ColorED: Color Edge and Segment Detection By Edge Drawing (ED)

Cuneyt Akinlar, Cihan Topal

{*cakinlar,cihant*}*@anadolu.edu.tr*

Anadolu University, Computer Engineering Department, Eskisehir, Turkey

Abstract

We extend our recently proposed real-time grayscale edge and segment detector, Edge Drawing (GrayED), to detect edge segments in color images. Edge Drawing for color images, named ColorED, takes in a color image, and outputs a set of edge segments, each a contiguous, 1-pixel wide chain of pixels. Detected edge segments are then passed through an *'a contrario'* validation step due to the Helmholtz principle to eliminate perceptually invalid detections. We quantitatively evaluate ColorED with different colorspaces and vector gradient operators within the precision-recall framework of the widely-used Berkeley Segmentation Dataset and Benchmark (BSDS300), and compare its results with those of GrayED and a color version of the Canny edge detector named ColorCanny. We conclude that color edge detection is in general superior to grayscale edge detection, and that ColorED with edge segment validation (ColorEDV) greatly outperforms GrayED, ColorED, and ColorCanny while taking reasonable time to execute.

Keywords:
Color Edge Detection, Edge Drawing (ED), Vector Gradient Operators, Compass, Edge Segment Validation, Helmholtz principle

1. Introduction

Edge detection is a very important problem and a first step in many image processing and computer vision applications including contour detection and image segmentation, robot vision, object recognition, object tracking, image registration, etc. While many traditional edge detectors [1, 2, 3, 4] work with grayscale images, color edge detection is gaining popularity since color reveals extra information that may help in detecting the correct boundaries between different objects in an image.

To motivate the need for color edge detection, Fig 1(a) shows an 8x8 colored checkerboard pattern where each color has the same intensity, i.e., each color evaluates to the same grayscale value. Fig. 1(b) shows the ground truth edges for this pattern as seen by a human observer. Although it is impossible for a grayscale edge detector to detect any edges for this image, a color edge detector can easily detect most of the edges as illustrated in Fig. 1(c). In general, when two objects in a scene have the same intensity, a grayscale edge detector fails to detect any edges between them although the objects may have different hue or saturation and the human visual system clearly separates the two objects from each other. In such cases, a color edge detector might be able to distinguish the two objects from each other and detect the boundary separating them.

While a color edge detector may be able to extract more edges than a grayscale edge detector, it has to process three times more data for an (R, G, B) image, which raises performance issues. Therefore, the biggest challenge facing a color edge detector is to process a multi-channel image as fast as possible so as to be suitable for use in high-speed applications. Obviously, the most important goal of extracting the correct edges should not be sacrificed for running time performance.

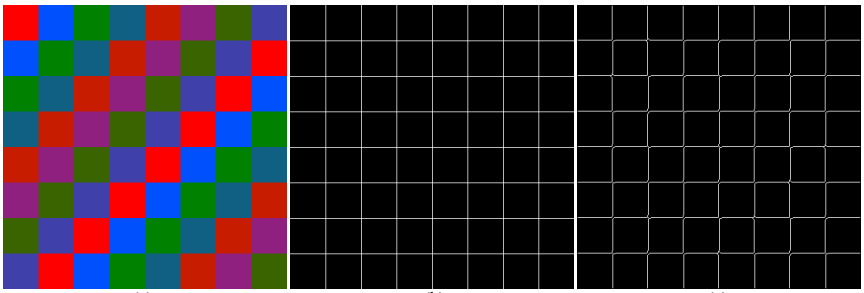
The simplest possible color edge detector is to process each color channel separately using a grayscale edge detector, and merge the edge map results of individual color channels together [6]. Edge localization during edge map merging is the biggest challenge facing this monochromatic-based approach, and it is not subject of this paper.

Due to the problems associated with monochromatic-based color edge detection techniques, vector-based approaches have been proposed. Authors in [5, 6] classify the vector-based color edge detection methods under three different categories: (1) Methods based on the first partial derivative of the color image, e.g., DiZenzo operator [7, 8], (2) Methods based on the second partial derivative of the color image, e.g., Cumani operator [9], and (3) Methods based on the vector order statistics [5, 10, 11, 12].

Koschan and Abidi [6] present an overview of the vector-based methods for color edge detection, and give a classification of the color edges as object, reflectance, shadow, specular, and occlusion edges.

Zhu et al. [5] give an analysis of color edge detection methods and study in detail the techniques based on vector order statistics. They conclude that the difference vector operator with adaptive filtering gives the most promising results. Shaikh [13] also concentrates on techniques based on vector order statistics, and introduces variations to the existing vector order statistics operators to attenuate noise.

Ruzon and Tomasi [14, 15] propose a completely new color gradient operator called the Compass. Their idea is to look at the color difference between two halves of a circle of a certain diameter using the Earth Mover's Distance [17] metric. The orientation that maximizes the color difference is assumed to



(a) (b) (c)

Figure 1: (a) An 8x8 colored checkerboard pattern, where each color evaluates to the same grayscale value, (b) Ground truth edges as seen by a human observer, (c) Edge map output by Color Edge Drawing (ColorED).

be the direction of the edge. Compass operator can also be used to detect junctions and corners in the color image.

Saez et al. [20, 21] present color gradients based on color visual perception that use the CIE Lab [22] colorspace. Their main objective is to study how different CIE color difference equations affect color gradient results in terms of correlation with the visual color perception. They evaluate different CIE color difference equations and conclude that CIEDE2000 [23, 24] gives the best results. Moreno et al. [25] also use an optimized version of the CIEDE2000 formula in their tensor based color edge detector.

In this paper, we propose a color edge and segment detector by extending our recently proposed grayscale edge segment detector, Edge Drawing (GrayED) [26]. The proposed algorithm, named ColorED, works by computing the vector gradient of the color image based on the first partial derivative, e.g., vector Prewitt/Sobel [8], DiZenzo [7], Compass [14, 15], and outputs a set of edge segments, each of which is a contiguous, 1-pixel wide chain of pixels. We also show how the detected edge segments can be passed through an *'a contrario'* validation step due to the Helmholtz principle [31, 33, 34] to eliminate perceptually invalid detections. We quantitatively evaluate ColorED with different colorspaces and vector gradient operators within the precision-recall framework of the widely-used Berkeley Segmentation Dataset and Benchmark (BSDS300) [37, 38], and compare its results with those of GrayED and a color version of the Canny edge detector named ColorCanny. Experimental results show that color edge detection is in general superior to grayscale edge detection, and that ColorED with edge segment validation (ColorEDV) greatly outperforms GrayED, ColorED, and ColorCanny, while taking reasonable time to execute making it suitable for high-speed image processing and computer vision applications.

The rest of the paper is organized as follows: Section 2 overviews GrayED and describes ColorED. Section 3 gives the details of different color vector gradient operators that may be used with ColorED. Section 4 describes the theory behind edge segment validation due to the Helmholtz principle. Experimental results are presented in section 5, and conclusions are described in section 6.

Algorithm 1 Grayscale Edge Drawing (GrayED)

Symbols used in the algorithm:
I: Input grayscale image
sigma: of the Gaussian smoothing kernel
thresh: Gradient threshold
MIN_SEGMENT_LEN: Length of the shortest segment
ES: Edge Segments

GrayED(I, sigma, thresh, MIN_SEGMENT_LEN)
I. S = SmoothImage(I, sigma);
II. GradientMap = ComputeGradientMap(S);
III. EdgeDirs = ComputeEdgeDirs(GradientMap);
IV. Anchors = ComputeAnchors(GradientMap, EdgeDirs);
V. ES = SmartRouting(GradientMap, EdgeDirs, Anchors, thresh, MIN_SEGMENT_LEN);
VI. return ES;

2. Color Edge Drawing (ColorED)

In this section, we describe how our grayscale edge segment detector, Edge Drawing (GrayED), can be extended to detect edge segments in color images. But first, we give an overview of GrayED and talk about what needs to be modified to make it suitable for color edge detection.

