Boundary Estimation from Intensity/Color Images with Algebraic Curve Models

Tolga Tasdizen
Division of Engineering
Brown University
Providence, RI 02906
tt@lems.brown.edu

David B. Cooper
Division of Engineering
Brown University
Providence, RI 02906
cooper@lems.brown.edu

Abstract

A new concept and algorithm are presented for non-iterative robust estimation of piecewise smooth curves of maximal edge strength in small image windows – typically 8×8 *to* 32×32*. This boundary-estimation algorithm has the nice properties that it uses all the data in the window and thus can find locally weak boundaries embedded in noise or texture and boundaries when there are more than two regions to be segmented in a window; it does not require step edges – but handles ramp edges well. The curve-estimates found are among the level sets of a d'th degree polynomial fit to "suitable" weightings of the image gradient vector at each pixel in the image window. Since the polynomial fitting is linear least squares, the computation to this point is very fast. Level sets then chosen to be appropriate boundary curves are those having the highest differences in average gray level in regions to either side. This computation is also fast. The boundary curves and segmented regions found are suitable for all purposes but especially for indexing using algebraic curve invariants in this form.*

Acknowledgment This work was partially supported by NSF Grants #IIS-9802392 and #BCS-9980091.

1. Introduction

We present a new concept and algorithm for estimating boundary curves in images. These have two uses: 1) for general applications, 2) for direct extraction from images of algebraic curves to represent image boundaries. Algebraic curve fitting to binary images has been studied extensively [2, 3, 6, 7, 4, 1, 5]. Boundary contours of objects of interest can be easily extracted from binary images. However, obtaining binary images automatically from real sensory data such as an intensity image involves the unsolved problem of segmentation. Consequently, these approaches to algebraic curve fitting which rely on a prior segmentation step will not work in a general setting. Our objective in this regard is to estimate algebraic curves that represent boundary curves in images without assuming any prior segmentation. Our algorithm can be seen as a combined algebraic curve fitting/contour detection which uses the representation power of algebraic curves to robustly detect contours from images.

Sec. 2.1 gives a brief overview of algebraic curves. A method for extracting "appropriate" polynomials from images is given in Sec. 2.2. Sec. 3 deals with estimating desired contours from images, based on the polynomials from Sec. 2.2.

2. Fitting Algebraic Curves to Images

2.1 Shape Representation by Algebraic Curves

A dth degree algebraic curve, also called an implicit polynomial curve (IP curve) is the set of points $\{(x,y)\}$ satisfying $f(x,y)=0$ where $f(x,y)$ is the $2D$ polynomial $\sum_{0\leq j+k\leq d} a_{jk}x^j y^k$. More generally, the $2D$ polynomial yields a set of IP curves $\{(x,y) : f(x,y)=L\}$ where $L\in\mathbf{R}$, the set of real numbers. These are the *level sets* of f. If we choose $L=0$, we obtain the *zero set* of f. If we let L assume values other than 0, multiple shapes can be represented by a single polynomial f. We make use of this in Sec. 2.2 and 3. A subset of *level sets* for a single polynomial is the model we extract for the salient boundary curves in a window of an image.

2.2 Direct Fitting to Intensity/Color Images

The polynomial estimation algorithm proposed next is based only on image gradient information. Let $I(x,y)$ be an intensity image and $\mathbf{G}(x,y) = \nabla I(x,y) = \left[\frac{\partial I}{\partial x}(x,y)\ \frac{\partial I}{\partial y}(x,y)\right]^T$ its gradient vector map. Let $w_i = ||\ \mathbf{G}(x_i,y_i)\ ||$, the length of the gradient vector at (x_i,y_i). If the input is a color image, $\mathbf{G}(x,y)$ is computed as in [8] to make full use of the color information. A few example gradient vectors in a window from an intensity image are drawn

in Fig. 1(a1). Notice that some gradient vectors are due to the boundary in the image and others are due to noise. Let $\mathbf{N}_i = \mathbf{G}(x_i, y_i)/w_i$, the unit vector in the direction of the intensity gradient pointing from darker to brighter regions. Compute $\mathbf{T}_i$, the orthogonal unit vector to $\mathbf{N}_i$ by rotating $\mathbf{N}_i$ 90° clockwise. Then the polynomial model coefficients a_{jk} can be estimated by minimizing

$$E = \sum_{1 \le i \le n} w_i^2 \left((\mathbf{N}_i \cdot \nabla f_i - 1)^2 + (\mathbf{T}_i \cdot \nabla f_i)^2 \right) \qquad (1)$$

