EFFICIENT IMAGE CONTOUR DETECTION USING EDGE PRIOR

Jiangping Wang[1]*, Changhu Wang*[2]*, and Thomas Huang*[1]

[1]Beckman Institute, University of Illinois at Urbana-Champaign, USA
[2]Microsoft Research Asia, Beijing, China
{jwang63,huang}@ifp.uiuc.edu, chw@microsoft.com

ABSTRACT

An effective and efficient image contour detector is highly desired due to its wide applications in computer vision and multimedia retrieval. However, the state-of-the-art image contour detection algorithms are very computationally intensive, and thus impractical for web-scale applications. In this work, we study the relationship between edge detection and contour detection, based on which an edge-based image contour detection algorithm is proposed. This algorithm fully makes use of cheap edge information for efficiency purpose. The experiments on benchmark data sets show that, the proposed contour detector works much faster than existing state-of-the-art algorithms while maintaining high accuracy, and thus suitable for large-scale applications.

Index Terms— Image contour detection, edge detection

1. INTRODUCTION

Image contour detection plays an important role in computer vision, and has been widely used in a variety of vision tasks such as image segmentation [1, 2], image retrieval [3], object detection [4], and object recognition [5]. In 2004, Martin *et al.* [6] proposed the classic gradient paradigm, referred to as standard Pb, in which a combination of local cues from brightness, color, and texture channels were learned from the Berkeley Segmentation Data Set (BSDS300) [7]. Since then, a number of image contour detectors have been proposed and validated on this data set, such as Boosted Edge Learning (BEL) [8], Ultrametric Contour Map (UCM) [9], mPb [10], and Min-Cover [11].

In [12], the so called globalized probability of boundary (gPb) method was proposed and achieves the highest accuracy in terms of F-measure on BSDS300 and an extended dataset BSDS500. However, gPb is quite computationally intensive since it extracts multiscale local gradient features and involves solving a generalized eigenproblem. Many web-scale applications could not make use of such effective contour detector due to its high computation cost. To speed up the detection process, a Graphics Processing Unit (GPU) based method was proposed in [13]. However, in most cases where GPUs are not affordable or impractical, the optimization in the principle of the algorithm itself becomes highly desired. Therefore, in the current era of big data, how to speed up the state-of-the-art contour detection algorithms while keeping high performance becomes a crucial problem, which is the main motivation for this work.

In this paper, we investigate the relationship between edge detection and contour detection, and speed up the contour detection algorithms by making use of edge information. Different from boundary detection, which aims at obtaining middle-level boundaries of objects or scenes, edge detection extracts low-level information such as sharp discontinuities in the brightness. Although it was pointed out in [6] that, edge detectors perform poorly for contour detection purpose because they cannot distinguish textured regions from object/scene boundaries, we will show in this work that edge information is very useful in the sense that it provides cheap priors for the contour detector, based on which an edge-based Pb detector is proposed. The proposed detector accelerates the detection process by almost an order of magnitude compared with the standard Pb while keeping high accuracy. It should be noted that, although we mainly study the standard Pb, the methodology naturally applies to other Pb-based detectors such as mPb and gPb.

The remainder of this paper is organized as follows. Section 2 introduces some related works including the standard Pb and edge detection. Section 3 presents our edge based contour detector. Section 4 presents experimental results. Finally, section 5 concludes our paper with discussion.

2. RELATED WORKS

2.1. The Standard Pb and Pb-based Contour Detectors

In the standard Pb detector, a function $Pb(x, y, \theta)$ is defined to predict the posterior probability of a boundary with orientation θ at each image pixel (x, y). It measures local cues computed over four channels: the L, a, b channels in the CIELab color space and a texture channel derived from a texton map. Each local cue is computed as histogram differences between

This work was partially done when the first author was an intern at Microsoft Research Asia. This work was supported in part by the National Science Foundation Grant DBI 10-62351.

the feature distributions in two half-disks with radius σ over eight orientations, in the interval $[0, \pi)$. Pb detector is constructed by linear combination of local cues at scale σ:

$$Pb(x, y, \theta) = \sum_i \alpha_i G_i(x, y, \theta), \quad (1)$$

where $G(x, y, \theta)$ is the gradient operator at (x, y) along direction θ, which is detailed in [6]. i denotes the feature channels. The weights α_i are learned on the training data set via logistic regression [6] or gradient ascent on F-measure [12]. The boundary strength at (x, y) is given by

$$Pb(x, y) = \max_\theta Pb(x, y, \theta), \quad (2)$$

based on which the probability of the pixel (x, y) being at a boundary is obtained by normalizing $Pb(x, y)$.

