the algorithm is guaranteed to find the best one.

Note that if we ever have two curves C and D of the same length ending at the same segment with $w(C) \leq w(D)$ we can forget about D because for every extension of D there is an extension of C with the same length that is at least as light. The algorithm simply keeps track of the lightest curve of length l ending at segment s for each pair $(s,l) \in S \times \mathbb{N}$. This is done using two sparse tables W and T. The value $W[s,l]$ is the weight of the lightest curve ending at (s,l) generated so far, while $T[s,l]$ is the predecessor of s in a lightest curve ending at (s,l). At any point throughout the run of the algorithm a curve of weight $W[s,l]$ can be obtained by starting from the last segment s and tracing the previous segments using T.

Below we will show that if $\mathcal{P}$ is the prefix of an optimal curve $\mathcal{O}$, then the cost density of $\mathcal{O}$ is at least the weight density of $\mathcal{P}$,

$$\frac{\mathrm{cost}(\mathcal{O})}{l(\mathcal{O})} \geq \frac{w(\mathcal{P})}{l(\mathcal{P})}. \tag{16}$$

This means that if at some point the algorithm has generated a curve with cost density at most the weight density of all curves in the queue it must have already generated the optimal curve. Until this happens we have to keep extending curves in the queue. We always extend the curve in the queue with lowest weight density — those are the most promising curves. Note that the bound in equation (16) implies that every prefix of an optimal curve has low weight density. By expanding curves in weight density order the algorithm quickly generates the optimal curve and rules out promising prefixes.

Pseudocode for the curve finding algorithm is shown in Figure 4. We use hashtables to keep track of W and T and a priority queue to keep track of Q. The value of $W[s,l]$ is taken to be infinity until it is defined for the first time. Our current implementation starts from scratch every time

Algorithm *FindCurve*

1. **for** Each segment $s \in S$
2. $W[s, l(s)] \leftarrow w(s)$
3. Insert $(s, l(s))$ in Q
4. **repeat**
5. Remove item (s, l) from Q with minimum weight density $W[s, l]/l$
6. **for** Each segment q following s
7. $v \leftarrow W[s, l] + t(s, q) + w(q)$
8. $k \leftarrow l + l(q)$
9. **if** $v < W[q, k]$
10. $W[q, k] \leftarrow v$
11. $T[q, k] \leftarrow s$
12. Insert (q, k) in Q
13. **until** $\min_{(s,l)}(W[s, l] + A)/l \leq \min_{(s,l)\in Q} W[s, l]/l$

Figure 4. Pseudocode for the curve finding algorithm.

it needs to recompute the best curve but we believe a further speed up could be obtained by reusing work from one run of the algorithm in the next run. In practice it takes a fraction of a second to find the optimal curve in an image.

There is a simple intuition behind the bound in equation (16). If we look at a curve $\mathcal{O}$ with prefix $\mathcal{P}$ and suffix $\mathcal{S}$, the only way for $\mathcal{O}$ to have low cost density when $\mathcal{P}$ has high weight density would be for $\mathcal{S}$ to have low weight density and be long. But if $\mathcal{O}$ is optimal $\mathcal{S}$ cannot be better. Let $\mathcal{O}$ be an optimal curve. For every curve C we have,

$$\frac{\text{cost}(C)}{l(C)} \geq \frac{\text{cost}(\mathcal{O})}{l(\mathcal{O})}, \tag{17}$$

$$w(C) + A \geq l(C)\frac{\text{cost}(\mathcal{O})}{l(\mathcal{O})}. \tag{18}$$

Let $\mathcal{O}$ be an optimal curve with prefix $\mathcal{P}$ and suffix $\mathcal{S}$. We use $t(\mathcal{P}, \mathcal{S})$ to denote the cost of transitioning from the last segment of $\mathcal{P}$ to the first segment of $\mathcal{S}$,

$$\frac{\text{cost}(\mathcal{O})}{l(\mathcal{O})} = \frac{w(\mathcal{P}) + w(\mathcal{S}) + t(\mathcal{P}, \mathcal{S}) + A}{l(\mathcal{P}) + l(\mathcal{S})}. \tag{19}$$