where i enumerates every pixel (x_i, y_i) in a image window and ∇f_i is the gradient vector of f at (x_i, y_i). Minimization of (1) is a linear least squares problem. Linear methods offer a definite speed advantage over others and speed is a primary concern in dealing with images since the amount of data in an image is normally orders of magnitude more than a data set of contour points. If a pixel (x_i, y_i) is on an edge contour, $\mathbf{T}_i$ and $\mathbf{N}_i$ by definition will approximately be in the directions tangent to the contour and perpendicular to the contour, respectively, and w_i will be the strength of the intensity gradient across the edge. The first term of (1) approximately constraints the directional derivatives of f to have value 1 across the contour and the second term approximately constrains them to have value 0 along the contour. This forces f to have *level sets* that are edge contours in the image. of f. This fact is the main idea behind the fitting/detection algorithm and is used in Sec. 3 to detect actual contours in the image.

Some insight into this choice of E is as follows. We do not want to model image intensity by $f(x, y)$ because the exact behavior of $I(x, y)$ is irrelevant to boundary estimation and estimation of f by approximation to I imposes unnecessary constraints on f – especially if I varies considerably. This leaves few degrees of freedom for $f(x, y)$ to model boundary curves. Hence, instead of forcing $f(x, y)$ to have gradient strength $|| \ \mathbf{G}(x, y) \ ||$, we simply force $\nabla f(x, y)$ to have the direction of $\mathbf{G}(x, y)$ but to have unit magnitude. It is this directional information that determines f. In addition, the unit-gradient magnitude restriction decreases the effects of noise and texture, since these can produce large gradients. Beyond these considerations, this restriction stabilizes the fitted f for other reasons, see [5]. Finally the weighting w_i^2 in (1) does give more influence to pixels (x_i, y_i) that have large image gradients. This will further improve fits in the presence of noise or texture if the gradient strengths of pixels that are on the contour are on the average larger than gradients due to noise/texture, an assumption likely to be met in the majority of images.

3. Boundary Contour Detection

After a polynomial f has been fitted to an image window such as Fig. 1(a1)-(a6), the objective is to detect which *level sets* if any correspond to actual boundary contours in the image. This is accomplished by the following steps:

1. *Level sets* of f are computed to sub-pixel accuracy on a discrete grid in a single step linear computation. $f(x_i, y_i)$ is computed for every pixel. Then *level sets* for desired level values are located by using linear interpolation between $f(x_i, y_i)$. The level values that are used in the previous computation are chosen such that every pixel (x_i, y_i) will be included in a *level set*. This results in a small number (typically around 20 for a 16×16 window) of *level sets* that cover all of the pixels in the window, Fig. 1(b1)-(b6).

2. *Level sets* are grouped into branches. If there is a single image boundary or non-intersecting multiple boundaries in the image window as in Fig. 1(a1)-(a3), all *level sets* will most likely (but not necessarily) be grouped in the same branch, Fig. 1(b1)-(b3). When a junction is present in the image window as in Fig. 1(a5)-(a6), it will manifest itself as a hyperbolic point of f and *level sets* around the hyperbolic point will be grouped into different branches that come together at this point, Fig. 1(b5)-(b6). Thus the existence of a hyperbolic point of f indicates a possible junction in the image. It is important to observe that f can have hyperbolic points even when the image does not have a junction, an example is Fig. 1(a4),(b4). Consequently, every possible junction has to be verified to be considered a salient junction; this is explained in the next step.

3. Define the *contour strength measure* for a curve as the average gray value in $I(x, y)$ in a 1-pixel wide region to one side of the curve minus that for the other side of the curve. Fig. 1(c1) shows this measure for all the curves. As expected it peaks very strongly for the *level set* matching the contour in the image. The horizontal dashed line indicates an adaptive threshold – the average gradient strength over the entire window. We detect boundary contours as *level sets* that are local maximums in terms of *contour strength measure* that are above this threshold.

