Automatic Edge Detection by Combining Kohonen SOM and the Canny Operator

P. Sampaziotis and N. Papamarkos

Image Processing and Multimedia Laboratory,
Department of Electrical & Computer Engineering,
Democritus University of Thrace,
67100 Xanthi, Greece
papamark@ee.duth.gr

Abstract. In this paper a new method for edge detection in grayscale images is presented. It is based on the use of the Kohonen self-organizing map (SOM) neural network combined with the methodology of Canny edge detector. Gradient information obtained from different masks and at different smoothing scales is classified in three classes (Edge, Non Edge and Fuzzy Edge) using an hierarchical Kohonen network. Using the three classes obtained, the final stage of hysterisis thresholding is performed in a fully automatic way. The proposed technique is extensively tested with success.

1 Introduction

Changes or discontinuities in an image amplitude attribute such as intensity are fundamentally important primitive characteristics of an image because they often provide an indication of the physical extent of objects within the image. The detection of these changes or discontinuities is a fundamental operation in computer vision with numerous approaches to it.

Marr and Hildreth [3] introduced the theory of edge detection and described a method for determining the edges using the zero-crossings of the Laplacian of Gaussian of an image. Canny determined edges by an optimization process [1] and proposed an approximation to the optimal detector as the maxima of gradient magnitude of a Gaussian-smoothed image. Lily Rui Liang and Carl G. Looney proposed a fuzzy classifier [2] that detects classes of image pixels corresponding to gray level variation in the various directions. A fuzzy reasoning approach was proposed by Todd Law and Hidenori Itoh [8], in which image filtering, edge detection and edge tracing are completely based on fuzzy rules. The use of self-organising map and the Peano scan for edge detection in multispectral images was proposed by P.J. Toivanen and J. Ansamaki [5]. In [10], Pihno used a feed-forward artificial neural of perceptron-like units and trained it with a synthetic image formed of concentric rings with different gray levels. Weller [11] trained a neural net by reference to a small training set, so that a Sobel operator was simulated. In Bezdek's approach [12], a neural net is trained on

M. Lazo and A. Sanfeliu (Eds.): CIARP 2005, LNCS 3773, pp. 954–965, 2005.

all possible exemplars based on binary images, with each windowed possibility being scored by the (normalised) Sobel operator.

Among the various edge detection methods proposed so far, the Canny edge detector is most widely used due to its optimality to the three criteria of good detection, good localization, and single response to an edge.

In this paper a new edge detection technique is proposed which improves the canny edge detection the following way:

- Utilizes edge information extracted not only from one edge detection masks but from a number of different masks.
- Uses Kohonen SOM in order to obtain three main classes of edges (Edge, Fuzzy-Edge, Non-Edge) that are next used to automatically obtain the final edge pixels according to the Canny's hysterisis thresholding procedure.

The proposed technique is extensively tested with many different types of images and it is found that it performs satisfactory even with degraded images.

2 Overview

A typical implementation of the Canny edge detector follows the steps below:

1. Smooth the image with an appropriate Gaussian Filter to reduce noise.
2. Determine gradient magnitude and gradient direction at each pixel.
3. Suppress non edge pixels with non maximum suppression. If the gradient magnitude at a pixel is larger than those at its two neighbors in the gradient direction, mark the pixel as an edge. Otherwise, mark the pixel as the background.
4. Remove the weak edges by hysteresis thresholding.

The first step of the Canny edge detector is the gaussian smoothing. Gaussian filters are low-pass filters and thus apart from filtering the noise they also blur an image. The Gaussian outputs a 'weighted average' of each pixel's neighborhood, with the average weighted more towards the value of the central pixels. The degree of smoothing is determined by the standard deviation of the filter. Filtering an image with a gaussian does not preserve edges. Larger values of standard deviation correspond to images at coarser resolutions with low detail level.

After the image filtering, the next step is the determination of the image gradient. The simplest method to compute the gradient magnitude $G(j,k)$ refers to the combination of row $G_R(j,k)$ and column $G_C(j,k)$ gradient. The spatial gradient magnitude is given by:

$$G(j,k) = \sqrt{G_C(j,k)^2 + G_R(j,k)^2} \tag{1}$$

and the orientation of the spatial gradient with respect to the row axis is:

$$\theta(j,k) = \arctan\left\{\frac{G_C(j,k)}{G_R(j,k)}\right\} \tag{2}$$

The discrete approximation of $G_R(j,k)$ and $G_C(j,k)$ can be given by the pixel difference, separated by a null value [9] :