The pseudocode for GrayED is given in algorithm 1. Similar to Canny [2], which is the most popular binary edge detector for grayscale images, ED takes in a grayscale image I, passes it through a Gaussian smoothing kernel with the user supplied *sigma* to remove noise (step I), and computes the gradient map and edge directions from the smoothed image S (steps II & III). Gradient computation is performed by means of a first order partial derivative operator such as Prewitt, Sobel, Sharr etc. Unlike Canny though, ED follows a completely different approach for edge segment detection afterwards. While Canny performs non-maximal suppression followed by hysteresis to compute a binary edge map [2, 8], ED first computes a set of anchors, which are stable edge points at the peeks of the gradient map, and then links these anchors together by a method called the *smart routing* [26]. Smart routing starts at an anchor and follows a path over the peeks of the gradient map linking anchors and creating a chain of edge pixels. This is the biggest advantage of ED over traditional binary edge detectors such as Canny;

Algorithm 2 Color Edge Drawing (ColorED)

Symbols used in the algorithm:
R: Red channel of the input color image
G: Green channel of the input color image
B: Blue channel of the input color image
sigma: of the Gaussian smoothing kernel
thresh: Gradient threshold
MIN_SEGMENT_LEN: Length of the shortest segment
ES: Edge Segments

ColorED(R, G, B, sigma, thresh, MIN_SEGMENT_LEN)
I. (SR, SG, SB) = SmoothImage(R, G, B, sigma);
II. GradientMap = ComputeGradientMap(SR, SG, SB);
III. EdgeDirs = ComputeEdgeDirs(GradientMap);
IV. Anchors = ComputeAnchors(GradientMap, EdgeDirs);
V. ES = SmartRouting(GradientMap, EdgeDirs, Anchors, thresh, MIN_SEGMENT_LEN);
VI. return ES;

Algorithm 3 Color Canny (ColorCanny)

Symbols used in the algorithm:
R: Red channel of the input color image
G: Green channel of the input color image
B: Blue channel of the input color image
sigma: of the Gaussian smoothing kernel
lowThresh: Canny low threshold
highThresh: Canny high threshold
BEM: Binary Edge Map

ColorCanny(R, G, B, sigma, lowThresh, highThresh)
I. (SR, SG, SB) = SmoothImage(R, G, B, sigma);
II. GradientMap = ComputeGradientMap(SR, SG, SB);
III. EdgeDirs = ComputeEdgeDirs(GradientMap);
IV. BEM = NonMaxSupression(GradientMap, EdgeDirs);
V. Hysteresis(BEM, lowThresh, highThresh);
VI. return BEM;

that is, rather than returning a binary edge map, ED returns its result as a set of edge segments, each of which is a contiguous, 1-pixel wide chain of pixels. The edge segments can then be used in many higher level processing applications such as line detection [27], arc, circle and ellipse detection [28], corner detection [29], etc.

In the case of color edge detection, we have a multi-channel image to work with. To extend GrayED's pseudocode given in algorithm 1 for color edge detection, we realize that the only thing that needs to be modified is the gradient map and edge direction computation. We just need to employ a color vector gradient operator based on the first partial derivative of the color image, and the rest of the code can run smoothly without any modification.

The pseudocode for ColorED is given in algorithm 2. ColorED takes in a color image with three components denoted as R, G and B. Like GrayED, ColorED starts by smoothing each color component separately using a Gaussian kernel with the user supplied *sigma* to remove noise (step I). We would like to note that noise removal can also be performed by other edge preserving filters such as bilateral filtering, anisotropic filtering, etc., but these are not explored in this paper. The next two steps, which usually are implemented together within the same piece of code, involve the computation of the gradient map and edge directions, which needs to be performed by a color vector gradient operator based on the first partial derivative of the color image (steps II & III). The last two steps, anchor computation and smart routing, are the same as in GrayED, and do not require any modification. Some of the vector gradient operators suitable for use with ColorED are described in section 3.

We finish this section by presenting a simple extension of the grayscale Canny edge detector to color images. The pseudocode for ColorCanny is given in algorithm 3. It takes in a color image with three components denoted as R, G and B. After smoothing each of the color components and computing the gradient map and edge directions using a vector-valued gradient operator, ColorCanny proceeds the same as the grayscale Canny edge detector: Non-maximal suppression thins down the gradient map by eliminating non-peek pixels, and hysteresis with double thresholding eliminates the weak pixels. In section 5, we will compare the results produced by ColorED with those of ColorCanny.

3. Color Vector Gradient Operators

In this section, we present three vector-valued gradient operators for color images. Once the gradient magnitude and edge directions for a color image is computed using one of these operators, the edges can easily be detected either by ColorCanny (non-maximal suppression followed by hysteresis), or by ColorED (anchor computation and smart routing).

3.1. Traditional Vector Gradient Operators

The simplest vector gradient operator is to extend the traditional grayscale gradient operators based on the first partial derivative, i.e., Prewitt, Sobel, Scharr etc., to multi-dimensional colorspace, and express the gradient as the vector sum of the gradients of the individual components of the color image [5, 6, 8]. Let (R, G, B) be the three components of the color image, and let r, g and b be unit vectors along each component's direction. Define the gradient vectors in x and y directions as follows:

$$\begin{aligned} u = (u_1, u_2, u_3) = g_x = \frac{\delta R}{\delta x} r + \frac{\delta G}{\delta x} g + \frac{\delta B}{\delta x} b \\ v = (v_1, v_2, v_3) = g_y = \frac{\delta R}{\delta y} r + \frac{\delta G}{\delta y} g + \frac{\delta B}{\delta y} b \end{aligned} \quad (1)$$

In equation 1, the first partial derivative operator used for each color component, i.e., $\frac{\delta}{\delta x}$, can be any one of Prewitt, Sobel, or Scharr.

So far, we have two gradient vectors: g_x along the image's x direction, and g_y along the image's y direction. In order to compute the gradient magnitude and direction at a pixel,

we need to convert these gradient vectors to scalar values, which can be done using the L_p norms. L_p norm of a vector $v = (v_1, v_2, \ldots, v_n)$ is defined as

$$\{|v_1|^p + |v_2|^p + |v_3|^p + \cdots + |v_n|^p\}^{1/p}$$

Thus, L_1 norms of u and v in equation 1 are

$$g_x = |u_1| + |u_2| + |u_3|, g_y = |v_1| + |v_2| + |v_3|$$

Similarly, L_2 norms of u and v in equation 1 are

$$g_x = \sqrt{|u_1|^2 + |u_2|^2 + |u_3|^2}, g_y = \sqrt{|v_1|^2 + |v_2|^2 + |v_3|^2}$$

Finally, L_∞ norms of u and v in equation 1 are

$$g_x = max\{|u_1|, |u_2|, |u_3|\}, g_y = max\{|v_1|, |v_2|, |v_3|\}$$

Once g_x and g_y are computed, the gradient magnitude, g(x, y), and direction, dir(x, y), at pixel (x, y) can be calculated as follows:

$$g(x, y) = \sqrt{|gx|^2 + |gy|^2}, dir(x, y) = tan^{-1}(gy/gx)$$

Conceptually, the vector-valued gradient operator defined so far computes the Manhattan distance (L_1 norm), or the Euclidean distance (L_2 norm) between the color vectors. The advantage of this gradient operator is that it is very cheap to compute as we show in section 5, and produces reasonable results. However, the authors in [5, 12] state that the operator is very sensitive to small texture variations and to noise.

3.2. DiZenzo: The Tensor Gradient

DiZenzo [8, 7] shows that the traditional vector gradient operator described in section 3.1 may produce inaccurate results. Simply put, DiZenzo states that if the vectors in different color components oppose each other in reverse directions, a simple addition of the gradient vectors reduces the total derivative strength, which is wrong. To solve this problem, DiZenzo proposes what is called the tensor gradient for multi-sprectral images, where opposing vectors in different color components reinforce each other. Assuming u and v are the first partial derivatives of the color image in x and y directions respectively as shown in equation 1, DiZenzo's operator is defined as follows [8, 7]:

$$g_{xx} = u.u = \left|\frac{\delta R}{\delta x}\right|^2 + \left|\frac{\delta G}{\delta x}\right|^2 + \left|\frac{\delta B}{\delta x}\right|^2$$

$$g_{yy} = v.v = \left|\frac{\delta R}{\delta y}\right|^2 + \left|\frac{\delta G}{\delta y}\right|^2 + \left|\frac{\delta B}{\delta y}\right|^2$$

$$g_{xy} = u.v = \frac{\delta R}{\delta x}\frac{\delta R}{\delta y} + \frac{\delta G}{\delta x}\frac{\delta G}{\delta y} + \frac{\delta B}{\delta x}\frac{\delta B}{\delta y}$$

$$dir(x, y) = \theta = \frac{1}{2}tan^{-1}\left[\frac{2g_{xy}}{(g_{xx} - g_{yy})}\right]$$

$$g(x, y) = \left\{\frac{1}{2}\left[(g_{xx} + g_{yy}) + (g_{xx} - g_{yy})cos2\theta + 2g_{xy}sin2\theta\right]\right\}^{1/2}$$

DiZenzo's gradient operator given above computes both the gradient magnitude and edge direction at each pixel (x, y), which are enough for the rest of the color edge detection to proceed either by ColorCanny or ColorED.

