A Comparative Study of Color Edge Detection Techniques

Masood Shaikh, Department of Electrical Engineering, Stanford University

Abstract—Edge detection has attracted the attention of many researchers and is one of the most important areas in low level computer vision. In recent years, the focus of edge detection has shifted from grayscale single component images to multicomponent color images that utilize the inter-spectral correlation of the neighboring color samples to eliminate color artifacts and increase the accuracy of edge detection process. This project presents the study of color edge detection based on vector order statistics operators. Variations are introduced in the vector order statistics color edge operator to improve noise performance and we demonstrate their ability to attenuate noise with added algorithm complexity. We present a performance evaluation framework to assess the quality of color edge detectors and compare it with the Canny edge detector which is the optimal edge detector used for grayscale images. The edge detectors are evaluated with subjective and objective tests using statistical indices by comparing it with human ground truth images extracted from the BSDS300 dataset.

Index Terms—color edge detection, computer vision, vector techniques, performance evaluation

1 Introduction

The capability to identify and segment edges along with texture regions in an image is critical to the analysis of natural scenes. It has been known for a long time that edges contain most of the information in an image while being represented far more compactly than the image itself. For this reason edge detection is perhaps the most basic (and therefore most widely studied) problem in computer vision. It is theorized [1] to be the first step in *contour* detection and image *segmentation*, the partitioning of an image into meaningful regions, which can then be analyzed independently to recognize different parts of a scene, and hopefully the context of the scene as a whole. The subject of color image processing in computer vision has gained increasing attention recently because color images convey more information about objects in a scene than gray-scale images and this information can be used to further refine the performance of an imaging system. The multicomponent nature of a color image, however, adds considerable complexity to the processing system. One of the challenges facing color image processing is to extract the additional color information without incurring large complexity in the system.

In monochrome images, an edge usually corresponds to object boundaries or change in physical properties of the image such as (illumination) luminance. In this sense, a color image (multi-spectral) contains more detailed edge information. Moreover, edge detection with monochromatic images may fail in certain applications since no edges will be detected in monochrome images if neighboring objects have different hue and saturation while their intensity values are same. Psychological research has concluded that color plays an important role in decidin object and scene contours in human visual system [2]. Since the capability to distinguish between different objects is crucial for applications such as object recognition image segmentation and scene understanding, the additional boundary information provided by color is of paramount importance. Color edge detection also outperforms monochrome edge detection in low contrast images [3]. There is thus a strong motivation to develop efficient color edge detectors that provide high quality edge maps. Numerous approaches of different complexities to color edge detection have been proposed. In this project we present the study of color edge detectors based on vector order statistics class [4]. It is important to identify their strengths and weaknesses in choosing the best color edge detector for an application. The major performance issues concerning edge detectors are their ability to extract edges accurately, their robustness to noise, and their computational efficiency. To provide a fair assessment, it is necessary to have a set of effective performance evaluation frameowork. Though numerous evaluation methods for edge detection have been proposed, there has not been any standardized objective or subjective evaluation method. While objective evaluation can provide analytical data for comparison purposes, it is not sufficient to represent the complexity of the human visual systems. In most computer vision applications, human evaluation is the final step. Hence in this project, both types of evaluation methods are utilized for comparing various edge detectors.

We start with an overview of techniques used in monochrome edge detection in Section 2. The edge detector illustrated and described in this section is the Canny Edge Detector [5] which is the optimal edge detector and the most widely used for monochrome images. Section 3 gives an overview of color edge detection techniques, where early approaches extended from monochrome

edge detection, as well as more recent vector space approaches and difference vector approaches [6] are addressed. Color edge detection based on vector-order statistics method [4] and a family of edge detectors based on this class is studied in detail in Section 4. Section 5 gives details of quantative performance evaluation procedure using BSDS300 [7] dataset for statistical evaluation of edge detectors. The evaluation results from both objective and subjective tests are listed in Section 6 followed by our conclusions in Section 7.

2 MONOCHROME EDGE DETECTION

In monochrome grayscale images, edges are commonly defined as sharp intensity discontinuities, as physical edges often coincide with places of strong illumination and reflection changes. Hence the most common method of edge detection in gray-scale images use some form of differential operators that detect intensity discontinuities. Another group of gradient-based edge operators locates an edge by finding the zero crossing of the second derivative of the image intensity function. The zero crossing carries the information about the local extremum of the first derivative and indicates a point of rapid change of the intensity function. By detecting zero crossings, edges of one pixel thickness can be obtained, which is hardly possible with the differential based methods.

2.1 Canny Edge Detection Algorithm

The Canny Edge Detector proposed by John F. Canny in 1986 [5] is one of the most widely used image processing tools in computer vision for detecting edges in monochrome images. The algorithm runs in 5 separate steps:

1. Smoothing

Images taken from a camera will contain some amount of noise. To prevent that noise to be mistaken for edges, noise must be reduced. Therefore the image is first smoothed by applying a Gaussian filter. The kernel of a Gaussian filter with a standard deviation of σ = 1.4 is shown in Equation (1)

$$g = \frac{1}{159} \times \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \quad (1)$$

2. Finding Gradients

The Canny algorithm basically finds edges where the grayscale intensity of the image changes the most. These areas are found by determining gradients of the image. The gradient of the image f(x,y) can be computed by convolving it with the first derivative of the the Gaussian filter used in step 1 in x and y directions as given by equation (2) and (3)

$$f_x(x,y) = f(x,y) * \left(\frac{-x}{\sigma^2}\right) \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \quad (2)$$

$$f_y(x,y) = f(x,y) * \left(\frac{-y}{\sigma^2}\right) \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \quad (3)$$

The gradient magnitudes (also known as the edge strengths) can then be determined as a Euclidean distance measure as shown in Equation (4). It is sometimes simplified by applying Manhattan distance measure as shown in Equation (5) to reduce the computational complexity. The Euclidean distance measure has been used in our implementation.

$$|f| = \sqrt{f_x^{\,2}(x,y) + f_y^{\,2}(x,y)} \quad (4)$$

$$|f| = |f_x(x,y)| + |f_y(x,y)| \quad (5)$$

Image of the gradient magnitude often indicate the edges quite clearly. However, the edges are typically broad and thus, do not indicate exactly the orientation of the edge. To determine this, the direction of the edges must be computed as shown in Equation (6).

$$\Theta = \arctan \frac{|f_y(x,y)|}{|f_x(x,y)|} \quad (6)$$

3. Non-Maxima Suppression

The purpose of this step is to convert the "blurred" edges in the image of the gradient magnitudes to "sharp" edges. Basically this is done by preserving all local maxima in the gradient image, and deleting everything else. The algorithm for each pixel in the gradient image is:

a. Round the gradient direction to nearest 45 degree corresponding to the use of 8-connected neighborhood.
b. Compare the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient direction, i.e. if the gradient direction is north (Theta = 90 degree), compare with pixels in north and south.
c. If the edge strength of the current pixel is largest; preserve the value of the edge strength. If not, suppress (i.e. remove) the value.