mPb was later proposed [10][12] which leverages multi-scale features to improve Pb. The mPb is described as

$$mPb(x, y, \theta) = \sum_s \sum_i \alpha_{i,s} G_{i,s}(x, y, \theta), \quad (3)$$

where $s \in \{\frac{\sigma}{2}, \sigma, 2\sigma\}$ indexes scale.

The state-of-the-art image contour detector gPb [14] takes mPb as an input to derive spectral cues sPb and finally combines mPb and sPb. gPb achieves highly accurate contours on several data sets but its applicability is rather limited by its prohibitively computational cost. It was recently speeded up by GPU in [13] but to our best knowledge no direct optimization method in algorithm has been proposed in the literature.

2.2. Edge Detector

As one of the fundamental problems in computer vision, edge detection has attracted many research efforts. Numerous edge detection algorithms have been proposed, such as Sobel operator, Prewitt operator, Laplacian detector and Canny detector. In our system, we employ Canny edge [15] detector to extract edge information from image, due to its very attractive properties including minimal response, good detection and localization ability. The standard Canny consists of four steps: smoothing the raw image with a Gaussian filter; applying directional Gaussian derivative filters and thus obtaining edge gradients and orientations; thinning the edge using non-max suppression; thresholding with hysteresis.

While it proves to be a very powerful technique for edge detection, unfortunately Canny edge detector performs poorly for contour detection [6]. The reason is that Canny detector cannot distinguish contours from highly textured areas. Is it possible to use edge detector such as Canny in contour detection technique? We will give the answer to this question in section 3.

Fig. 1. Top row: RGB images; Second row: Canny edges; Third row: Pb contours; Bottom row: human labeled groundtruth.

3. EDGE BASED CONTOUR DETECTION

3.1. Problem Formulation

From Eqn. (1) we can see that, Pb computes the boundary strength for every pixel in the image, and the four local cues along eight directions are extracted. This is inherently computationally intensive. For example, it takes more than one minute to extract contours from a 321×481 image, if implemented with C# on a single-core CPU. This speed is impractical for web-scale applications such as building sketch-based image search engine [3].

In this work, we aim at speeding up the state-of-the-art contour detectors from the algorithm perspective. From Figure 1 we can see that, the edge maps extracted by Canny detector contain almost all of the location information of the contours obtained by Pb and groundtruth. In other words, almost all the contours extracted by Pb is a subset of the edge map. In some sense, we can regard low-level edge information as kind of prior, which may contain very important clue for boundaries. We can show with simple probability calculation that making use of low-level edge information is possible and reasonable to speed up contour detection.

We adopt the notations in [8] where $S(x, y) = 1$ if the boundary passes through location (x, y), and $S(x, y) = 0$ otherwise. Let $F(x, y)$ denote the feature vector at location (x, y), and $P(S(x, y) = 1|e(x, y))$ the probability that the pixel (x, y) is on the boundary given the edge information. Thus, given the feature $F(x, y)$, the probability of the pixel

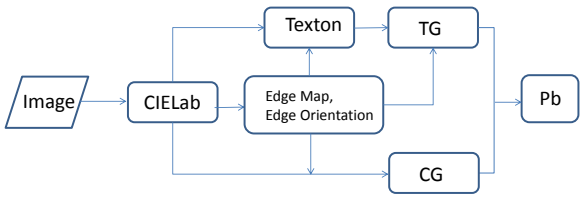


Fig. 2. System flowchart. CG denotes color gradients from L,a,b channels. TG is the texton gradient.