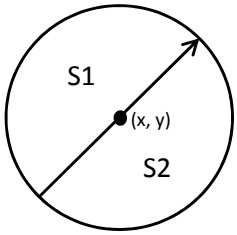


Figure 2: The Compass operator places a circular window of a certain diameter at a pixel (x, y) and computes the distance between the color distributions of the two halves of the circle, S1 and S2, using the Earth Mover's Distance. The orientation that maximizes the distance is assumed to be the direction of the edge.

3.3. The Compass Operator

Unlike the traditional way of first smoothing the image and then applying a derivative operator to compute the image gradient, Ruzon and Tomasi propose a totally different approach. Their proposal, named the Compass operator [14, 15, 16], not only computes the gradient magnitude and edge directions, but can also be used to detect junctions and corners in the image. The idea is to place a circular window of a certain diameter at a pixel (x, y), look at the color distributions of two halves of the circle and compute the distance between the color histograms or signatures of the two halves. The color difference between the two halves of the circle, S1 and S2 shown in Fig. 2, is computed by the Earth Mover's Distance (EMD) [17, 18], which is proven to be more robust than other distance measures. EMD arises in transportation problems and finds the minimum amount of physical work needed to move the masses in one pile to the other so that they are equal to each other. The operator tries out many different orientations (which is pointed to by the needle of the Compass), and the orientation that maximizes the difference between the two halves of the circle is assumed to be the orientation of the edge. The EMD distance of the two halves of the circle at the edge orientation is taken to be gradient magnitude for the center pixel (x, y).

Compass[1] is a very good gradient operator, but it is extremely slow as we also report in section 5. Recently, researchers have attempted ways of speeding up the Compass operator [19]. It is also important to note that when the Compass operator is used for color edge detection in algorithms 2 and 3, the image smoothing step is not executed. That is, the Gaussian smoothing of step I is omitted, and the input color image denoted by (R, G, B) is directly fed into the Compass operator for gradient magnitude and edge direction computation. Furthermore, the parameter *sigma* of the Gaussian smoothing kernel is converted into the radius of the Compass operator by the formula $radius = \lceil 3 * sigma \rceil$. Thus, if $sigma = 1.5$, then the radius of the Compass operator becomes 5 pixels.

4. Edge Segment Validation by the Helmholtz Principle

Given a set of edge segments, it is possible to pass them through a validation step to eliminate invalid detections. The

[1]The source code of the Compass operator can be downloaded from [16].

theory behind edge segment validation is based on the computational Gestalt theory and the Helmholtz principle [33, 34], which states that a geometric structure is perceptually *meaningful* if its expectation by chance is very small in a random situation. Authors in [31] show that a suitable random background model is the Gaussian white noise, where all pixels, and thus the gradient values and angles, are independent. According to the Helmholtz principle, no perceptually meaningful structure is visible within the background model, i.e., within a Gaussian white noise image, and any large deviation from the background model is perceptually visible if it corresponds to a predefined set of structures such as lines, curves, circles, ellipses etc. This essentially is an *'a contrario'* model, where the objects are detected as outliers of the background model.

The mathematical background of the theory for edge segment validation is described in [31, 33], and utilized in [32, 30] for grayscale images. Below, we simply extend that theory to color images to validate ColorED's edge segments.

Let I be a color image having $N \times N$ pixels, and let g be the gradient magnitude of this image computed by the traditional vector gradient operator described in section 3.1 via finite differences. Define H to be the cumulative distribution of the gradient g as follows:

$$H(\mu) = \frac{1}{M} \#\{x \in I | g(x) \geq \mu\}$$

where M is the total number of pixels having a gradient value bigger than 0, and # is the *'number of'* operator.

Consider the i^{th} edge segment S_i having length l_i. The number of connected pieces P of S_i is $[l_i \times (l_i - 1)]/2$, and the total number of connected pieces of all edge segments, N_p, is

$$N_p = \sum_i \frac{l_i \times (l_i - 1)}{2}$$

Given a piece $P = x_1, x_2, \ldots, x_l$ of an edge segment having length l, assume μ is the minimum gradient value of the pixels of P. Then, the Number of False Alarms (NFA) of this edge segment piece, P, is defined as $NFA(P) = N_p \times H(\mu)^l$ [31, 33, 32]. For all practical purposes, P is assumed to be valid if $NFA(P) \leq 1$, which corresponds to 1 false detection per image.

The edge segment validation method described so far is summarized in algorithm 4. Having computed a set of edge segments by ColorED, we simply pass them through this validation algorithm to eliminate false detections. The pieces of the edge segments that survive validation are then returned as output of ColorED with Validation (ColorEDV).

All and all, the validation method described in this section is the same as the one outlined for grayscale images in [31, 32, 30] except for the computation of the gradient magnitude g, which in turn determines the cumulative gradient distribution H. While the edge segment validation for grayscale images makes use of a scalar gradient operator via finite differences [31], the edge segment validation for color images makes use of a color vector gradient operator via finite differences as described in section 3.1.

Algorithm 4 Edge Segment Validation

Symbols used in the algorithm:
R: Red channel of the input color image
G: Green channel of the input color image
B: Blue channel of the input color image
ES: Edge segments detected by ColorED

ValidateEdgeSegments(R, G, B, ES)
I. g = ComputeGradientMap(R, G, B);
II. $H(\mu) = \frac{1}{M} \#\{x \in I | g(x) \geq \mu\}$
III. $N_p = \sum_i \frac{l_i \times (l_i - 1)}{2}$
IV. For each edge segment piece P of ES having length l do:
  (a) Let μ be the minimum gradient value on P
  (b) Compute $NFA(P) = N_p * H(\mu)^l$
  (c) If $NFA(P) \leq 1$, then P is valid
  (d) else P is invalid
V. Output all valid edge segment pieces, and discard others

5. Experimental Results

We analyze the performance of the proposed color edge detection algorithms, ColorED and ColorEDV, in seven steps. In section 5.1, we talk about the colorspace used in the experiments, and describe the framework and the performance metrics used for quantitative evaluation. In section 5.2, we compare the results by ColorED with the results by GrayED to show the need for and the importance of color edge detection. In section 5.3, we compare the results by ColorED with the results by ColorCanny to compare two different ways of obtaining edges. Section 5.4 shows the importance of edge segment validation. Section 5.5 presents an overall comparison of different edge detection algorithms and color vector gradient operators. Section 5.6 presents the running time results for the proposed algorithms. Finally, section 5.7 analyzes the performance of the proposed algorithms in noisy images.

5.1. Colorspace and the Evaluation Framework

The first thing that any color edge detection algorithm must decide is the colorspace to be used. Among the many different alternatives, we prefer the CIE Lab [22] colorspace, where L represents the lightness of the color, a represents its position between red/magenta and green, and b represents its position between yellow and blue [21]. CIE Lab is a uniform colorspace, meaning that the perceptual difference between any two colors in the colorspace can be measured by their Euclidean distance [21]. Although we do not present any experimental results to back our claims, it suffices to say that we experimented with many different colorspaces, and concluded that the best results are obtained with the CIE Lab colorspace. CIE Lab is also commonly used by other researchers in the literature for color edge and contour detection [25, 39, 40, 21].