Since $t(\mathcal{P}, \mathcal{S})$ is non-negative we can get an inequality,

$$(l(\mathcal{P}) + l(\mathcal{S}))\frac{\text{cost}(\mathcal{O})}{l(\mathcal{O})} \geq w(\mathcal{P}) + w(\mathcal{S}) + A. \tag{20}$$

Now we use equation (18) where C is taken to be $\mathcal{S}$,

$$(l(\mathcal{P}) + l(\mathcal{S}))\frac{\text{cost}(\mathcal{O})}{l(\mathcal{O})} \geq w(\mathcal{P}) + l(\mathcal{S})\frac{\text{cost}(\mathcal{O})}{l(\mathcal{O})}. \tag{21}$$

Canceling $l(\mathcal{S})\frac{\text{cost}(\mathcal{O})}{l(\mathcal{O})}$ from both sides gives us the bound expressed by equation (16).

7. Results

As described in the introduction one of our main goals is to detect the boundaries of objects in natural images. Figure 5 illustrates some results obtained using our curve finding method. These results were obtained using the "probability of boundary" (PB) measure from [10] to define the local responses F_s for each segment in the image. The PB measure assigns a number between zero and one to each image pixel, corresponding to a measure of edge strength. The value is based on a combination of both color and texture information around a pixel. In practice we let F_s be the average PB value along segment s, while $w(s) = -\log(F_s+1)$. The value of $cost(s)$ is set to a constant b. In this case the background model does not influence the selection of the optimal curves, it only defines a stopping criteria. The mincover algorithm keeps picking optimal curves until the cost density of the best curve is at least b. For the transition costs we let $t(s, q)$ be zero if the orientation of q is almost the same as the orientation of s and infinity otherwise.

The examples in Figure 5 demonstrate how we can get a good interpretation of an image in terms of just a few smooth curves. The output of our curve finding algorithm is significantly better than the output produced by local edge detection methods. In particular for the desert scene, local methods produce much more cluttered results.

We have also used our curve finding method to produce subpixel estimates for the curves in an image. In this case segments in S connect pairs of subpixel locations in neighboring pixels. For estimating subpixel curves we used the derivative of the image along a direction perpendicular to s as the local measure F_s. We let $w(s) = -\log(F_s+1)$ when $F_s > 0$ and zero otherwise. This makes the curve model have a notion of which side of the curve is brighter than the other. Figure 6 illustrates the results on a simple image. Our algorithm is able to find good subpixel estimates for the curves in the image by simultaneously selecting curves and regularizing their shapes. For this example we picked transition costs $t(s, q)$ so that curves can only turn in one direction, giving a stronger shape regularization constraint. The algorithm simultaneously searched for curves that turn right and curves that turn left.

Finally, we have used the Berkeley segmentation dataset [10] to evaluate the performance of our curve finding method on a wide variety of images. The dataset contains human-segmented natural images for testing boundary detection algorithms. Performance is quantified using a precision-recall framework. The output of a boundary detection algorithm is rated according to the fraction of the detected edges that are true boundaries (precision) and the fraction of true boundaries that are detected by the algorithm (recall). Figure 7 compares the results of our algorithm using PB for the local measurements with the performance of the CRF model from [13] and the performance