4. Min-Max Thresholding

The edge-pixels remaining after the non-maximum suppression step are (still) marked with their strength pixel by pixel. Many of these will probably be true edges in the image, but some may be caused by noise or color variations for instance due to rough surfaces. The simplest way to discern between these would be to use a threshold, so that only edges stronger that a certain value would be preserved. The Canny edge detection algorithm uses minmax thresholding. Edge pixels stronger than the max threshold are marked as strong; edge pixels weaker than the min threshold are suppressed and edge pixels between the two thresholds are marked as weak.

5. Hysteresis Edge Tracking

Strong edges are interpreted as "certain edges", and can immediately be included in the final edge image. Weak edges are included if and only if they are connected to strong edges. The logic is of course that noise and other small variations are unlikely to result in a strong edge (with proper adjustment of the threshold levels). Thus strong edges will (almost) only be due to true edges in the original image. The weak edges can either be due to true edges or noise/color variations. The latter type will probably be distributed independently of edges on the entire image, and thus only a small amount will be located adjacent to strong edges. Weak edges due to true edges are much more likely to be connected directly to strong edges. Edge tracking is implemented by grouping the edge pixels into a group of 8-connected neighbourhood. The group that contains atleast one strong edge pixel is preserved, while other groups are suppressed.

Figure 1 shows the output of various stages of the Canny edge detection algorithm applied on the monochrome (luminance) component of a color image taken from the BSDS300 dataset.

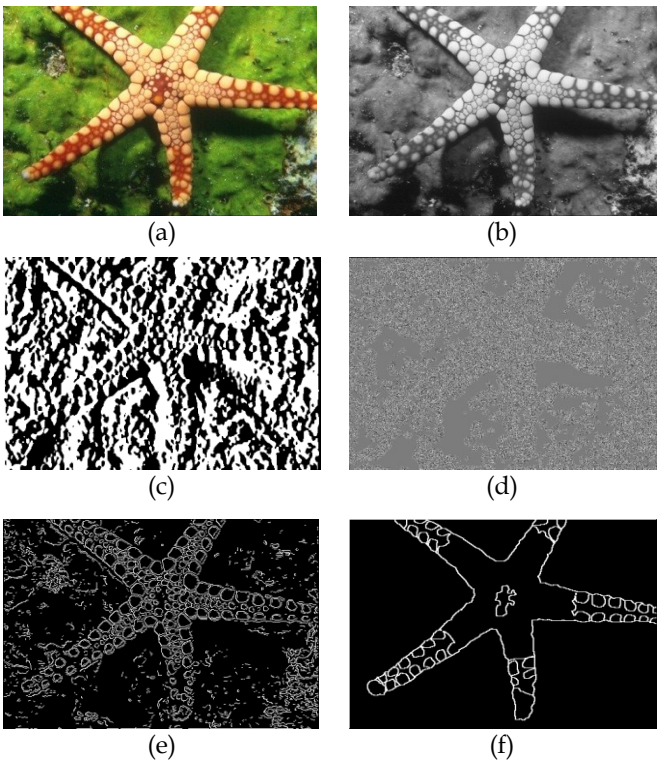


Fig. 1. (a) Original image 12003.jpg (481x321) from BSDS300. (b) Gray-scale Gaussian filtered image (c) Gradient of the Gaussian filter output. (d)Norm of the Gradient. (e) After non-maxima suppression and hysteresis tracking (f) Ground truth image.

3 Overview of Color Edge Detection Techniques

3.1 Techniques extended from monochrome edge detection

In a monochrome image, an edge is defined as an intensity discontinuity. In the case of color images, the additional variation in color must also be considered. Early approaches to color edge detection are extensions of monochrome edge detection. These techniques are applied to the three color channels independently and then the results are combined using certain logical operation. Several standard techniques can be applied in this way [8]. One of the representative classes of edge detection is the Sobel operator. It can be realized by convolving the image with the following two convolution masks:

$$g_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad g_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

These two masks are applied to each color channel independently and the sum of the squared convolution gives an estimated gradient in each channel. A pixel is regarded as an edge point if the maximum of the gradient magnitude values in the three channels exceeds a predetermined threshold.

Another operator that can also be used in a similar fashion is the eight-neighbor Laplacian operator [8]. The Laplacian convolution mask is defined as follows:

$$g = \begin{bmatrix} 10 & 22 & 10 \\ 22 & 128 & 22 \\ 10 & 22 & 10 \end{bmatrix} \tag{7}$$

Again, the Laplacian mask is applied to the three color channels independently and edge points are located by thresholding the maximum gradient magnitude.

Another group of edge detectors commonly used in monochrome edge detection is based on second derivative operators, and they can also be extended to color edge detection in the same way. A second derivative method can be implemented based on the preceding operator. The Mexican hat operator uses convolution masks generated based on the negative Laplacian derivative of the Gaussian distribution:

$$-\nabla^2 f(x,y) = \frac{x^2+y^2-2\sigma^2}{2\pi\sigma^6} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{8}$$

Edge points are located if zero-crossing occur in any one color channel.

One common problem with the preceding approaches is that they failed to take into account the correlation among the color channels, and as a result, they are not able to extract certain crucial information conveyed by color. For example, they tend to miss edges that have the same strength but in opposite direction in two of their color components. Consequently, the approach to treat the color image as vector space has been proposed.

3.2 Vector Space Approach

Various approaches proposed consider the problem of color edge detection in vector space. Color images can be viewed as a 2-D three-channel vector field [9], which can be characterize by a discrete integer function **f**(x,y). The value of this function at each point is defined by a 3-D

vector in a given color space. In the RGB color space, the function can be written as **f**(x,y) = (*R*(x,y), *G*(x,y), *B*(x,y)), where (x,y) refers to the spatial dimensions in the 2-D plane. Most existing edge detection algorithms use either first or second difference between neighboring pixels for edge detection. A significant change gives rise to a peak in the first derivative and zero-crossing in the second difference, both of which can be identified fairly easily. Some of these operators are considered in the following.