The quantitative evaluation of the proposed color edge detection algorithms is performed within the precision-recall framework of the famous Berkeley Segmentation Dataset and Benchmark (BSDS300) [37, 38]. BSDS300 consists of 300 color images with 5 to 8 human annotated ground truth boundary infor-

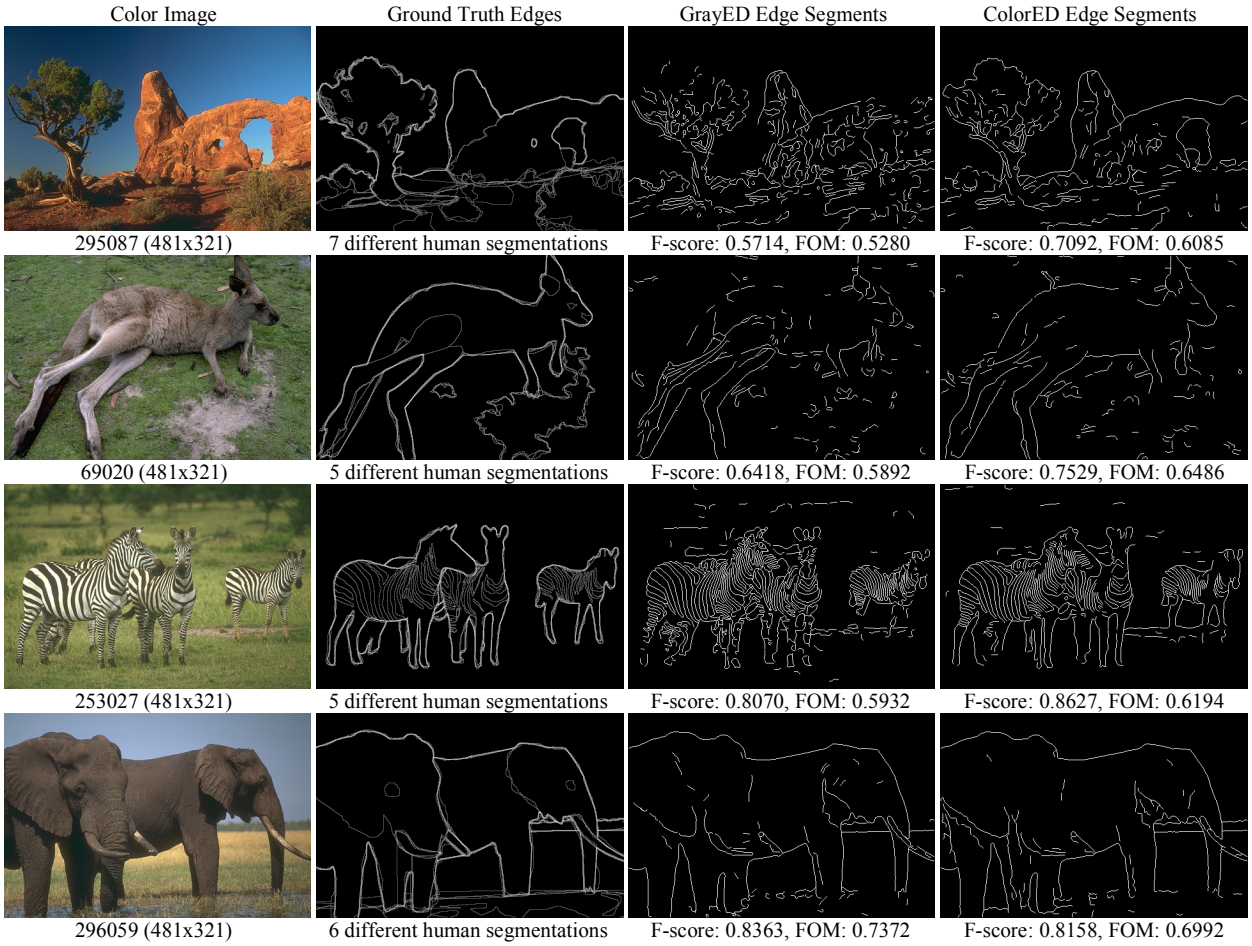


Figure 3: The best results by GrayED and ColorED for 4 images from the Berkeley Segmentation Dataset and Benchmark (BSDS) test set. The images were first smoothed by a Gaussian kernel with $\sigma = 1.5$, and the gradient threshold that gives the best result for each image is used. For color gradient computation, DiZenzo's operator [7] is used. ColorED clearly outperforms GrayED for the first three images, while GrayED produces a slightly better result for the fourth image as indicated by the F-score and Pratt's FOM values listed below each result.

mation for each image. 200 of these images are the training set and are used to tune up the parameters of a boundary detector. The remaining 100 images are the test images used to evaluate the performance of a boundary detection algorithm. The evaluation is based on comparing the boundary map produced by an algorithm with all human annotated ground truth (GT) boundaries of the image, and computing a goodness score (called the F-score) that tells how good the boundaries produced by the algorithm are compared to the GTs.

Let the boundaries returned by an algorithm for an image be A, and the ground truth boundary information be GT. Then, precision P, recall R, and F-score, which is the harmonic mean of precision and recall, are defined as follows.

$$\begin{aligned} P &= \frac{(A \cap GT)}{A} \\ R &= \frac{(A \cap GT)}{GT} \\ F - score &= \frac{(2PR)}{P+R} \end{aligned} \tag{2}$$

BSDS300 evaluation framework not only computes an F-score for each image, but it also computes a cumulative F-score over all 100 test images. Thus the performance of different boundary detection algorithms can objectively be compared against each other.

Another common metric used to grade the goodness of a boundary map A compared to its GT is the Pratt's Figure Of Merit (FOM) [35, 36], which is defined as follows:

$$FOM = \frac{\sum_{i=1}^{|A|}(1/(1+\alpha d(i)^2)}{max\{|A|,|GT|\}} \tag{3}$$

where $|A|$ and $|GT|$ are the number of boundary pixels in A and GT respectively, d(i) is the Euclidean distance of the i^{th} boundary pixel to its nearest GT pixel, and α is a scaling constant controlling the sensitivity of FOM to the differences between A and GT, usually set to 1/9 [36]. FOM takes values between 0 and 1, with 1 being a perfect match between the detected edges A and the ideal edges GT. Given a boundary map computed by an algorithm, we compare it against all of its GTs and compute an overall FOM score for the edge map.

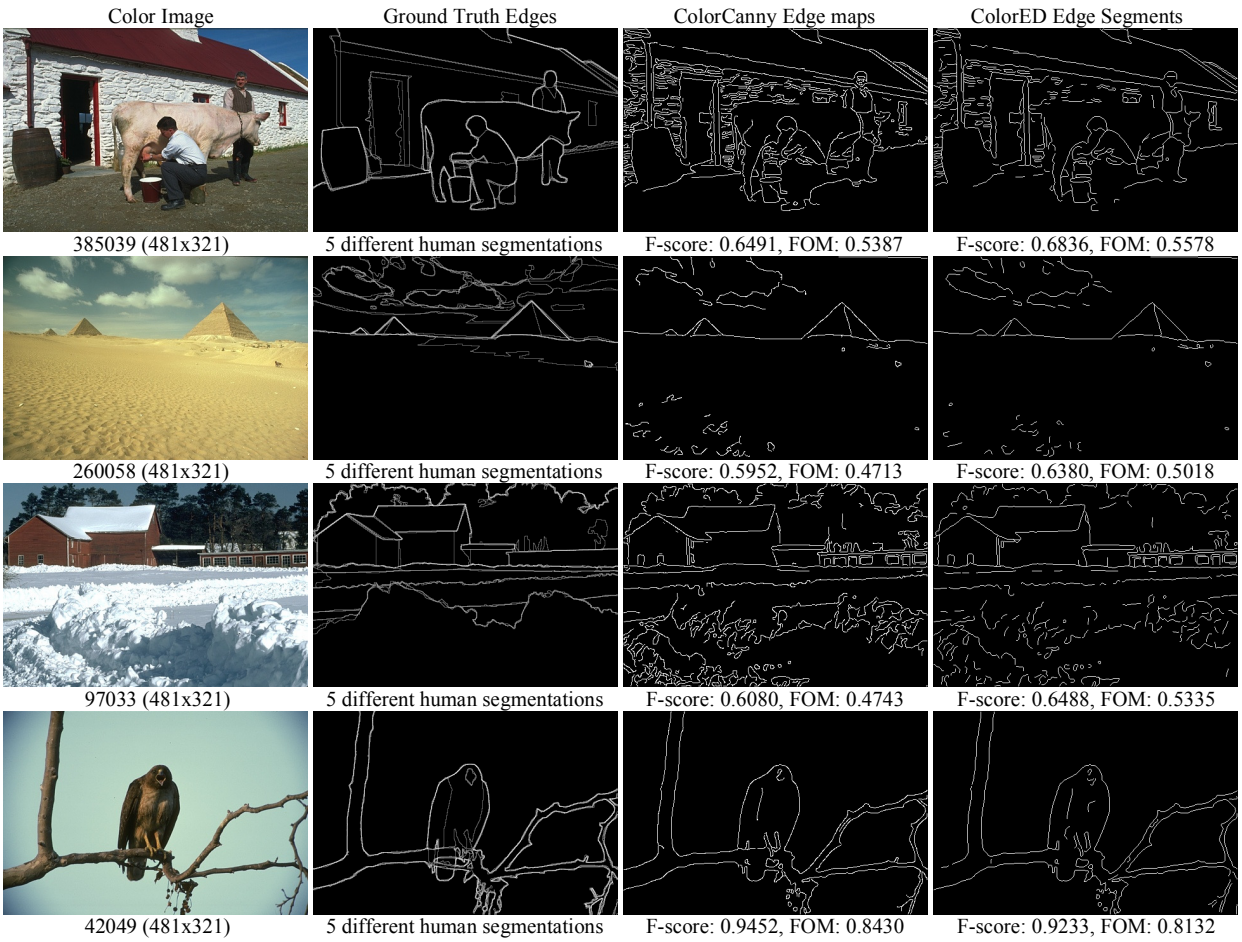


Figure 4: The best results by ColorCanny and ColorED for 4 images from the BSDS test set. The images were first smoothed by a Gaussian kernel with $\sigma = 1.5$, and the gradient threshold that gives the best result for each image is used. For color gradient computation, DiZenzo's operator [7] is used. ColorED clearly outperforms ColorCanny for the first three images, while ColorCanny produces a slightly better result for the fourth image as indicated by the F-score and Pratt's FOM values listed below each result.