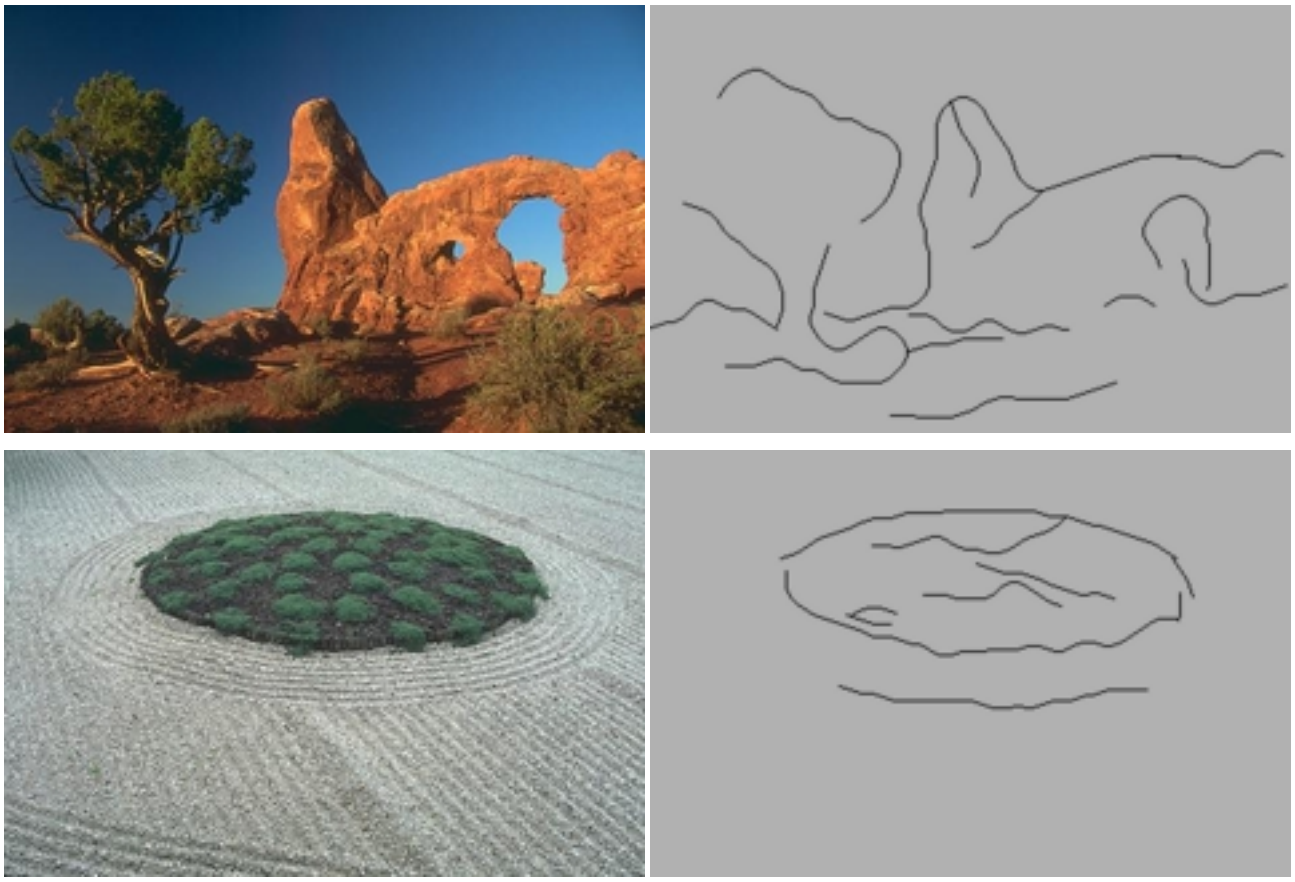

Figure 5. Example results of finding curves in natural images from the Berkeley segmentation dataset. Our algorithm finds good interpretations of these images using just a few smooth curves. Local edge detection methods tend to produce much more cluttered results.