$$G_R(j,k) = P(j,k+1) - P(j,k-1) \tag{3}$$

$$G_C(j,k) = P(j-1,k) - P(j+1,k) \tag{4}$$

The separated pixel difference is sensitive to small luminance fluctuations in the image and thus it is preferred to use 3×3 spatial masks which perform differentiation in one coordinate direction and spatial averaging in the and orthogonal direction simultaneously. The most widely used masks are the Sobel, Prewitt and Frei-Chen operators. As show in figures 1 and 2 these masks have different weightings, in order to adjust the importance of each pixel in terms of its contribution to the spatial gradient. Frei and Chen have proposed north, south, east, and west weightings so that the gradient is the same for horizontal, vertical, and diagonal edges, the Prewitt operator is more sensitive to horizontal and vertical edges than to diagonal edges and the reverse is true for the Sobel operator.

$$\frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \text{ (a)} \qquad \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \text{ (b)} \qquad \frac{1}{2+\sqrt{2}}\begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix} \text{ (c)}$$

Fig. 1. Row gradient masks: (a) Sobel (b) Prewitt (c) Frei-Chen

$$\frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \text{ (a)} \qquad \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \text{ (b)} \qquad \frac{1}{2+\sqrt{2}}\begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix} \text{ (c)}$$

Fig. 2. Column gradient masks: (a) Sobel (b) Prewitt (c) Frei-Chen

Hysterisis thresholding uses a high threshold T_{high} and a low threshold T_{low} which both are user-defined. Every pixel in an image that has gradient magnitude greater than T_{high} or less than T_{low} is presumed to be an edge or a non-edge pixel respectively. Any other pixel that is connected with an edge pixel and has gradient magnitude greater than T_{low} is also selected as edge pixel. This process is repeated until every pixel is marked as edge or non edge pixel. In terms of clustering, by selecting the two thresholds, the image pixels are grouped in three clusters : Edge cluster, non-edge cluster and fuzzy-edge cluster with fuzziness defined by means of spatial connectivity with edge pixels.

The basic idea of this work is to automate the edge map clustering using the Kohonen self-organizing map. As described previously in this section, gradient depends on the size of the gaussian filter and the differentiation operator . Thus

it is more robust to create a feature space with gradient information obtained from different masks and at different detail levels of the image and represent the gradient magnitude in a vectorial form and not with a scalar value.

2.1 Kohonen SOM Neural Network

The Kohonen SOM is a neural network that simulates the hypothesized self-organization process carried out in the human brain when some input data are presented [4]. The Kohonen network consists of two layers. The first layer is the input layer and the second is the competition layer in which the units are arranged in a one or two dimensional grid. Each unit in the input layer has a feed-forward connection to each unit in the competition layer. The architecture of the Kohonen network is shown in figure 3. The network maps a set of input vectors into a set of output vectors (neurons) without supervision. That is, there is no a-priori knowledge of the characteristics of the output classes. The training algorithm is based on competitive learning and is as follows :

1. Define of the desired set A of output classes c_i

$$A = \{c_1, \ldots, c_N\} \tag{5}$$

   and the topology of the competition layer neurons.
2. Initialize output units c_i with reference vectors $\mathbf{w}_{c_i}$ chosen randomly from a finite data set $\mathbf{D} = \{\mathbf{d}_1, \ldots, \mathbf{d}_M\}$ and set the time parameter $t = 0$.
3. Present an input vector $\mathbf{d}$ and find the winner output neuron $s(\mathbf{d}) = s$:

$$s(\mathbf{d}) = arg\ min_{c \in A} \|\mathbf{d} - \mathbf{w}_c\| \tag{6}$$

4. Adapt each unit c according to

$$\Delta \mathbf{w}_c = \epsilon(t) h_{sh}(\mathbf{s} - \mathbf{w}_c) \tag{7}$$

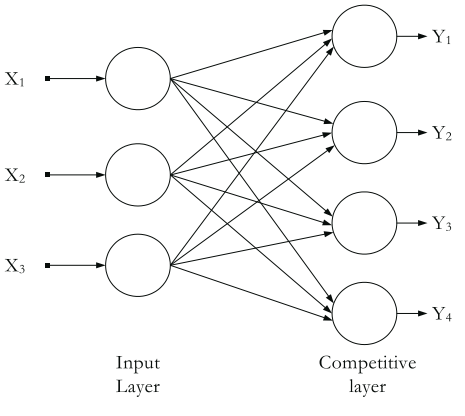


Fig. 3. Architecture of the Kohonen Self-Organising Map

where $\epsilon(t)$ is the function that controls the learning rate and h_{sh} the function that defines the neighborhood units of the winner neuron that will be adapted. For the learning rate in this work we used the function:

$$\epsilon(t) = \epsilon_{initial} \left(\frac{\epsilon_{final}}{\epsilon_{initial}} \right) \tag{8}$$

and as a neighborhood function the gaussian :

$$H_{cs} = exp \left(\frac{-\|c - s\|}{2\sigma^2} \right) \tag{9}$$

with standard deviation varied according to

$$\sigma(t) = \sigma_{initial} \left(\frac{\sigma_{final}}{\sigma_{initial}} \right) \tag{10}$$

.