5.2. ColorED vs. GrayED

We start the experiments by comparing the performance of ColorED with GrayED [26] to motivate why color edge detection is necessary and important. Fig. 3 shows the best results for GrayED and ColorED for 4 images from the BSDS test set. To obtain these edge maps, the images were first smoothed by a Gaussian kernel with $\sigma = 1.5$, and the gradient threshold that gives the best result (the edge map that maximizes the F-score against the GTs) for each image is used. The optimal threshold for each image were determined by an extensive search of the threshold space; starting with a very small threshold and increasing it until the best result is obtained. For color gradient computation, DiZenzo's operator [7] is used.

Visual analysis and quantitative performance metric values indicated by the F-score and FOM values listed below each result clearly show that ColorED greatly outperforms GrayED for the first three images, while GrayED produces a slightly better result for the fourth image. Although we only present the results for 4 images in Fig. 3, we show in section 5.5 that ColorED has a much better performance compared to GrayED for most of the images in BSDS, and a much higher cumulative F-score. In cases where GrayED produces better results, ColorED produces almost as good results, and is only slightly behind as can also be observed in the last row of Fig. 3.

5.3. ColorED vs. ColorCanny

In this section, we compare ColorCanny and ColorED since these are two different ways of obtaining an edge map after the gradient map and edge directions are calculated using a color vector gradient operator. As we described in section 2, ColorCanny uses non-maximal suppression followed by hysteresis with double thresholding (refer to algorithm 3), while ColorED uses anchor computation and smart routing with a single threshold (refer to algorithm 2). Fig. 4 shows the best results for ColorCanny and ColorED for 4 images from the BSDS test set. To obtain these edge maps, the images were first smoothed by a Gaussian kernel with $\sigma = 1.5$, and the gradient threshold that gives the best result (the edge map that maximizes the F-score against the GTs) for each image is found. For color gradient computation, DiZenzo's operator [7] is used.

Visual analysis and quantitative performance metric values listed below each result in Fig. 4 show that ColorED produces slightly better results in general, although ColorCanny may also give better results for some images as the fourth row indicates.

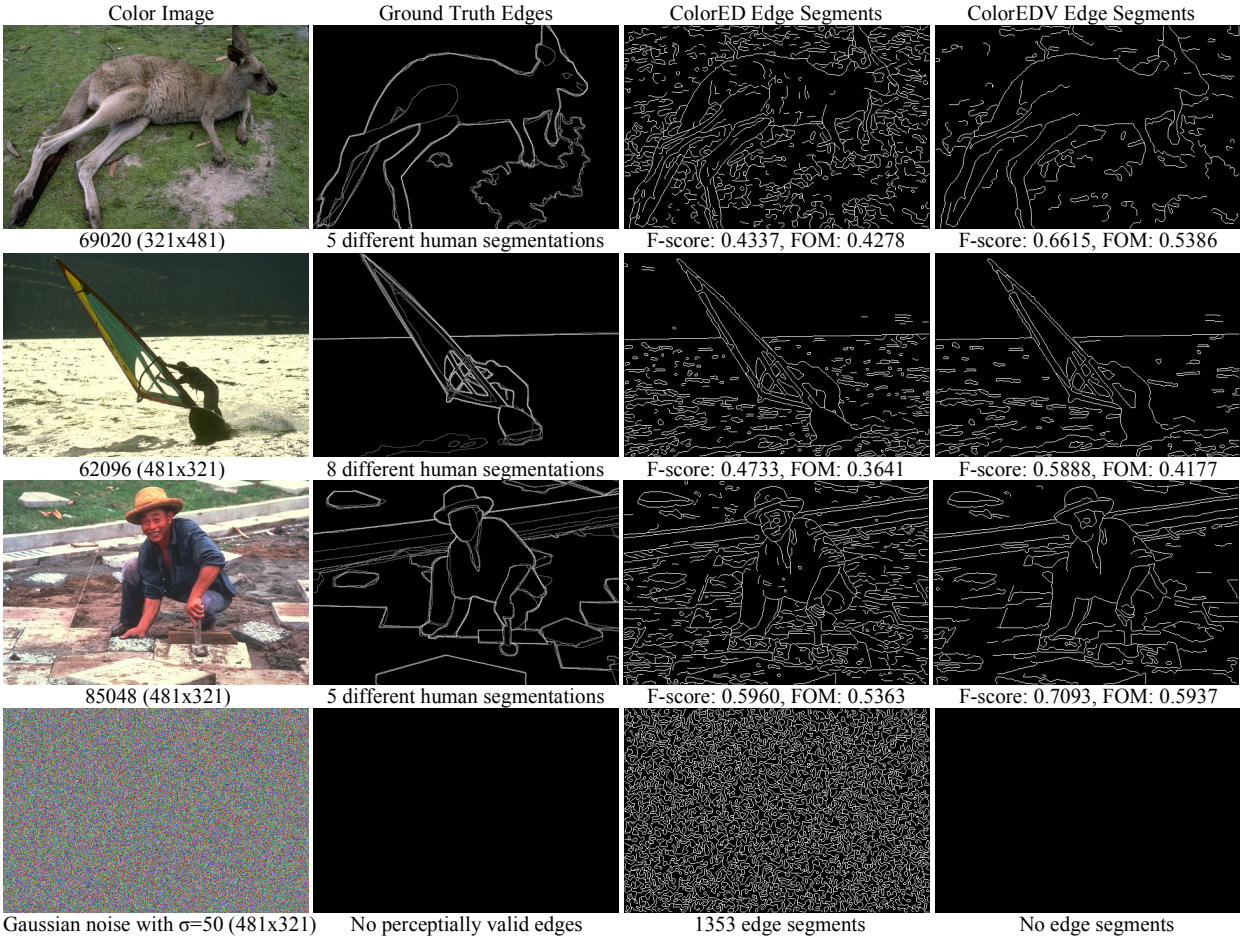


Figure 5: The edge maps produced by ColorED (third column) for 3 test images from the BSDS (the first three rows) and for a Gaussian white noise image with $\sigma = 50$ (the fourth row). The images were first smoothed by a Gaussian kernel with $\sigma = 1.5$, and ColorED results were obtained by using the same fixed gradient threshold of 36 for all images. The last column shows ColorEDV results, which were obtained by validating ColorED's edge segments. Clearly, validation eliminates many incorrectly detected edge segments improving the modal quality of the edge maps. Pay special attention to the Gaussian white noise image at the last row, where all edge segments are eliminated after validation as the theory suggests.

We will show in section 5.5 that ColorED indeed performs better in general with a higher cumulative F-score for the BSDS test images. It is also important to note that the output of ColorCanny or any other color edge detector found in the literature is a binary edge map, which usually contains low quality artifacts such as gaps between edge groups, noisy edgel formations and double edges. All of these low quality artifacts can be observed in ColorCanny's results shown in Fig. 4. However, ColorED outputs a set of edge segments each of which is a contiguous chain of pixels, which can then be used for higher level processing. Additionally, ColorED's edge segments can be fed into validation to eliminate false detections, which greatly improves ColoED's results as we next show in section 5.4, whereas, a binary edge map output by other color edge detectors cannot be validated as is. We can therefore say that the modal quality of the edge maps output by ColorED is in general better than those of ColorCanny.

5.4. ColorED vs. ColorEDV

Our next experiment is to show the importance of edge segment validation. As we described in section 4, the edge segments output by ColorED can be fed into a validation algorithm due to the Helmholtz principle to eliminate false detections.

Fig. 5 shows the edge maps produced by ColorED for 3 images from the BSDS test set (the first three rows) and for a Gaussian white noise image with $\sigma = 50$ (the fourth row). To obtain these results, the images were first smoothed by a Gaussian kernel with $\sigma = 1.5$, and ColorED results were obtained by using the same fixed gradient threshold of 36 for all images. For color gradient computation, DiZenzo's operator [7] is used. The last column shows ColorEDV results, which were obtained by validating ColorED's edge segments.