As opposed to *average gradient strength measure* along the contour, this measure can be seen as a hypothesis test: a *level set* is hypothesized as being an actual contour, gray level averaging is done along the contour on either side but not across the contour and the absolute difference in averages is computed. Hypothesis averaging is much more effective than averaging by a Gaussian smoothing in two ways: (i) when the hypothesis is true, *contour strength measure* is not weakened by averaging across the contour whereas a gaussian average and thus *average gradient strength measure* is and (ii) when the hypothesis is false and is in a noisy region, *contour strength measure* is much smaller than *average gradient strength measure* because of the larger extent of the smoothing along the false contour. *Contour strength measure* makes it possible to detect contours in images with large amounts of noise, Fig. 1(a2), and textured images, Fig. 1(a3). Fig. 1(a2) has the same bound-

ary as Fig. 1(a1), but the average region intensities have been moved closer to each other at 100 and 150, and white noise with standard deviation 25 has been added. Recall that no prior smoothing is performed on the image. Thus, it is remarkable that the curve is still detected ; it is barely above the adaptive threshold, but it is still a strong local maximum. Edge detection methods which are based on much more local operations are very likely to fail on such images as this. Fig. 2(a) and (b) show the results of the *Canny edge detector* run on Fig. 1(a2) and (a3), respectively. A common set of parameters was chosen manually to approximately optimize performance of the *Canny edge detector* for these two image windows. As expected *Canny edge detection* results are not good here. The *Canny edge detector* works fine with the other images in Fig. 1; however, it is important to point out that its output is individual edge elements that need to be grouped together, usually a very hard task in the presence of noise or gaps. The output of our approach is curves of moderate length that can be put together to form complete curves, an easier task than grouping edge elements.

A junction in the image is detected if f has a hyperbolic point indicating the existence of a possible junction in the image window and if salient contours are detected, as explained above, on the branches that form the hypothesized junction. One junction was detected in Fig. 1(a5) and two were detected in Fig. 1(a6). When a junction is present in the image window, the contour representations obtained from the *level sets* of f are not of as high quality as when there is no junction. This is because polynomials are smooth functions and can not model junction curves exactly. A future research direction is using iterative refinement methods such as active contour models using the polynomial *level sets* as good initial estimates if a junction is detected in an image window.

4. A segmentation test is performed to verify the validity of the regions formed by the detected contours. The average intensity in every region is computed. Absolute differences in averages for pairs of adjacent regions is tested against a threshold: the minimum valid average intensity difference for distinct regions. It is computed globally by dividing the intensity range of the entire image by 16. Here 16 is the only absolute threshold in the system and corresponds roughly to the maximum number of distinct gray levels allowed in the perception of a scene. Threshold at this stage is justified because local information has been collected into regional information which is more robust. Using this final segmentation test we are able to discard false contours that manage to pass the *contour strength measure* test, see Fig. 1(c5). Because it is based on the entire region segmentation and it uses a global threshold, this test is more powerful. *Level sets* corresponding to detected boundaries are marked in Fig. 1(b1)-(b6).

Automatically choosing a degree for f

So far we have not specified how to choose the degree of the polynomial necessary for any given image window. For example, degrees 2 and 4 were used for the first 3 and the last 3 rows in Fig. 1, respectively. Given a desired maximum degree, an appropriate degree that is less than or equal to it is chosen automatically. Polynomials of degrees 1 to the maximum degree are fit, and the appropriate degree is chosen based on the resulting *contour strength measures*. For a given degree, we require a 5% increase in this measure for the curves detected over curves from lower degree polynomials to justify this degree. The maximum justifiable degree is chosen.

Processing an Image

For stability and faster computation purposes, it is necessary to keep the maximum degree of the polynomials to moderate degrees. This is achieved by processing images by dividing them into windows and processing each window independently. On the other hand, more regional information is available in larger windows making accurate curve detection easier. Using 16×16 windows and restricting f to be of maximum degree 4 provides a good tradeoff between the maximum degree of the polynomial used and curve detection power for most of the images the algorithm was tried on. The windows are placed at intervals of 8 pixels and thus overlap. This to make sure that curves that are close to the boundary between two adjacent non-overlapping windows are not missed. Fig. 3 show an intensity image and the boundary curves estimated by our algorithm. It has relatively few false alarms in the textured background in the upper half of the image. It also does a good job on boundary curve detection where salient boundaries exist in the image. It takes approximately 1 minute to process a typical 256×256 image on a *SUN Ultrasparc* workstation.

4. Conclusions

The advantage of this approach to boundary estimation versus using local edge detectors and relaxing/grouping salient subsets of their outputs in two independent steps is that our approach treats all the data in a window of image simultaneously and allocates limited boundary representation resources to the most "salient" boundary curves within the window. There are no parameters to be tweaked! Maximum complexity of boundary representation is determined by polynomial degree. Our procedure is not plagued by gaps, blobby noise and other situations that create problems for iterative algorithms. An important advantage of using only *derivative constaints* in the fitting is the capability of representing multiple curves with a single polynomial. The *zero set* of f, the standard algebraic curve representation, is