3.2.1 Vector Gradient Operators

The vector gradient operator employs the concept of a gradient operator [10], except that instead of scalar space the operator operates in a 2-D three-channel color vector space. There are several ways of implementing the vector gradient operator. One simple approach is to employ a 3 x 3 window centered on each pixel and then obtain eight distance values (D_1, D2,..., *D8*) by computing the Euclidean distance between the center vector and its eight neighboring vectors. The *vector gradient* Γ is then chosen as

$$\Gamma = \max D_k : k = 2,...8 \tag{9}$$

Another approach called *vector directional gradient* employs directional operators [11]. Let the image be a vector function **f**(x,y) = (*R*(x,y),*G*(x,y),*B*(x,y)), and let **r**, **g**, and **b** be the unit vectors along the R, G, and B axes, respectively. The horizontal and vertical directional operators can be defined as:

$$\mathbf{u} = \frac{\partial R}{\partial x}\mathbf{r} + \frac{\partial G}{\partial x}\mathbf{g} + \frac{\partial B}{\partial x}\mathbf{b} \tag{10}$$

$$\mathbf{v} = \frac{\partial R}{\partial y}\mathbf{r} + \frac{\partial G}{\partial y}\mathbf{g} + \frac{\partial B}{\partial y}\mathbf{b} \tag{11}$$

$$g_{xx} = \mathbf{u}\cdot\mathbf{u} = \left|\frac{\partial R}{\partial x}\right|^2 + \left|\frac{\partial G}{\partial x}\right|^2 + \left|\frac{\partial B}{\partial x}\right|^2 \tag{12}$$

$$g_{yy} = \mathbf{v}\cdot\mathbf{v} = \left|\frac{\partial R}{\partial y}\right|^2 + \left|\frac{\partial G}{\partial y}\right|^2 + \left|\frac{\partial B}{\partial y}\right|^2 \tag{13}$$

$$g_{xy} = \frac{\partial R}{\partial x}\frac{\partial R}{\partial y} + \frac{\partial G}{\partial x}\frac{\partial G}{\partial y} + \frac{\partial B}{\partial x}\frac{\partial B}{\partial y} \tag{14}$$

Then the maximum rate of change of **f** and the direction of the maximum contrast can be calculated as:

$$\theta = \frac{1}{2}\arctan\frac{2g_{xy}}{g_{xx} - g_{yy}} \tag{15}$$

$$F(\theta) = \frac{1}{2}\left[\left(g_{xx} + g_{yy}\right) + \cos 2\theta\left(g_{xx} - g_{yy}\right) + 2g_{xy}\sin\theta\right] \tag{16}$$

Edges can then be obtained by thresholding $[F(\theta)]^{1/2}$. This approach of vector gradient operator was further extended by Ruzon and Tomasi [12] in their compass operator, where it was assumed that edges occur when there are local statistical differences in the distribution of color image samples. To locate edges, a circular operation mask, called a compass, which measures the difference between the distributions of the pixels between two halves of a circular window, is utilized. The orientation producing the maximum difference is the direction of the edge, and the difference between the distributions yields a measure of the edge strength.

Unlike the gradient operator extended from the mentioned monochrome edge detection, the vector gradient operator can extract more color information from the image because it considers the vector nature of the color image. On the other hand, the vector gradient operator is very sensitive to small texture variations. This may be undesirable in some cases since it can cause confusion in identifying the real objects. The operator is also sensitive to Gaussian and impulse noise.

3.3 Difference Vector Operators

The class of *difference vector operators* [6] can be viewed as first derivative like operators. This group of operators is extremely effective from the point of view of the computational aspects. In this approach, each pixel represents a vector in the RGB color space, and a gradient is obtained in each of the four possible directions (0, 45, 90, and 135 degree) by applying convolution kernels to the pixel window. Then a threshold can be applied to the maximum gradient vector to locate edges. The gradients are defined as:

$$\left|\nabla f\right|_{0\deg} = \left\|\mathbf{Y}_{0\deg} - \mathbf{X}_{0\deg}\right\| \tag{17}$$

$$\left|\nabla f\right|_{90\deg} = \left\|\mathbf{Y}_{90\deg} - \mathbf{X}_{90\deg}\right\| \tag{18}$$

$$\left|\nabla f\right|_{45\deg} = \left\|\mathbf{Y}_{45\deg} - \mathbf{X}_{45\deg}\right\| \tag{19}$$

$$\left|\nabla f\right|_{135\deg} = \left\|\mathbf{Y}_{135\deg} - \mathbf{X}_{135\deg}\right\| \tag{20}$$

$$DV = \max\left(\left|\nabla f\right|_{0\deg}, \left|\nabla f\right|_{90\deg}, \left|\nabla f\right|_{45\deg}, \left|\nabla f\right|_{135\deg}\right) \tag{21}$$

where $\|\cdot\|$ denotes the L_2 norm, and **X** and **Y** are three dimensional vectors used as convolution kernels. The variations in the definitions of these convolution kernels give rise to a number of operators.

The basic operator of this group employs a 3 x 3 window involving a center pixel and eight neighboring pixels. Let each pixel denote **v**(x0 ,y0) in all four directions are defined as:

$$\mathbf{X}_{0\deg} = \mathbf{v}(x_{-1}, y_0), \mathbf{Y}_{0\deg} = \mathbf{v}(x_1, y_0) \tag{22}$$

$$\mathbf{X}_{45\deg} = \mathbf{v}(x_{-1}, y_1), \mathbf{Y}_{45\deg} = \mathbf{v}(x_1, y_{-1}) \tag{23}$$

$$\mathbf{X}_{90\deg} = \mathbf{v}(x_0, y_{-1}), \mathbf{Y}_{90\deg} = \mathbf{v}(x_0, y_1) \tag{24}$$

$$\mathbf{X}_{135\deg} = \mathbf{v}(x_1, y_1), \mathbf{Y}_{135\deg} = \mathbf{v}(x_{-1}, y_{-1}) \tag{25}$$

This operator requires the least amount of computation among the edge detectors considered so far. However, like with the vector gradient operator, the difference vector operator is also sensitive to impulsive and Gaussian

noise [6]. As a result, more complex operators with subfiltering are designed. A larger window size is required in this case to allow more data for processing. Although there is no upper limit on the size of the window, usually a 5 x 5 window is preferred since the computational complexity is directly linked to the size of the window. In addition, when the window becomes too large it can no longer represent the characteristics of the local region.