(x, y) being on the boundary is:

$$\begin{aligned} &P(S(x,y)|F(x,y)) \\ &= \sum_{i=0}^{1} P(S(x,y), e(x,y) = i|F(x,y)) \\ &= \sum_{i=0}^{1} P(S(x,y)|F(x,y), e(x,y) = i)P(e(x,y) = i|F(x,y)) \end{aligned} \tag{4}$$

Edge information is easy to compute. Meanwhile, if we assume that the majority of pixels on the boundaries also lie on the edges, i.e., $P(S(x, y) = 1|e(x, y) = 0)$ is extremely low, Eqn. (4) can reduce to

$$\begin{aligned} &P(S(x,y) = 1|F(x,y)) \\ &\approx P(S(x,y) = 1|F(x,y), e(x,y) = 1)P(e(x,y) = 1|F(x,y)) \end{aligned} \tag{5}$$

whose interpretation is that if the edge map contains the majority of pixels of the contours, we can only apply the contour detector to the edge pixels.

Furthermore, the orientation information of edges can benefit gradient feature-based contour detectors such as Pb-based methods. *Ideally* if the orientation of a boundary is equal to that of an edge, then Pb can further reduce to

$$Pb(x, y) = \sum_i \alpha_i G_i(x, y, \theta_e), \tag{6}$$

where $(x, y) \in supp(e)$, e is the edge map, $supp(e)$ is the support set of e, i.e., the edge pixels, and θ_e is the edge orientation at (x, y) (quantized to be one of eight orientations). It is obvious that the new detector will dramatically reduce the computational cost, which will be validated in Section 4. Of course, this derivation is based on our heuristic and ideal assumptions for demonstration purpose. In the following section, we will introduce the system design and the approaches to handling the practical issues.

3.2. Edge-based Contour Detector

Figure 2 illustrates the flowchart of our contour detection system. First, the edge information such as edge map and corresponding orientations is extracted from the input image. For gray images, the algorithm works on the intensity channel; for color images, three channels L,a,b in CIELab color space are computed, and then the edge information extracted from each channel is combined and further thinned to form a single edge map with orientations. In practice, we find that extracting edge information from the intensity channel of color image is fairly enough for further processing, especially for natural images.

With edge information, we can greatly reduce the computational cost of the standard Pb detector by the following strategies. 1) For the texton computation, *only* assign texton indexes to those pixels within the disc with radius σ centered at each edge pixel. 2) Extract local cues *only* on edge pixels along a subset of eight orientations. Ideally, one would like to extract the gradient cues along the edge orientation. However, there are two issues that should be considered. First, an edge orientation may not be consistent with a boundary orientation, especially in the joint locations in the image. Second, it is well known in computer graphics that a pixel has an area, and even a visually straight line with some slope in the image is not really straight in pixel domain [16]. This makes it difficult to calculate the exact edge or boundary orientation. Thus, the assumption that the edge orientation is equal to the boundary orientation in Pb is too strict and impractical. To make use of the edge orientation to speedup Pb, here, we relax this assumption by taking into account the computed edge orientation (that is quantized) and its adjacent orientations as well. In other words, we extract local cues along several orientations excluding those (almost) perpendicular to the edge orientation.

In the implementation, the Canny edge detector is adopted, which has become one of the most successful edge detectors since it was invented in 1986 [15]. Its good detection and localization properties make our assumption reasonable, while its minimal response property means less noisy edges are extracted, suggesting less pixels considered for further processing.

Our training phase is slightly different from the standard Pb in that only local cues on the edge map are used as positive or negative features, without particular smoothing processing. To overcome the well known double-peak problem in texture edge detection [6, 1], Pb applies directional 2D Savitsky-Golay (SG) filtering to localize the local cues. In contrast, our approach naturally avoids the double-peak problem due to the good localization ability of Canny detector, though at cost of underestimating the texture gradient response.

4. EXPERIMENTAL RESULTS

4.1. Evaluation Measurement

For performance evaluation, we base the analysis on BSDS300 and BSDS500 and adopt the evaluation methodology introduced in [6]. It compares machine detected bound-

Table 1. Comparison of average runtime in seconds for each image of size 321×481 or 481×321 on the testing set of BSDS500 Color.

Method	Preprocess	Textons	Local Cues	Postprocess	Total
Pb	0.09	12.79	77.86	0.56	91.30
Ours	0.49	8.11	1.43	0.04	10.07

aries to human marked boundaries in the precision-recall framework. To better quantify the performance of the detector, F-measure is used, which is defined as the harmonic mean of precision and recall:

$$F(P, R) = \frac{2PR}{P + R}. \tag{7}$$

Thus the performance of the detector is quantified by the maximal F-measure on the precision-recall curve on the testing dateset.