It is clear from the last column in Fig. 5 that validation eliminates many incorrectly detected edge segments greatly improving the modal quality of the edge maps as can be observed visually and verified by the quantitative evaluation metric values listed below each result. Pay special attention to the Gaussian white noise image at the last row, where all edge segments are eliminated after validation as the theory suggests. In section 5.5, we will show that edge segment validation indeed greatly improves the quality of ColorED's edge segments re-

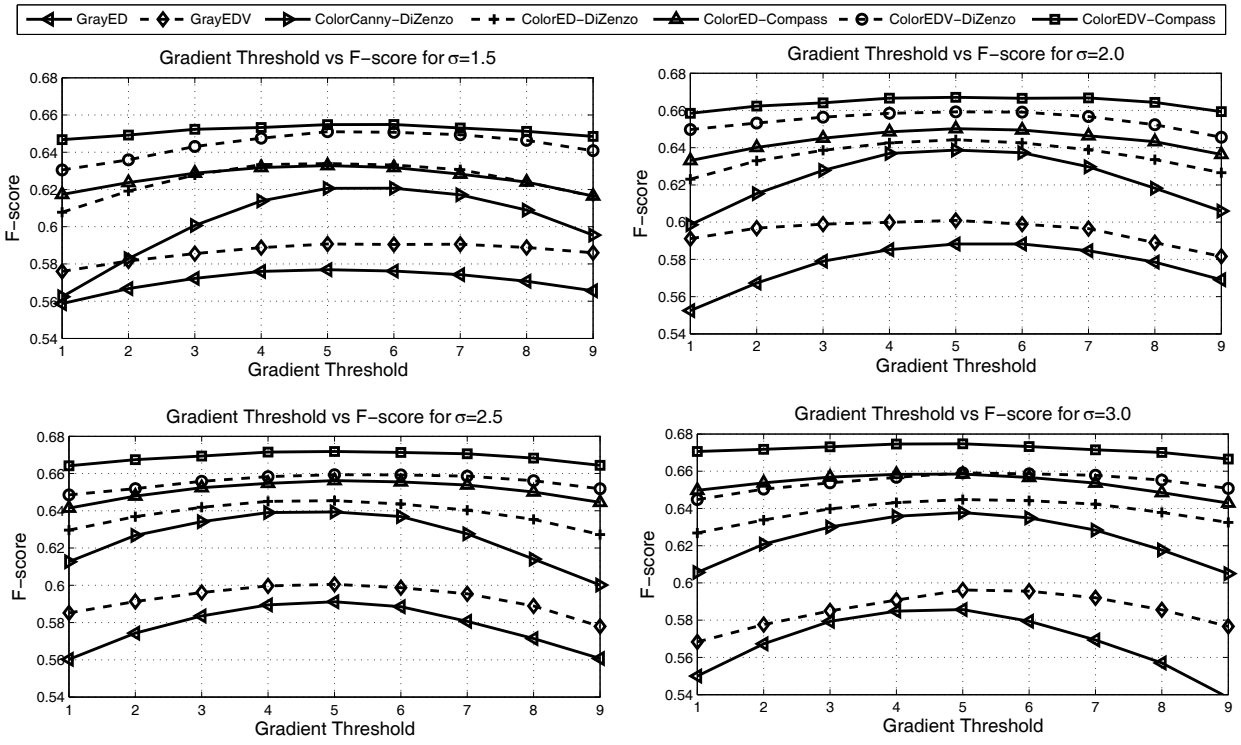


Figure 6: Threshold versus F-score curves for different edge detection algorithms and color gradient operators for Gaussian smoothing kernels with σ = 1.5, σ = 2.0, σ = 2.5, and σ = 3.0.

sulting in a much higher cumulative F-score for the BSDS test images than ColorED and ColorCanny.

5.5. Overall Comparison of All Edge Detectors

We now compare the overall performance of different edge detectors with each other within the BSDS testbed using the cumulative F-score metric as our yardstick. Recall from algorithm 2 that ColorED has two parameters that must be supplied by the user: the sigma of the Gaussian smoothing kernel, and the gradient threshold. Our evaluation methodology is to fix both of these parameters to specific values, e.g., (sigma=1.5, threshold=36), and run an edge detector with these fixed values for all 100 images in the BSDS test set. The edge map results for the 100 test images are then evaluated within the BSDS testbed, and a cumulative F-score is obtained for the particular edge detector for the specific (sigma, threshold) pair. Since we do not know what threshold value gives the highest F-score for an edge detector at a specific sigma value, the same experiment is repeated with different threshold values as follows: We start with a small threshold value, and increase it by small increments repeating the same experiment for each threshold. Thus we get many F-score values, one for each (sigma, threshold) pair. What we observe is the following: At small threshold values, the recall is high while the precision is low as expected. As the threshold increases, the recall goes down, but the precision goes up. Since F-score is the harmonic mean of precision and recall, it is very low for small threshold values, but starts increasing as the threshold is increased. At a certain threshold, the recall and the precision becomes balanced so as to maximize the F-score. Beyond this threshold, the precision keeps increasing, but the big drop in recall cancels out any gain, decreasing the cumulative F-score. Although the gradient threshold at which the F-score is maximized is different for each edge detector, they all follow a similar pattern. We should also note that ColorCanny has an additional high threshold parameter. We simply set it to twice the value of the low threshold as is the usual practice, which in fact causes ColorCanny to perform the best.

Fig. 6 shows gradient threshold versus F-score curves for different edge detection algorithms and color gradient operators for Gaussian smoothing kernels with σ = 1.5, σ = 2.0, σ = 2.5, and σ = 3.0. Each chart plots the F-score for 9 different gradient threshold values for each algorithm. In all charts, a gradient threshold of 5 denotes the threshold at which the F-score for an algorithm is maximized. For example, when the sigma is 1.5, the optimal threshold at which ColorED-DiZenzo reaches the highest F-score is 64, while the optimal threshold for ColorCanny-DiZenzo and ColorEDV-DiZenzo is 48. For each gradient threshold beyond 5 in the charts, the actual threshold is incremented by 4, and for each gradient threshold below 5, the actual threshold is decremented by 4. Thus, for ColorED-DiZenzo edge detector, the gradient thresholds 1 through 9 in the chart for σ = 1.5 correspond to actual threshold values ranging from 48 to 80 in increments of 4.

Looking at the charts, we have the following observations:

(1) Color edge detection performs much better compared to grayscale edge detection as seen by comparing the performance of GrayED, ColorED and ColorCanny.

Table 1: The best F-score values obtained by ColorEDV for different gradient operators as the Gaussian smoothing sigma increases. In the case of Compass, there is no Gaussian smoothing, and the sigma parameter is converted to the operator's radius by the formula $\lceil 3 \times \sigma \rceil$.

Gaussian	Gradient Operators					
Sigma (σ)	Prewitt-L1	Sobel-L1	Prewitt-L2	Sobel-L2	DiZenzo	Compass
1.5	0.6496	0.6481	0.6505	0.6496	0.6510	0.6548
2.0	0.6585	0.6587	0.6592	0.6590	0.6592	0.6670
2.5	0.6583	0.6581	0.6584	0.6585	0.6593	0.6718
3.0	0.6566	0.6576	0.6575	0.6587	0.6591	0.6747

(2) ColorED performs slightly better than ColorCanny. This is expected as the modal quality of the edge segments output by ColorED are of higher quality in general compared to the binary edge maps output by ColorCanny as was also illustrated in Fig. 4.

(3) Edge segment validation greatly improves the modal quality of both GrayED's and ColorED's edge segments. This is also expected as the edge segment validation removes many incorrectly detected edge segment pieces leaving only statistically valid edge segments as defined by the Helmholtz principle, which greatly improves the precision of the edge segments as was also illustrated in Fig. 3.

(4) Of the gradient operators tested, Compass gives out the best results with a maximum F-score value of 0.6747 for a radius of $\lceil 3 \times \sigma = 3 \rceil = 9$ pixels. For a single scale color edge detector, this is a very good value compared to the results obtained by other complex contour detectors in the literature [38, 39, 40].

Table 1 shows the best F-score values obtained by ColorEDV for different gradient operators as the Gaussian smoothing sigma increases. In the case of Compass, there is no Gaussian smoothing, and the sigma parameter is converted to the operator's radius by the formula $\lceil 3 \times \sigma \rceil$. Looking at the results we see that there is not much of a difference between vector Prewitt/Sobel L1 or L2 norms and DiZenzo, except that DiZenzo gives slightly better results especially for bigger sigma values. Compass on the other hand, greatly outperforms the other gradient operators.