4 VECTOR ORDER STATISTICS OPERATORS

4.1 Introduction

One important family of operators for image processing is based on order statistics [4]. It has played an important role in monochrome image processing and it is also extended to color image filtering and edge detection. This approach is inspired by the morphological edge detectors that have been proposed for the monochrome images. This class of color edge detectors is characterized by linear combinations of the sorted vector samples. Different sets of coefficients of the linear combination give rise to different edge detectors that vary in performance and efficiency. The primary step in order statistics is to arrange a set of random variables in ascending order according to certain criteria. In color space, since we are dealing with 2-D, multichannel variables, there is no universal way of defining an ordering. A number of ways have been proposed to perform multivariate data ordering [13] and they can be classified into marginal ordering (*M*-ordering), reduced aggregate ordering (*R*-ordering), partial ordering (*P*-ordering), and conditional ordering (*C*-ordering). In *M*-ordering, the ordered vectors do not co respond to the original vectors, and *P*-ordering is difficult to implement for digital image processing. *C*-ordering considers only one color component. *R*-ordering is hence more appropriate for color image processing. *R*-ordering reduces each multichannel variable to a scalar value according to a distance criterion.

Let the image vectors in a window W denote $\mathbf{X}_i$, i=1,2,...,n and $D(\mathbf{X}_i, \mathbf{X}_j)$ be a measure of distance between vectors $\mathbf{X}_i$ and $\mathbf{X}_j$. The reduced scalar quantity associated with $\mathbf{X}_i$ is defined as

$$d_i = \sum_{k=1}^{n} D(\mathbf{X}_i, \mathbf{X}_j), i = 1,2,....n \tag{26}$$

The arrangement of d_i in ascending order ($d^{(1)} < d^{(2)} < ... < d^{(n)}$) corresponds to the same ordering to the multivar ate variables:

$$\mathbf{X}^{(1)} \le \mathbf{X}^{(2)} \le ... \mathbf{X}^{(n)} \tag{27}$$

In the ordered sequence, $\mathbf{X}^{(1)}$ is the vector median and vectors appearing at high ranks are referred to as outliers because they diverge the most from the data population.

4.2 Edge Detectors

The vector range (VR) edge detector is the simplest color edge detector based on order statistics. It expresses the deviation of the vector outlier in the highest rank from the vector median in W as follows:

$$VR = D(\mathbf{X}^{(n)}, \mathbf{X}^{(1)}) \tag{28}$$

where VR is small in a uniform area since all vectors are close together, and it gives large output when discontinuities exist. Edges can be obtained by thresholding the VR outputs.

The VR detector, though simple and efficient, is sensitive to noise, especially to impulsive noise. It will respond to a noise pixel at the center of W with n pixels. To improve noise performance, a more general class of operators, vector dispersion edge detector (VDED), is defined as a linear combination of the ordered vectors:

$$VDED = \left\| \sum_{i=1}^{n} \alpha_i \mathbf{X}^{(i)} \right\| \tag{29}$$

where $\|\cdot\|$ denotes the appropriate norm. Note that VR is a special case of VDED with $\alpha_1 = -1$, $\alpha_n = 1$ and $\alpha_i = 0$, i =2,..., n-1. The preceding equation can be further generalized by employing several sets of coefficients and combining the resulting vector magnitude in a suitable way.

The coefficients can be chosen in a way to attenuate noise. One proposed class of operator is the minimum vector dispersion (MVD) detector, and it is defined as:

$$MVD = \min_j \left\{ D\left[\mathbf{X}^{(n-j+1)}, \sum_{i=1}^{l} \frac{\mathbf{X}^{(i)}}{l} \right] \right\}$$

$$j = 1,2,.....,k, k, l < n \tag{30}$$

The choice of k and l depend on n, the size of W. These two parameters control the trade-off between complexity and noise attenuation. This more computationally involved operator can improve edge detection performance in the presence of both impulsive and Gaussian noise. By their nature, impulsive noise differs from the rest of the pixels by a large amount. Therefore, after ordering, the impulsive noise pixels have the highest ranks, $\mathbf{X}^{(n-k+2)}$, $\mathbf{X}^{(n-k+3)}$...,$\mathbf{X}^{(n)}$. Since the distance between these noise pixels and the rest of the pixels are large, Equation (30) can be reduced to the form stated above. Notice that none of the noise pixels appears at this equation, and thus would not affect the edge detection process. The MVD is also robust in Gaussian noise due to the l-points average term.

An alternative design of the generalized VDED operators utilizes the adaptive nearest-neighbor (NN) filter. The coefficients are chosen to adapt to local image characteristics. Instead of constants, the coefficients are determined by an adaptive weight function for each window W. The operator is defined as the distance between the outlier and the weighted sum of all the ranked vectors:

$$NNVR = D\left[\mathbf{X}^{(n)}, \sum_{i=1}^{n} w_i \mathbf{X}^{(i)} \right] \tag{31}$$

The weight function w_i is determined adaptively using transformations of a distance criterion at each image location and it is not uniquely defined. There are two constraints on the weight function:

1. Each weight coefficient is positive, $w_i > 0$.
2. The weight function is normalized,

$$\sum_{i=1}^{n} w_i = 1$$

Since the operator should also attenuate noise, it is important to assign a small weight to the pixels with high ranks i.e. outliers. A possible weight function can be defined as follows:

$$w_i = \frac{d^{(n)} - d^{(i)}}{n \cdot d^{(n)} - \sum_{j=1}^{n} d^{(j)}} \tag{32}$$

One special case for this weight function occurs in highly uniform areas where all pixels have the same distance. The preceding weight function can not be used since the denominator is zero. Since no edge exists in this area, the difference measure of NNVR is set to zero.

The MVD operator can also be incorporated with the nearest neighbor filter to further improve its performance in the presence of impulse noise as follows:

$$MVD = \min_j \left\{ D\left[\mathbf{X}^{(n-j+1)}, \sum_{i=1}^{n} w_i \mathbf{X}^{(i)} \right] \right\}$$

$$j = 1,2,.....,k, k < n \tag{33}$$

A final annotation on the class of vector order statistic operators concerns the distance measure $D(\mathbf{X}_i, \mathbf{X}_j)$. By convention, the Euclidean distance measure L_2 norm is adopted. The use of L_1 norm can also be considered because it reduces the computational complexity by computing the absolute values instead of squares and square root, and it shows no notable deviation in performance. A few other distance measures can also be considered in the attempt to locate an optimal measure, namely, the Manhattan distance metric, the Canberra metric, the Czekanowski coefficient, and the angular distance measure.

5 Quantative Performance Evaluation Procedure

There are two schools of thought with regards to the evaluation of computational vision algorithms in general. The first maintains that vision algorithms should be evaluated evaluated in the context of particular task as suggested by [14]. In the context of edge detection and other low-level vision tasks this translates to measuring how much a particular algorithm contributes to the success of higher-level procedures that carry out the low level operation, for example, image segmentation and object recognition.