4.2. Experiment Setting

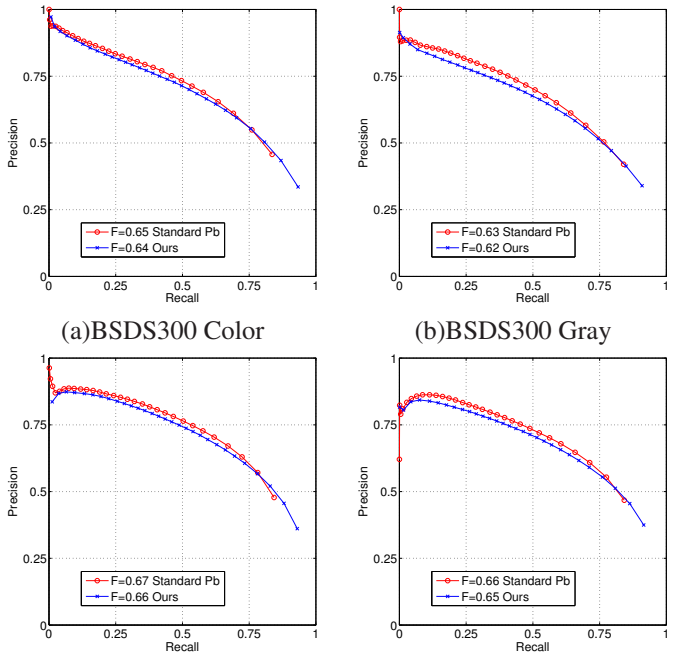
To ensure the integrity of the evaluation, we optimize all the parameters only on the training and validation set and perform testing blindly on the testing set. For Canny edge detector, there are three parameters: the standard deviation of Gaussian filters, the percentage of non-edge pixels (PNP), and the threshold ratio. For the Pb method, there are a set of parameters like the scale of local cues and fitting weights for logistic regression, all of which are discussed in details in [6]. In our experiments, we found that the optimal standard deviation of Gaussian filters in all cases is 1. The F-measure is not so sensitive to PNP and threshold ratio when they are in normal range. Suggested value for PNP is 0.6 and threshold ratio is 1. The experiments were conducted on both color and grayscale versions of BSDS300 and BSDS500.

For a fair comparison, we implemented both standard Pb and the proposed detector in C# on a single core of Intel(R) Core(TM)2 Quad CPU Q9400.

4.3. Performance Analysis

4.3.1. Comparison with Standard Pb

Figure 3 presents the comparison results between Pb and our method on BSDS300 and BSDS500. It can be observed that our method achieves competitive performance in terms of precision-recall curves and F-measures, in all cases as opposed to Pb. Our detector attains F-measure 0.66 on color images of BSDS500 and 0.64 on BSDS300, suggesting its high-quality compared to the state-of-the arts [14]. Also this is a strong sign that a good edge detector such as Canny detector can capture salient contours in the image.



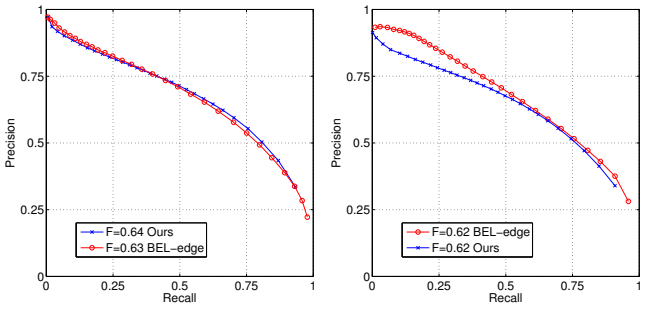
(a)BSDS300 Color (b)BSDS300 Gray

(c)BSDS500 Color (d)BSDS500 Gray

Fig. 3. Comparison of the proposed contour detector and the standard Pb method on BSDS.