5. Repeat steps 3 and 4 until all the vectors of the training dataset **D** are presented to the network.
6. Increase the time parameter:

$$t = t + 1; \tag{11}$$

7. If $t < t_{max}$ continue with step 3.

3 Description of the Method

As shown in figure 4 , the proposed edge detection method consists of two parts.

The first one follows the the three first steps of the Canny edge detector. Firstly we smooth the grayscale image I with an appropriate Gaussian Filter of standard deviation $\sigma_{central}$ in order to reduce image noise. We call the smoothed image I_C. Then we calculate gradient magnitude and direction using the Sobel operator and perform non maximum-suppression. Every pixel with gradient magnitude greater than zero is set 0 (edge) and all the other pixels are set to 255 (non-edge). This process leads to a single-pixel width binary edge map M. The second part is the classification of images pixel into three clasees (Edge, Non-Edge, Fuzzy Edge). We separately smooth the original grayscale image I

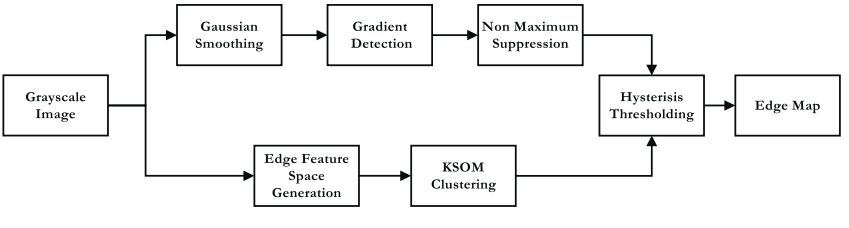


Fig. 4. Flowchart of the proposed method

with a gaussian filter of standard deviation σ_{low} and σ_{high}. The values of σ_{low} and σ_{hight} have a small deviation above and below σ_{center} respectively in order to create different detail levels but also avoid the problem of edge dislocation. For each of these three smoothed images I_L, I_C and I_H, we compute the gradient magnitude using Sobel,Prewitt, Frei-Chen and Separated Pixel Difference operator.

For every pixel P of the image I we assign a 12-component vector. Each vector's element represents gradient magnitude from different combination of smoothing scale and differentiation mask. This process produces a 12-dimension feature space **D**, which will be sampled in order to train the Kohonen SOM.

As shown in figure 5 we approach the clustering process in a hierarchical way, which has been carried out after a large number of experiments.

At the first level, we use a Kohonen map with three output units connected in line topology. These output units represent three clusters: high,medium and low gradient class. The training dataset for the Kohonen map consists of randomly choosen vectors of the input space **D**. After the training of the Kohonen network we assign each pixel of the image to one of the output classes according to the euclidean distance between the pixel's vector in feature space **D** and the vectors of the SOFM output units.

At the second level, all the pixels that are mapped into the high and medium gradient class are grouped in order to form the Edge Pixel class. The Low Gradient class is splitted in two classes: the Fuzzy-Edge Pixel class and the Non-Edge Pixel class, using a Kohonen map with the same topology as the one at the first