The second school of thought holds that vision algorithms can be evaluated in terms of their performance with regard to some suitably defined ground truth data as described by [15]. In this project we adopt this latter philosophy, and present the results of evaluating the performance of the edge detection algorithms described above on a set of real and synthetic images for which the ground truth is known. This has been made possible by the introduction of the Berkeley Segmentation Database (BSDS300) [7].

The current public distribution of the BSDs300 contains 300 colour images of size 481 × 321 pixels. For each of these images, the database provides between 4 and 9 human segmentations in the form of label maps. The segmentations are provided separately for the gray-scale and colour versions of each image, and the complete database is split in two sets. A training image set consisting of 200 images and their corresponding segmentations, and a testing data set consisting of the remaining 100 images and their human segmentations. The BSDS300 was originally designed for evaluation of image segmentation algorithms. Since edge detection or boundary marking is an essential part of the heriarchical image segmentation algorithms, we can derive the ground truth data for edge detection from the human segmentation label maps available in this data set as described in the next sub-section.

Apart from the BSDS300, we also included a few machine generated synthetic images that particularly used different gradient of colors in the image segments to effectively evaluate the performance of the color edge detectors.

5.1 Extracting Ground Truth Edges from Human Segmentation

The segmentations from the human ground truth data in the BSDS300 are stored as labeled images where pixels within the same region have identical labels. To extract the edges from the labeled images we could, as a first approach, simply mark as an edge any pixels that have a neighbor with a different label. This, however, yields edges that are two pixels thick. Thick edges create two problems. First, very thin or very small regions will disappear altogether, having been replaced by solid clusters of edge pixels within which the region-structure of the image is lost. Secondly, thick edges complicate the matching procedure between the edge detection algorithm and ground truth data and are likely to introduce unwanted artifacts in the resulting performance parameters (e.g precision/ recall scores). We could also attempt to mark pixels uniformly on one side (e.g. to the left and above) of edge boundaries but this also introduces artifacts.

To eliminate these problems, we first generate edge maps that have twice the resolution of the segmentations. In these higher-resolution images, edges can be accurately localized to lie between the pixels corresponding to the original regions in the low resolution segmentation. The procedure for generating the super-sampled edges is simple: We super-sample each individual region in the original segmentation and find its edge in the higher resolution image. The final edge map is then formed by down-scaling the super-sampled image. Figure 2 shows segmentation, the boundaries extracted by marking as boundary any pixels that have neighbors with a different label, the super-sampled edge map and the downsampled edge map that is used as the final ground truth image.

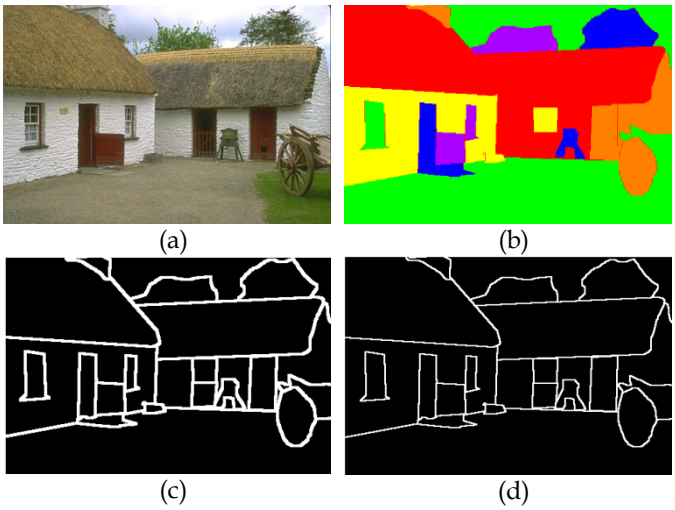


Fig. 2. (**a**) Original image BSDS300 385028.jpg (481x321). (**b**) Segmentation labels as given by human annotation (**c**) Edges generated by marking every pixel that has at least one neighbour with a different label (superscaled edges are 2 pixels thick). (**d**) Down-sampled (481x321) image. Edges are now more accurately localized.

5.2 Performance Parameters

Comparison of an edge map, obtained by the edge detection algorithm, to its ground truth map is achieved by the measurement of the statistical parameters of the matching algorithm. These statistical parameters include metrics such as the number of correctly detected edge pixels, called Total Positives (TP), the number of pixels erroneously classified as edge pixels, called False Positives (FP), the amount of edge pixels that were not classified as edge pixel, called False Negatives (FN) and the Precision and Recall scores. From these measures, we compute the following statistical metrics:

The percentage of pixels that were correctly detected (Pco):

$$P_{co} = \frac{TP}{\max(N_I, N_B)} \tag{34}$$

where N_I represents the number of edge pixels of the ground truth image and N_B represents the number of edge pixels actually detected by the algorithm.

The percentage of pixels that were not detected, i.e. the percentage of false negatives (Pnd):

$$P_{nd} = \frac{FN}{\max(N_I, N_B)} \tag{35}$$

The percentage of pixels that were erroneously detected as edge pixels, i.e. the percentage of false positives (Pfa):

$$P_{fa} = \frac{FP}{\max(N_I, N_B)} \tag{36}$$

Precision is defined as the proportion of edge pixels in ground truth image for which we can find a matching edge pixel in the edge detected output im age,

$$PR = \frac{TP}{N_B} \tag{37}$$

In a similar way, Recall is defined as the proportion of pixels in edge detected output image for which we can find a suitable match in the ground truth image,

$$RC = \frac{TP}{N_I} \tag{38}$$

The figure of merit of Pratt [16] is another useful measure for assessing the performance of edge detectors. This measure uses the distance between all pairs of points corresponding to quantify, with precision, the difference between the edges. The figure of merit is defined as:

$$FOM = \frac{1}{\max(N_I, N_B)} \sum_{i=1}^{N_B} \frac{1}{1+\alpha \cdot d_i^2} \tag{39}$$

where N_I and N_B are the points of edges in the edge detected image and ground truth image, respectively, d_i is the distance between a edge pixel and the nearest edge pixel of the ground truth and α is an empirical calibration constant and was used α=1/9, optimal value established by Pratt [16]. The figure of merit of Pratt's FOM is an indicator of the quality of edge, and reflects the overall behavior of the distances between the edges, being a relative measure, which varies in the range [0, 1], where 1 represents the optimal value, i.e., the edges detected exactly coincide with the ground truth.