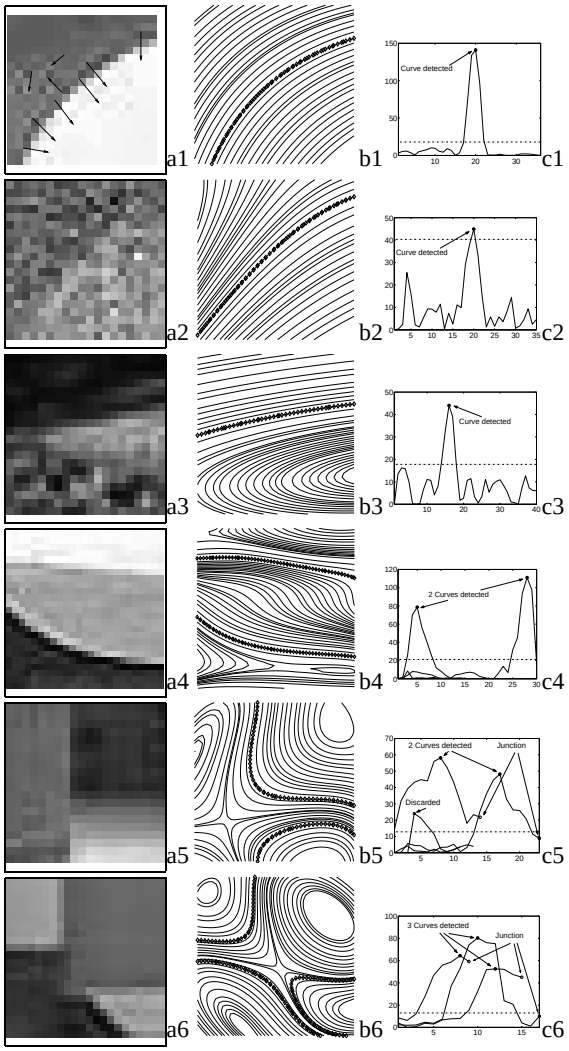
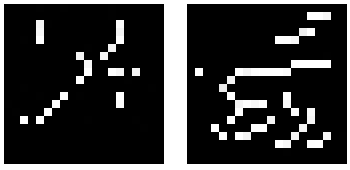
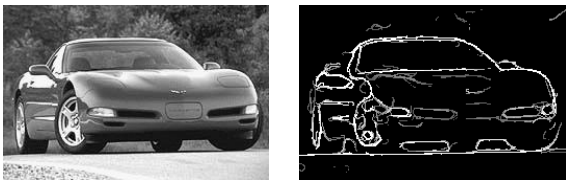


Figure 1. (a)20x20 image windows. (b) Level sets; detected contours are marked with symbols. Degree 2 and 4 polynomials were used for the first 3 and the last 3 rows, respectively. (c) Contour strength measure (vertical axis) in different branches plotted against consecutive level sets (horizontal axis).



(a) (b)

Figure 2. Canny edge detector outputs for windows Fig.1(a2) and (a3)



(a) (b)

Figure 3. (a) An intensity image, (b) Curves detected with the proposed algorithm

capable of representing only a single curve except under unusual circumstances. Finally, since boundaries are extracted and estimated as algebraic curves, invariants of these curves can immediately be used for object recognition or image indexing purposes. Having a single polynomial representing two or more curves is potentially very powerful for invariant shape-based indexing into image databases.

References

[1] M. M. Blane, Z. Lei, H. Civi, and D. B. Cooper. The 3l algorithm for fitting implicit polynomial curves and surfaces to data. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(3):298–313, March 2000.

[2] D. Keren, D. B. Cooper, and J. Subrahmonia. Describing complicated objects by implicit polynomials. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(1):38–53, January 1994.

[3] D. Keren and C. Gotsman. Fitting curves and surfaces with constrained implicit polynomials. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(1):31–41, Jan. 1999.

[4] Z. Lei, M. M. Blane, and D. B. Cooper. 3L fitting of higher degree implicit polynomials. In *Proc. of Third IEEE Workshop on Applications of Computer Vision*, pages 148–153, Sarasota, Florida, Dec. 1996.

[5] T. Tasdizen, J.-P. Tarel, and D. Cooper. Improving the stability of algebraic curves for applications. *IEEE Trans. on Image Processing*, 9(3):405–416, March 2000.

[6] G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Trans. on PAMI*, 13(11):1115–1138, Nov. 1991.

[7] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D. Kriegman. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE Trans. on PAMI*, 16(3):287–303, March 1994.

[8] S. D. Zenzo. A note on the gradient of a multi-image. *Computer Vision, Graphics, And Image Processing*, 33:116–125, 1986.