Table 1 presents the comparison of average runtime between the proposed detector and Pb, on the testing set of BSDS500 Color. Overall, the proposed detector works almost 9 times as fast as Pb. Note that the runtime of the proposed contour detector could be less if using more advanced computer hardware. In the preprocess step, our detector consumes slightly more runtime due to the extra Canny edge detection, but the total runtime of this step is less than half a second and thus can be neglected. In all other steps, it clearly demonstrates the efficiency of the proposed detector over Pb, which is consistent with the previous analysis in Section 3. Specifically, in the texton computation step, instead of applying Gaussian derivative filter bank that assigns texton index to every pixel in the image, the proposed detector only works on the pixels around the edges in the edge map, reducing the runtime almost by one third. As for the extraction of local cues, we can observe a huge drop of runtime by using our detector. Pb computes the local cues *in every pixel along eight orientations*, which is time consuming and inefficient. In contrast, our detector only extracts local cues on edge pixels along fewer orientations (the edge orientation and orientations nearby, see Section 3.2), which dramatically decreases the runtime *by over* 50 *times* in this step. The post-processing of the proposed method requires less time because it computes the probability of boundary map on fewer pixels and fewer orientations, without using SG filter for localization purpose.

It is worth noting that, we are not targeting at designing



(a)BSDS300 Color (b)BSDS300 Gray

Fig. 4. Comparison of the proposed contour detector and BEL-edge on BSDS300.

the best contour detector with highest possible F-measure on BSDS; instead, we aim at speeding up the state-of-the-art contour detectors from the algorithm perspective without (if any) much degradation of performance. On the one hand, Pb is often used as a high-quality image contour detector in practice for a number of real-world applications [3], and it is highly desired to obtain a faster version of Pb; on the other hand, Pb is an essential and fundamental part of the state-of-the-art contour detectors, such as mPb and gPb whose applicability is certainly limited by their prohibitively computational cost, so our work provides one means of speeding up mPb and gPb. In cases where GPUs are not practical or CPUs are preferred (e.g., distributed computing system), the efficient image contour detection is highly desired. Also, a high end GPU is required to speed up the Pb method by an order of magnitude [13]. As such, our proposed edge based contour detection algorithm can be regarded as complementary of the GPU-based algorithm [13].

4.3.2. Comparison with BEL-edge

We notice that the fast version of BEL [8] (referred to as BEL-edge) was posted on `http://www.loni.ucla.edu/~ztu/Download.htm`, which claims around 12 seconds of runtime on an image of 481×321. We are interested in comparing our detector with BEL-edge. Since we cannot access the source code of BEL-edge, in the experiment we run the provided executable file of its detector trained on BSDS300[1]. Figure 4 shows the comparison results in terms of precision-recall curve and F-measure. As we can see, our detector performs better than BEL-edge, though not significantly. Actually, our detector is much simpler, involving much fewer features than BEL-edge (50,000 features). Furthermore, our detector potentially can be used to speed up the state-of-the-art image contour detector gPb.

[1]Since the BEL-edge detector trained on BSDS500 is not provided by authors, we cannot compare the two algorithms on BSDS500.

4.3.3. Strengths and Weaknesses of the New Method

The extensive experiments conducted on BSDS300 and BSDS500 above have demonstrated the high-quality of the proposed image contour detector. Figure 5 shows illustrative results using newly proposed image contour detector. We can see our contour detector performs quite well on the left two images, compared to the Pb. Here we emphasize the scalability of the new contour detector with respect to image size. From Table 1, we can find that, for the new detector, computation of textons consumes most of the total runtime required for processing one image of size 321×481 or 481×321. For images with size 321×481 or 481×321 in BSDS500, edge pixels by Canny detector occupy on average about 10% of the total pixels depending on the choices of scale and thresholds, the pixels around the edges which require texton computation take up about half the total pixels in the image, in this single scale case. As image size grows, the proportion of the edge pixels in one image generally should be smaller or roughly remain the same, say 10%, but the proportion of pixels around the edges for texton computation will decrease fast, due to the fact that the radius of disc in local cues extraction grows slowly than image size. Hence, it is reasonable to say that the larger the image size is, more efficient our proposed contour detector becomes compared to the standard Pb.

Regarding the weakness of the proposed image contour detector, when the assumption that the majority of pixels on the boundaries also lie on the edges is invalid, i.e., $P(S(x,y) = 1|e(x,y) = 0)$ is not low, our detector will fail to some extent. For instance, in Figure 5, for tiger image, our detector performs worse than Pb due to the failure of the assumption and lack of smoothing filtering. The boundary of the tiger in the image is not explicit but determined by disconnected edges near the boundary. Since Canny edge detector does not perform smoothing or connect edges, the resulted contour is not so smooth as Pb contour. However, in our experiments on BSDS, when our detector fails, the standard Pb does not perform well either in most cases. In practice, the edge based contour detector gives competitive results, compared to the standard Pb.