5.3 Matching Strategy

To compute the statistical parameters described above, we need a method for determining correspondence between edge pixels in the ground truth image and the corresponding output image of edge detection algorithm. We used the following steps in the matching strategy:

a) Count the edge pixels in the edge detected output image that exactly match with the ground truth image. These are correct identifications and are termed as True Positives.

b) Eliminate the correctly identified edge pixels from the edge detected output image and the ground truth image. For the remaining edge pixels, count the edge pixels in the detected image that are at an Euclidean distance of sqrt(2) (i.e. close vicinity) from the ground truth edge pixel. These are the edge pixels that are just one pixel away from the ground truth edge pixel. These are locational errors and are termed as local positives.

We take Total Positives as the sum of true positives and the local positives in our metric calculations.

6 Experimental Results and Observations

A total of 4 edge detection algorithms from the class of the vector order statistic operators are implemented and their performances are evaluated along with the Canny edge detector implementation. Table 1 provides a list of the 4 edge detectors. As stated above, we have used real images from the BSDS300 dataset that provide human ground truth edge data for experimentation. Apart from

these real images, we have used machine generated synthetic images and some random images in the subjective and quantitative evaluation.

Table 1 Edge Detectors for Evaluation

Edge Detector	Description
Canny	Monochrome edge detection
VR	VR operator (W:3x3) with L_1 norm
MVD	MVD operator (W:3x3) with k=2, l=3
NNMVD	NNMVD operator (W:3x3) with k=3
NNVR	NNVR operator (W:3x3)

6.1 Subjective Evaluation

Figure 3 shows the application of the edge detection algorithms from Table 1 on real image from the BSDS300 dataset along with its human ground truth edge data. Similarly Figure 4 shows the response of edge detection algorithms on a machine generated synthetic image. The subjective evaluation of edge detectors can be based on several criterions: ease in recognizing objects, continuity of edges, thinness of edges and performance in suppressing noise.

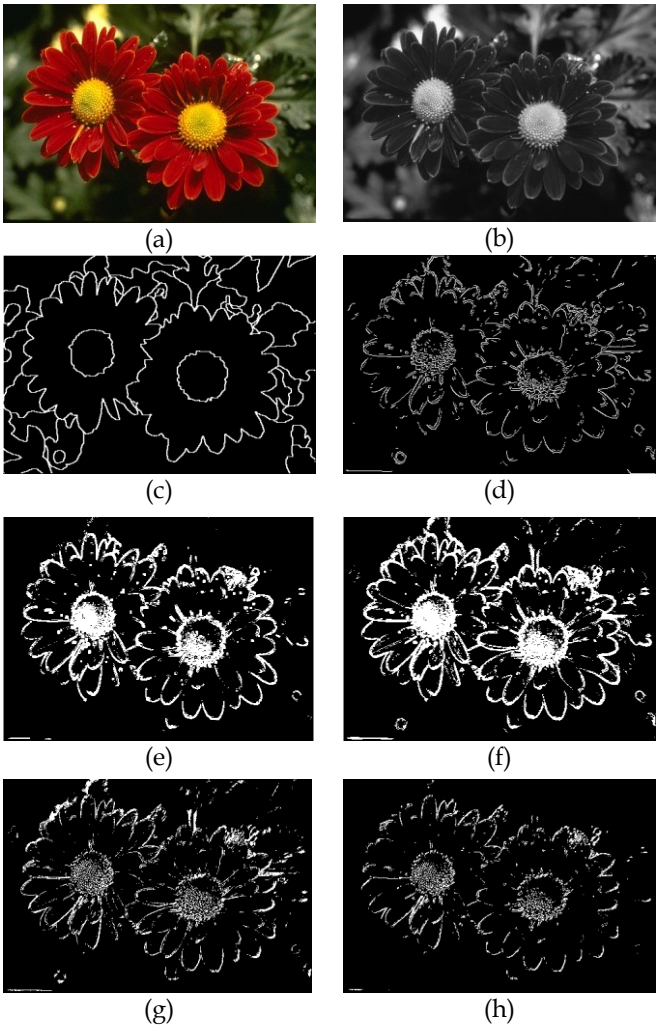


Fig. 3. (**a**) Original image 124084.jpg (481x321) from BSDS300. (**b**) Gray-scale image. (**c**) Human ground truth image. (**d**) Canny Edge Output. (**e**) VR Edge Output. (**f**) MVD Edge Output. (**g**) NNMVD Edge Output. (**h**) NNVR Edge Output.

A few observations can be drawn from the visual inpsection of the edge maps produced from various edge detectors applied to real and synthetic images:

- The performance of Canny, NNVR and NNMVD edge detectors is same in the sense that they produce more or less similar edge maps. For large color variations NNVR and NNMVD perform better than Canny as seen in case of synthetic image (Figure 4)
- VR and MVD perform superior as compared to the other edge detectors as they produce more prominent edges. The MVD edge detector produce thinner edges and is less sensitive to small texture variations because of the averaging operation which smooths out small variations.
- MVD edge detector is more sensitive to color variations and hence can produce edges with very small color gradients as well. This can be further controlled by properly choosing the threshold value.

6.2 Statistical Evaluation

As described in section 5.2, comparison of an edge map obtained by the edge detection algorithm, with its ground truth map is achieved by the measurement of the statistical parameters such as percentage of correctly detected edge pixels (Pco), percentage of erroneously classified edge pixels (Pfa), percentage of pixels not detected as edge pixels (Pfn), Precision and Recall scores and Pratt's figure of merit (FOM). Table 2 and 3 gives the statistical parameters calculated for edge detection from real image and synthetic image.

Table 2 Statistical Edge Parameters for 124084.jpg image

Params	Canny	VR	MVD	NNMVD	NNVR
Pco	31.6%	24.5%	28.3%	22.6%	22.5%
Pfn	68.4%	28.7%	42.0%	77.4%	77.5%
Pfa	46.1%	75.5%	71.7%	36.4%	38.0%
Precision	40.7%	24.5%	28.3%	38.3%	37.2%
Recall	31.6%	46.1%	40.2%	22.6%	22.5%
FOM	42.8%	50.2%	51.3%	32.6%	32.6%