5.6. Running Time Performance

Table 2 shows the dissection of the average running time for ColorEDV and ColorCanny on the 100 BSDS test set images (each image is either 481x321 or 321x481 pixels) for Gaussian smoothing $\sigma = 1.5$. The running times were obtained on a labtop with an Intel core i7-2670QM CPU running at 2.2 GHz. After RGB2Lab conversion, which takes 20.5 ms, the rest of ColorEDV takes just 31.5 ms with most of the time (22.5 ms) being spent on gradient computation by DiZenzo. ColorCanny also takes a similar amount of time for a total of 31 ms. It is clear from the results that what makes color edge detection costly is the color vector gradient computation. Table 2 also shows that if the traditional vector Prewitt/Sobel gradient operator described in section 3.1 is used, then the gradient computation takes only 9 ms, which would make the total execution time for ColorEDV 18.2 ms, and for ColorCanny 17.7 ms.

Table 2: Dissection of the average running time for ColorEDV and ColorCanny on the 100 BSDS test set images (481x321 or 321x481 pixels each) for Gaussian smoothing $\sigma = 1.5$.

Step	Time (ms)
RGB2Lab	20.5
Gaussian smoothing	2.0
Gradient Computation (Prewitt/Sobel)	9.0
Gradient Computation (DiZenzo)	22.3
Anchor Computation and Smart Routing	3.0
Validation	4.2
Total time for ColorEDV (with DiZenzo)	31.5
Total time for ColorCanny (with DiZenzo)	31.0

Table 3: Average running time in seconds for the Compass operator on the 100 BSDS test set images (481x321 or 321x481 pixels each). For Compass, there is no Gaussian smoothing, and the Gaussian smoothing sigma is converted to the radius of the operator by the formula $radius = \lceil 3 * \sigma \rceil$.

Sigma	Radius	Compass (sec)
1.0	3	22.5
1.5	5	75.2
2.0	6	109.3
2.5	8	142.6
3.0	9	158.4
3.5	11	170.5
4.0	12	177.8

Even by including the RGB2Lab convertion time of 20.5 ms, ColorEDV would then take just 38.7 ms, which is almost real-time for 481x321 color images. Although ColorEDV performs better with DiZenzo in general, its performance with vector Prewitt/Sobel is also good enough for all practical purposes. Therefore, if the speed is important, then vector Prewitt/Sobel should be used with ColorEDV instead of DiZenzo.

Finally, Table 3 shows the average running time in seconds for the Compass operator on the 100 BSDS test set images. For Compass, there is no Gaussian smoothing, and the Gaussian smoothing sigma is converted to the radius of the operator by the formula $radius = \lceil 3 * \sigma \rceil$. As the table shows, even for very small radius sizes, e.g., radius=3, the Compass operator takes about 1000 times more time than DiZenzo. As the radius of the operator increases, the running time also increases drastically. Although Compass gives the best results with ColorEDV on the BSDS benchmark, it is very slow to be of any practical use.

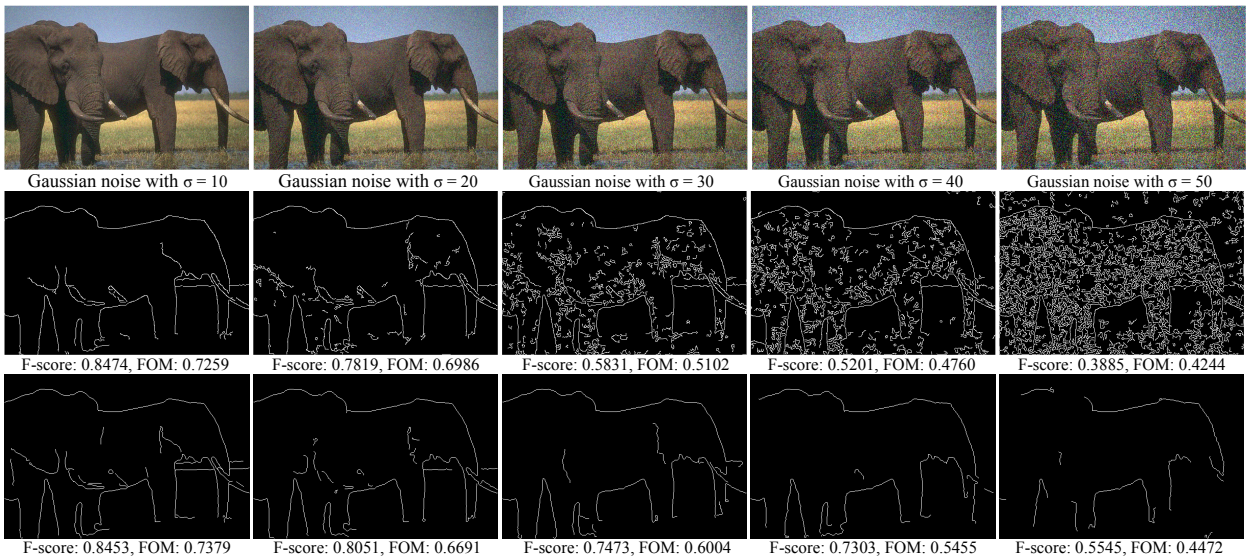


Figure 7: First row: Image 296059 from BSDS test set with increasing amount of Gaussian noise. Second row: ColorCanny results with lowthresh=48, highthresh=96 after Gaussian smoothing with $\sigma = 1.5$. Third row: CEDV results with thresh=48 after Gaussian smoothing with $\sigma = 1.5$.

5.7. Performance of the Algorithms in Noisy Images

In this section, we test the performance of the proposed algorithms on noisy images. Fig. 7 shows image 296059 from the BSDS test set with increasing amounts of Gaussian noise added. The second row of Fig. 7 shows the edge map results by ColorCanny in the noisy images with lowthresh=48 and highthresh=96, after the images have been smoothed by a Gaussian kernel with *sigma* = 1.5. We observe that at low level of noise ColorCanny still produces good edge maps. But as the level of noise increases, ColorCanny starts producing too many false detections especially around textured regions. The third row of Fig. 7 shows the edge map results by ColorEDV on the noisy images using the same threshold value of 48. We observe that at low levels of noise, ColorEDV's results are very good. As the amount of noise increases, ColorEDV's performance deteriorates but it is still able to extract the major boundaries of the objects in the image producing much better results compared to ColorCanny. The decline in ColorEDV's performance at high noise levels has to do with the smart routing step: At low noise levels, the initial Gaussian smoothing takes care of the image noise enabling smart routing to create long edge segments over the actual boundaries of the image, which leads to high quality edge segments. But at high levels of noise, the edge directions become unreliable deviating smart routing from the actual boundaries of the objects and leading it towards the noisy parts of the image, which causes short and noisy edge segment formations that get eliminated during validation. We should still note that regardless of how the edge segments are created by smart routing, ColorEDV never returns perceptually meaningless edge segments as defined by the Helmholtz principle, which is very important.

Finally, Fig 8 shows the cumulative F-score achieved by ColorCanny and ColorEDV on the BSDS test set as the amount of Gaussian noise increases up to $\sigma = 50$. A fixed gradient threshold value of 48 was used for ColorCanny and ColorEDV to obtain these results as these are the thresholds for which the algorithms produce the best cumulative F-score values. We see from the figure that as the noise level increases, the performance of both algorithms deteriorate quickly although ColorEDV's performance is better than that of ColorCanny.

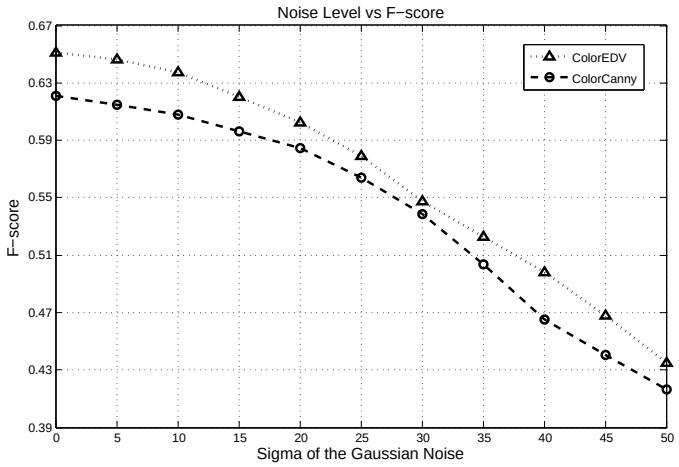


Figure 8: Cumulative F-score of ColorCanny and ColorEDV on the BSDS test set as the amount of Gaussian noise increases up to $\sigma = 50$.