5. CONCLUSION

In this paper, we have proposed a fast contour detection algorithm that essentially combines Canny edge detector and the standard Pb method. The experiments have shown that the proposed detector works much faster than Pb while maintaining comparable performance. So we contend that Canny detector is able to detect salient contours in the image and thus can be used to speed up the Pb as well as other existing contour detection algorithms. Our work bridges the gap between edge detection and contour detection in the sense that edge detector provides cheap location and orientation information for contour detector. The good performance and high efficiency

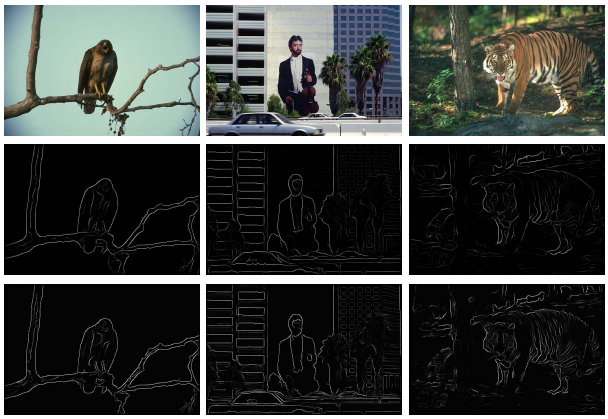


Fig. 5. Top row: RGB images; Second row: Pb contours; Bottom row: Edge-based image contours.

of our proposed detector make it applicable and promising in large-scale applications.

6. REFERENCES

[1] J. Malik, S. Belongie, T. Leung, and J. Shi, "Contour and texture analysis for image segmentation," *IJCV*, vol. 43, no. 1, pp. 7–27, 2001.

[2] P. Arbeláez, M. Maire, C. Fowlkes, and J. Malik, "From contours to regions: An empirical evaluation," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2009, pp. 2294–2301.

[3] Y. Cao, C. Wang, L. Zhang, and L. Zhang, "Edgel index for large-scale sketch-based image search," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2011, pp. 761–768.

[4] M. Maire, S.X. Yu, and P. Perona, "Object detection and segmentation from joint embedding of parts and pixels," in *International Conference on Computer Vision*. IEEE, 2011, pp. 2142–2149.

[5] S. Belongie, J. Malik, and J. Puzicha, "Shape matching and object recognition using shape contexts," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 4, pp. 509–522, 2002.

[6] D.R. Martin, C.C. Fowlkes, and J. Malik, "Learning to detect natural image boundaries using local brightness, color, and texture cues," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 530–549, 2004.

[7] 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 *International Conference on Computer Vision*, 2001.

[8] P. Dollar, Z. Tu, and S. Belongie, "Supervised learning of edges and object boundaries," in *Conference on Computer Vision and Pattern Recognition*. IEEE, 2006, vol. 2, pp. 1964–1971.

[9] P. Arbelaez, "Boundary extraction in natural images using ultrametric contour maps," in *Proceedings 5th IEEE Workshop on Perceptual Organization in Computer Vision (POCV'06)*, 2006.

[10] X. Ren, "Multi-scale improves boundary detection in natural images," *European Conference on Computer Vision*, pp. 533–545, 2008.

[11] P. Felzenszwalb and D. McAllester, "A min-cover approach for finding salient curves," in *IEEE CVPR workshop*. IEEE, 2006.

[12] M. Maire, P. Arbeláez, C. Fowlkes, and J. Malik, "Using contours to detect and localize junctions in natural images," in *Conference on Computer Vision and Pattern Recognition*, 2008.

[13] B. Catanzaro, B.Y. Su, N. Sundaram, Y. Lee, M. Murphy, and K. Keutzer, "Efficient, high-quality image contour detection," in *International Conference on Computer Vision*. IEEE, 2009, pp. 2381–2388.

[14] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, , no. 99, pp. 1–1, 2011.

[15] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, , no. 6, pp. 679–698, 1986.

[16] J.E. Bresenham, "Algorithm for computer control of a digital plotter," *IBM Systems journal*, vol. 4, no. 1, pp. 25–30, 1965.