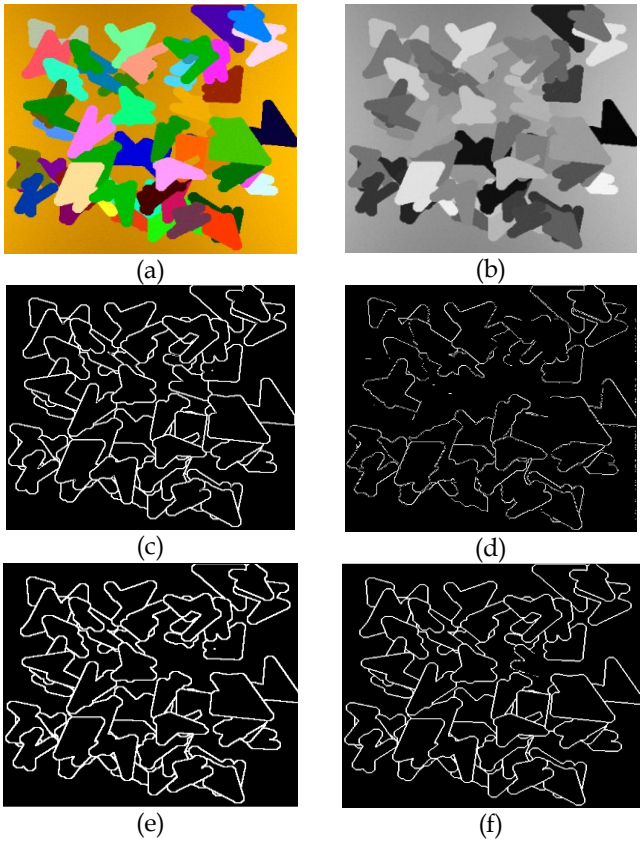
Table 3 Statistical Edge Parameters for synthetic image

Params	Canny	VR	MVD	NNMVD	NNVR
Pco	40.1%	84.6%	82.3%	35.9%	40.2%
Pfn	59.9%	2.5%	17.7%	64.1%	59.8%
Pfa	3.0%	15.4%	0.3%	0.2%	0.3%
Precision	93.0%	84.6%	99.7%	99.6%	99.3%
Recall	40.1%	97.1%	82.3%	35.9%	40.2%
FOM	39.7%	98.5%	82.5%	35.8%	40.0%

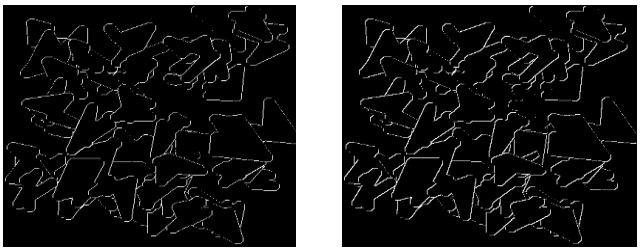
From the results in Table 2 and Table 3, few conclusions can be drawn:

- For real color images, Canny and MVD edge detectors have higher percentage of correctly detected edge pixels as compared to VR, NNVR and NNMVD.
- But at the same time VR and MVD has smaller Pfn scores which indicate that Canny edge detector has higher percentage of pixels that were not detected as true edges.
- In terms of Pratt's FOM, VR and MVD have better scores among all the edge detectors since they both detect edge pixels that are closer to the real ground truth edge pixels.
- For the synthetic color image with large color variation, VR and MVD has huge performance gain in terms of correctly detected edge pixels as compared to Canny, NNVR and NNMVD.
- Moreover, MVD, NNVR and NNMVD have a very high precision score among all the edge detectors while VR has the best FOM score. Clearly we observe that for color images with large color variation, color edge detectors give better quality performance as compared to monochrome edge detection techniques like Canny edge detector.

For statistical evaluation we used L2 norm for all the distance calculations and a constant window size of 3 x 3. If computational complexity permits, then we can use a larger window size like 5 x 5 and some more accurate distance measures such as Canberra metric and angular distance measure.



(a) (b) (c) (d) (e) (f)



(g) (h)

Fig. 4. (**a**) Original synthetic image 20004.ppm (400 x 400). (**b**) Gray-scale image. (**c**) Ground truth image. (**d**) Canny Edge Output. (**e**) VR Edge Output. (**f**) MVD Edge Output. (**g**) NNMVD Edge Output. (**h**) NNVR Edge Output.

6.3 Noise Performance

A number of edge detection experiments have been conducted using various noise distributions at various noise levels to contaminate the test image. In each case, *FOM* has been measured and used as the performance criterion. We considered Gaussian noise distribution, but other forms of noise corruption such as Gaussian Impulsive noise, exponential or correlated noise can also be used. For the corruption of the synthetic image, the noise process in each channel has been considered as an independent process. While in case of real image, the noise process has been considered as a correlated process since there is some indication that this type of correlation may exist in real color images. The noise performance of the color edge detectors are shown graphically in Figure 5. The edge detection output on a BSDS300 real image corrupted with Gaussian noise is depicted in Figure 6.

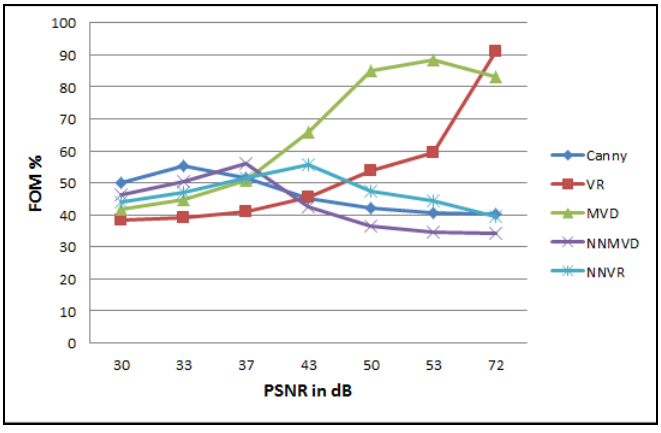


Fig. 5a. FOM plots of color edge detectors under noise corruption for synthetic image.

A few observations can be made from the results :

- The Canny, VR and NNVR edge detectors are very sensitive to Gaussian noise. As observed VR fires indiscriminately for Gaussian noise corruption since it just takes the difference between the first and the last vector sample in the ordered vector space.
- Canny edge detector can produce better noise performance with added complexity by increasing the variance σ in the Gaussian filtering stage.
- MVD and NNMVD vector order statistics edge

detectors show more robustness in the presence of noise because of the inherent averaging operation in the design of these detectors. We can also confirm that noise performance improves with increase in the complexity of these edge detectors, which are controlled by the two parameters k and l.

- As observed from the FOM plots (Figure 5) of real and synthetic color images, MVD edge detector has superior performance among all edge detectors in various noise distributions.