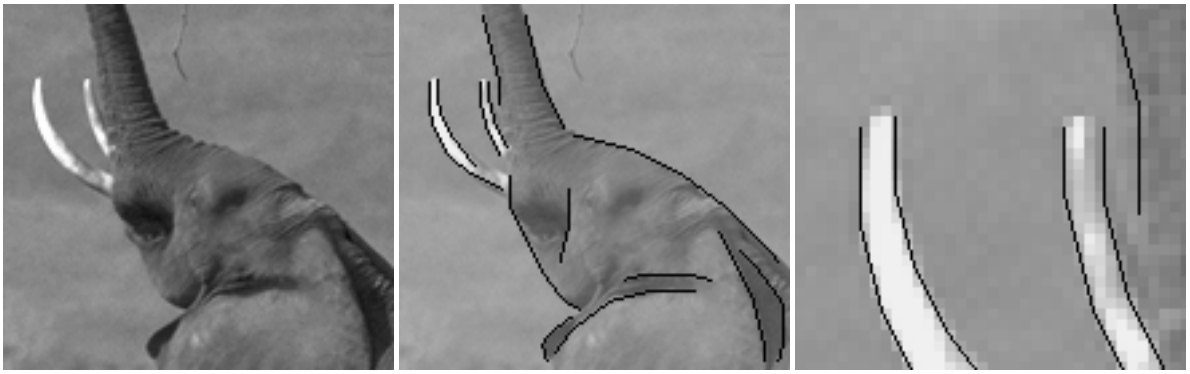

Figure 6. Subpixel curves found in the elephant image. The rightmost picture shows the subpixel estimates around the tusk area. Note how we get a good description of the image in terms of very few smooth curves.

obtained using PB alone.

In the Berkeley segmentation dataset our algorithm outperforms edge detection using PB alone and the CRF model in [13] in the low-recall/high-precision regime. All algorithms have similar performance in the high-recall regime. Note that the three algorithms perform better than classical edge detection methods. Although our results on this dataset are reasonably good, we do not feel that the metric used in this experiment is an appropriate measure of performance for our algorithm. The metric depends only on which pixels are marked as edges. Therefore it does not check that the edges were grouped into meaningful curves. Construction of an appropriate empirical metric for multiple curve detection remains an open problem.