6. Conclusions

We present a color edge segment detector, named ColorED, by extending our recently proposed grayscale edge segment detector, Edge Drawing (ED). ColorED works by converting the input color image in RGB format to CIE Lab colorspace, runs a color vector gradient operator to compute the gradient map and edge directions, and extracts the edge segments by anchor computation followed by smart routing. Each edge segment is a contiguous chain of pixels that can be used for such higher level

processing jobs as line, arc, corner, circle, ellipse, or similar shape detection applications. Edge segments can also be passed through a validation method due to the Helmholtz principle to eliminate false detections. Quantitative experiments performed within the precision-recall framework of the famous Berkeley Segmentation Dataset and Benchmark (BSDS300) show that ColorED with Validation (ColorEDV) produces the best results. Of the color vector gradients tested, Compass gives out the best results but is extremely slow to be any practical use. For high speed applications, DiZenzo's operator or the traditional vector Prewitt/Sobel operators can be used. Interested readers can download the code for the proposed algorithms from ColorED's Web site [41], and repeat the same experiments presented in this paper. Our future goal is to combine ColorEDV's results at multiple scales similar to what we did for grayscale images in [42, 43], and come up with a high-speed contour detector for color images that can compete with state of the art contour detectors on the BSDS benchmark [38, 39, 40].

References

[1] D. Marr, E. Hildreth, Theory of Edge Detection, Proceedings of the Royal Society of London, Biological Sciences, 207(1167) (1980), pp. 187-217.

[2] J. Canny, A computational approach to edge detection, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6) (1986), pp. 679-698.

[3] V.S. Nalwa, T.O. Binford, On detecting edges, IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6) (1986), pp. 699-714.

[4] E. Deriche, Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector, International Journal of Computer Vision, 1(2) (1987), pp. 167-187.

[5] S.Y. Zhu, K.N. Plataniotis, A.N. Venetsanopoulos, Comprehensive analysis of edge detection is color image processing, Opt. Eng., 38(4) (1999), pp. 612-625.

[6] A. Koschan, M. Abidi, Detection and Classification of Edges in Color Images, Signal Processing Managize, Special Issue on Color Image Processing, 22(1) (2005), pp. 64-73.

[7] S. DiZenzo, A Note on the Gradient of a Multi-Image, Computer Vision, Graphics, and Image Processing, 33 (1986), pp. 116-125.

[8] R.C. Gonzales, R.E. Woods, Digital Image Processing, Pearson Prentice-Hall, (2008), pp. 447-451.

[9] A. Cumani, Edge detection in multispectral images, Computer Vision, Graphics and Image Processing: Graphical Models in Image Processing, 53(1) (1991), pp. 40-51.

[10] P.W. Trahanias, A.N. Venetsanopoulos, Color edge detectors based on multivariate ordering, Proc. SPIE Visual Communications Image Processing, 1818 (1992), pp. 1396-1407.

[11] P.W. Trahanias, A.N. Venetsanopoulos, Color edge detectors based using vector order statistics, IEEE Transactions on Image Processing, 2(2) (1993), pp. 259-264.

[12] P.W. Trahanias, A.N. Venetsanopoulos, Vector order statistics operators as color edge detectors, IEEE Transactions on Systems, Man. Cybernetics, 26(1) (1996), pp. 135-143.

[13] M. Shaikh, A comparative study of color edge detection techniques, CS231A Winter-1314 Project Report, Stanford University, (2014).

[14] M. Ruzon, C. Tomasi, Color Edge Detection with the Compass Operator, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), (1999), pp. 160-166.

[15] M. Ruzon, C. Tomasi, Edge, junction, and corner detection using color distributions, IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(11) (2001), pp. 1281-1295.

[16] Compass Web site, `http://robotics.stanford.edu/~ruzon/compass`, Accessed: September, 2015.

[17] Y. Rubner, C. Tomasi, L.J. Guibas. A metric for distributions with applications to image databases, International Conference on Computer Vision (ICCV), (1998), pp. 5966.

[18] Earth Mover's Distance Web site, `http://www.ariel.ac.il/sites/ofirpele/fastemd/code`, Accessed: September, 2015.

[19] X. Gong, J. Liu, A Daisy-like Compass Operator, International Conference on Image Processing (ICIP), (2011), pp. 1041-1044.

[20] A. Saez, C. Serrano, B. Acha, Evaluation of Perceptual Color Edge Detection Algorithms, Conference on Colour in Graphics, Imaging, and Vision (CGIV), (2010), pp. 222-227.

[21] A. Saez, C.S. Mendoze, B. Acha, C. Serrano, Development and evaluation of perceptually adapted colour gradients, IET Image Processing, 7(4) (2013), pp. 355-363.

[22] K. McLaren, The development of the CIE 1976 (L*a*b*) uniform colour-space and colour-difference formula, Journal of the Society of Dyers and Colourists, 92(9) (1976), pp. 338-341.

[23] CIE, Improvement to industrial colour-difference evaluation, CIE Publication 142, Central Bureau of the CIE, (2001).

[24] G. Sharma, W. Wu, E.N. Dala, The CIEDE2000 Color-Difference Formula: Implementation Notes, Supplementary Test Data, and Mathematical Observations, Color research and application, 30(1), (2005), pp. 21-30.

[25] R. Moreno, M.A. Garcia, D. Puig, C. Julia, Robust Color Edge Detection Through Tensor Voting, International Conference on Image Processing (ICIP), (2009), pp. 2153-2156.

[26] C. Topal, C. Akinlar, Edge Drawing: A Combined Real-Time Edge and Segment Detector, Journal of Visual Communication and Image Representation, 23(6) (2012), pp. 862-872.

[27] C. Akinlar, C. Topal, EDLines: a real-time line segment detector with a false detection control, Pattern Recognition Letters, 32(13) (2011), pp. 1633-1642.

[28] C. Akinlar, C. Topal, EDCircles: A real-time circle detector with a false detection control, Pattern Recognition, 46(3) (2013), pp. 725-740.

[29] C. Topal, B. Benligiray, K. Ozkan, C. Akinlar, A Robust CSS Detector Based on Turning Angle Curvature, IEEE Int. Conference on Acoustics, and Signal Processing (ICASSP), (2013), pp. 1444-1448.

[30] C. Akinlar, C. Topal, EDPF: A Real-time Parameter-free Edge Segment Detector with a False Detection Control, International Journal of Pattern Recognition and Artificial Intelligence, 26(1) (2012).

[31] A. Desolneux, L. Moisan, J.M. Morel, Edge Detection by Helmholtz Principle, Journal of Mathematical Imaging and Vision, 14(3) (2001), pp. 271-284.

[32] E. Meinhardt-Llopis, Edge Detection by Selection of Pieces of Level Lines, International Conference on Image Processing (ICIP), (2008), pp. 613-616.

[33] A. Desolneux, L. Moisan, J.M. Morel, Gestalt theory and Computer Vision, Book chapter in Seeing, Thinking and Knowing, Kluwer Academic Publishers, (2004), pp. 71-101.

[34] A. Desolneux, L. Moisan, J.M. Morel, From Gestalt Theory to Image Analysis: A Probabilistic Approach, Springer, (2008).

[35] W.K. Pratt, Digital Image Processing, Wiley-Interscience, New York, (2007).

[36] I. Abdou, W.K. Pratt, Quantitative design and evaluation of enhancement/thresholding edge detetors, Proc. IEEE, 67 (1979), pp. 753-763.

[37] D. Martin, C. Fowlkes, D. Tal, J. Malik, A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics, IEEE International Conference on Computer Vision (ICCV), (2001), pp. 416-423.

[38] The Berkeley Segmentation Dataset and Benchmark Web site, `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds`, Accessed: July, 2015.

[39] P. Arbelaez, M. Maire, C. Fowlkes, J. Malik, Contour Detection and Hierarchical Image Segmentation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(5) (2011), pp. 898-916.

[40] X. Ren, L. Bo, Discriminatively Trained Sparse Code Gradients for Contour Detection, Advances in Neural Information Processing Systems (NIPS), (2012).

[41] Color Edge Drawing (ColorED) Web site, `http://ceng.anadolu.edu.tr/cv/ColorED`, Accessed: September, 2015.

[42] C. Akinlar, C. Topal, EDContours: High Speed Contour Detection Using Scale-Space EDPF, International Symposium on Multimedia (ISM), (2012), pp. 153-156.

[43] EDContours Web site, `http://ceng.anadolu.edu.tr/cv/EDContours`, Accessed: September, 2015.