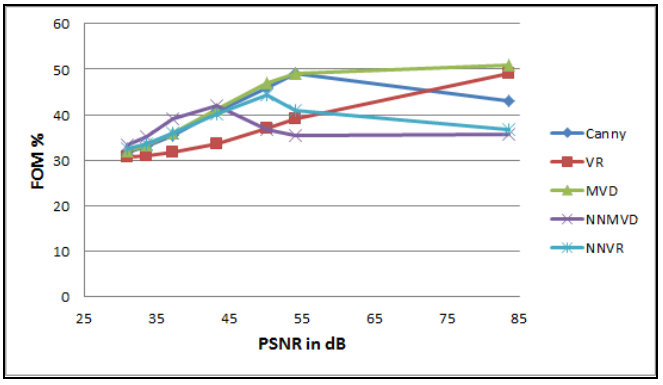
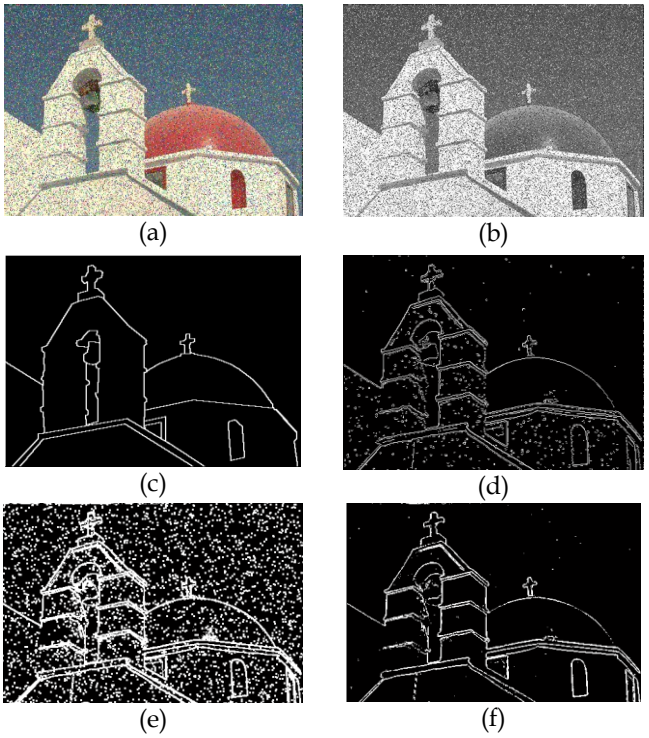


Fig. 5b. FOM plots of color edge detectors under noise corruption for real image.



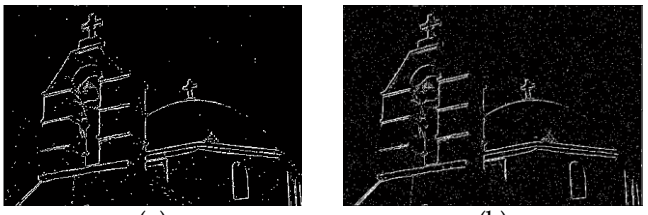


Fig. 6. (**a**) Original image 118035.jpg (481x321) from BSDS300 corrupted with Gaussian noise. (**b**) Gray-scale image. (**c**) Ground truth image. (**d**) Canny Edge Output. (**e**) VR Edge Output. (**f**) MVD Edge Output. (**g**) NNMVD Edge Output. (**h**) NNVR Edge Output.

7 Conclusion

The problem of color edge detection has been studied using vector order statistics in this project. **A** family of color edge detectors based on vector order statistics has been proposed because these are effective with multichannel data and are computationally efficient. The design parameters of this class of edge detectors can be appropriately choosen for better noise suppression at the cost of increasing complexity. The performance of all edge detectors was evaluated both subjectively and objectively using various statistical parameters. The Minimum Vector Dispersion (MVD) color edge detector scores high points in objective tests and the edge maps produced by this edge detector are perceived favorably by human eyes under subjective evaluation. Different vision applications have different requirements on edge detection, and though some of the general characteristics of color edge detectors were addressed, it is still better to select edge detector that is optimum for a particular application.

References

[1] J. Thomas Allen and T. Huntsberger, "Comparing Color Edge Detection and Segmentation Methods," *IEEE Proceedings Southeastcon*, pg. 722, Session 11C5, 1989.

[2] K. N. Plataniotis and A. N. Venetsanopoulos, "*Color Image Processing and Applications,*" Springer, 2000.

[3] P. Androutsos, D. Androutsos, K.N. Plataniotis, and A.N. Venetsanopoulos. "Colour Edge Detectors: An Overview and Comparison," *IEEE Proceedings*, pg. 607, 1997.

[4] P.E. Trahanias and A.N. Venetsanopoulos, "Vector order Statistics Operators as Color Edge Detectors," *IEEE Transaction on Systems, Man and Cybernetics-Part B: Cybernetics*, Vol. 26, No. 1, pg. 137, February 1996.

[5] J. Canny, "A computational approach to edge detection,", *Pattern Analysis and Machine Intelligence*, 8(6):679-698, November 1986.

[6] I. Pitas and A.N. Venetsanopoulos. "Edge detectors based on nonlinear filters," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. PAMI-8, no. 4, pp. 538-550, July 1986.

[7] Martin, D., & Fowlkes, C. (2001). "The Berkeley segmentation database and benchmark," *Computer Science Department*, Berkeley University. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/.2001

[8] D. Marr and E. Hildreth, ''Theory of edge detection,'' *Proc. R. Soc. London, Ser. B* **207**, 187–217, 1980.

[9] R. Machuca and K. Phillips, ''Applications of vector fields to image processing,'' *IEEE Trans. Pattern. Anal. Mach. Intell.* PAMI-5(3), 316–329, 1983.

[10] S. D. Zenzo, ''A note on the gradient of a multiimage,'' *Comput. Vis. Graph. Image Process.* 36, 1–9, 1986.

[11] J. Scharcanski and A. N. Venetsanopoulos, ''Colour image edge detection using directional operators,'' in *Proc. IEEE Workshop on Nonlinear Signal and Image Processing*, Vol. II, pp. 511–515, 1995.

[12] M. A. Ruzon and C. Tomasi, "Color edge detection with the compass operator," In Proceedings of *IEEE Conference on Computer Vision and Pattern Recognition*, pages 160-166, June 1999.

[13] V. Barnett, ''The ordering of multivariate data,'' *J. R. Stat. Soc. A* 139(3), 318–343, 1976.

[14] M. Heath, S. Sarkar, T. Sanocki and K. Bowyer, ''Comparison of edge detectors: a methodology and initial study,'' in *IEEE Conf. on Computer Vision and Pattern Recogn tion*, San Francisco 1996.

[15] Martin, D., Fowlkes, C., Tal, D., & Malik, J. "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," *IEEE International Conference on Computer Vision* (pp. 416-425), 2001.

[16] I. A. Abdou and W. Pratt, "Quantitative design and evaluation of enhancement/thresholding edge detectors," in *Proceedings of the IEEE*, vol. 67, no. 5, pp. 753 766, 1979.