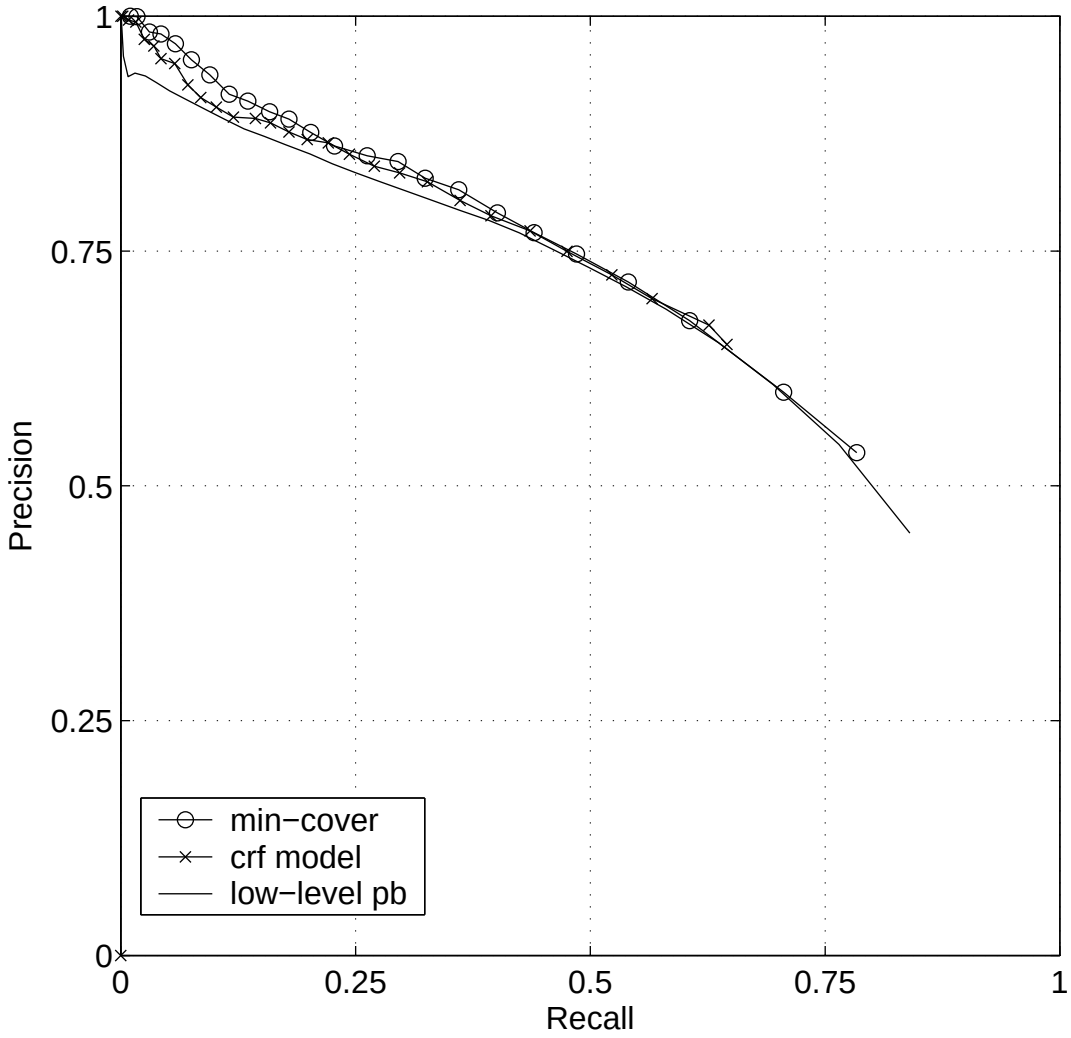

Figure 7. Berkeley segmentation dataset results. The graph compares our curve finding algorithm, the CRF model from [13] and the local detector from [10]. See text for a description of the experiment.

We have also tested our algorithm in the horse dataset from [2]. In this case the min-cover algorithm performs better than PB alone but not as well as the CRF method. The horse dataset poses a significantly different problem than the Berkeley segmentation dataset — the goal is to detect the boundaries of a horse in each image, and no other boundaries. Most of the images contain objects such as fences, posts and trees. The CRF method is able to suppress many non-horse boundaries because it takes into account relationships among curves. For example, it seems to suppress boundaries that end in a T-junction which is a good cue for figure/ground separation. Our algorithm assumes that the curves in an image are independent so it has no ability to reason about junctions.

In the future we will explore how to postprocess the output of our method to do figure/ground separation. One possibility would be to build a CRF model on top of the curves we find.

8. Conclusion

The minimum-cover framework gives a general, well-founded approach for interpreting images with multiple objects. In particular, the greedy approximation algorithm can be used in a wide variety of situations, including cases where there are different types of objects in an image. It provides an alternative to sequentially selecting the best single object interpretation of the image.

For the case of finding salient curves in images we introduced a new algorithm for finding curves with minimum cost density. Our method exploits the fact that curves can be generated by a sequential growth process that starts from

a relatively small number of basic tokens (the segments). We have shown how the quality of a curve can be used as an effective figure of merit to decide which curves to keep growing and when the process can stop. The resulting algorithm is able to quickly produce a good interpretation of an image in terms of a small set of curves.

References

[1] J. August and S. Zucker. Sketches with curvature: The curve indicator random field and markov processes. *PAMI*, 25(4):387–400, April 2003.

[2] E. Borenstein and S. Ullman. Class-specific, top-down segmentation. In *ECCV*, page II: 109, 2002.

[3] S. Caraballo and E. Charniak. Figures of merit for best-first probabilistic chart parsing. *Computational Linguistics*, 24:275–298, 1998.

[4] J. M. Coughlan and A. L. Yuille. Bayesian A* tree search with expected $o(n)$ convergence rates for road tracking. In *EMMCVPR*, 1999.

[5] D. Crandall, P. Felzenszwalb, and D. Huttenlocher. Spatial priors for part-based recognition using statistical models. In *CVPR*, pages I: 10–17, 2005.

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

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

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

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

[10] D. Martin, C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *PAMI*, 26(5):530–549, May 2004.

[11] D. Mumford. Elastica and computer vision. In *Algebraic Geometry and Its Applications*, pages 491–506. Springer-Verlag, 1994.

[12] P. Parent and S. Zucker. Trace inference, curvature consistency, and curve detection. *PAMI*, 11(8):823–839, August 1989.

[13] X. Ren, C. Fowlkes, and J. Malik. Scale-invariant contour completion using conditional random fields. In *ICCV*, 2005.

[14] X. Ren and J. Malik. A probabilistic multi-scale model for contour completion based on image statistics. In *ECCV*, page I: 312, 2002.

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

[16] L. Williams and D. Jacobs. Stochastic completion fields: A neural model of illusory contour shape and salience. In *ICCV*, 1995.

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