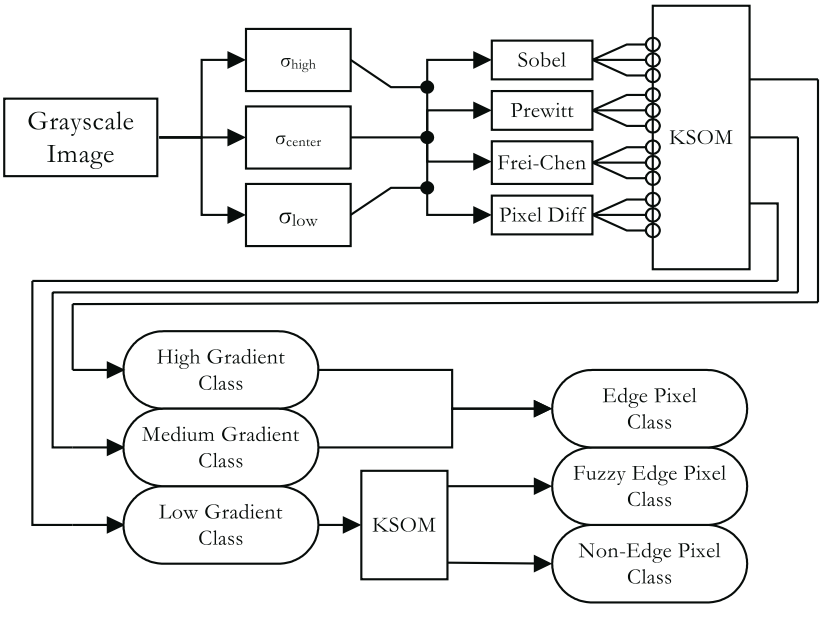


Fig. 5. Flowchart of the clustering process

level but with two output units. All the pixels that were mapped into the Low Gradient class on the first level, are now assigned at these two classes.

The next step of the method is the hysterisis thresholding in a revised way. By integrating the information of the pixel class labeling performed in the previous step, there is no need for the user-defined thresholds T_{low} and T_{high}. Hysterisis thresholding is summarized as follows:

1. Mark as Non-Edge class pixel, every pixel that is marked as non-edge (255) in the binary edge map M.
2. Select a pixel P that belongs to the Edge class.
3. Every pixel that is connected with 8-neighborhood with P and belongs to the Fuzzy-Edge class is marked as edge pixel(0) and it is classified into the Edge class.
4. Repeat step 2 for all pixels of the Edge-class.
5. The remaining Fuzzy-edge class pixels are classified into the Non-edge class and marked as non-edge pixels (255).

4 Experimental Results

The method analysed in this paper is implemented in visual enviroment (Borland Delphi) and tested on several images with satisfactory results. For an AMD Athlon 64 (2GHz) bazed PC with 512 MB RAM, the processing time for a 512×512 image with a Kohonen Som network trained for 300 epochs with 1000 samples, was 2.55 seconds. Edges extracted with the proposed method are shown in figure 6. In 6(b) we have the binary edge map M after gaussian smoothing with $\sigma_{central} = 1$, gradient detection with the sobel operator and non-maximum suppression. In 6(c) we can see the result of the classification using the Kohonen SOM. Pixels classified to the Non-Edge class are shown in black color. Red coloured pixels are the pixels that belong to the Edge class and the pixels of the Fuzzy-edge class are shown in green color. The parametres of the Kohonen maps for these examples are: $\epsilon_i = 0.9$, $\epsilon_f = 0.01$ and $T_{max} = 400$ with training vectors from an input space formed as described previously with $\sigma_{low} = 0.8$ and $\sigma_{high} = 1.2$. The final edges exctracted with automatic hysterisis thresholding are shown in 6 (d). Two additional examples are shown in figures 7 and 8.

In order to have some comparative results, our technique was tested against the results of objective edge evaluation and detector parameter selection method proposed in [6]. In this work, Yitzhak Yitzhaky and Eli Peli propose a statistical objective performance analysis and detector parameter selection method, using detection results produced by different parameters of the same edge detector. Using the correspondence between the different detection results, an estimated best edge map, utilized as an estimated ground truth (EGT), is obtained. This is done using both a receiver operating characteristics (ROC) analysis and a Chi-square test. The best edge detector parameter set (PS) is then selected by the same statistical approach, using the EGT. This method was implemented in Matlab for the canny edge detector.

For the tests we used six ground truth images from the GT dataset used in [7] which is freely available on the internet. In figure 9 we see the test images and corresponding ground truth images. In these manually created GT images black represents edge, gray represents no-edge and white represents dont care. The GT is created by specifying edges that should be detected and regions in which no edges should be detected. Areas not specified either as edge or as no-edge default to dont-care regions. This makes it practical to specify GT for images that contain regions in which there are edges but their specification would be tedious and error-prone (for example, in a grassy area) [7]. The results of the pixel based comparison between the ground truth and the edge images were based on the following values:

- True positives (TP): Number of pixels marked as edges, which coincide with edge pixels in the GT.
- False positives (FP): Number of pixels marked as edges, which coincide with non-edge pixels in the GT.
- True negatives (TN): Number of pixels marked as non-edges, which coincide with non-edge pixels in the GT.
- False negatives (FN): Number of pixels marked as non-edges, which coincide with edge pixels in the GT.

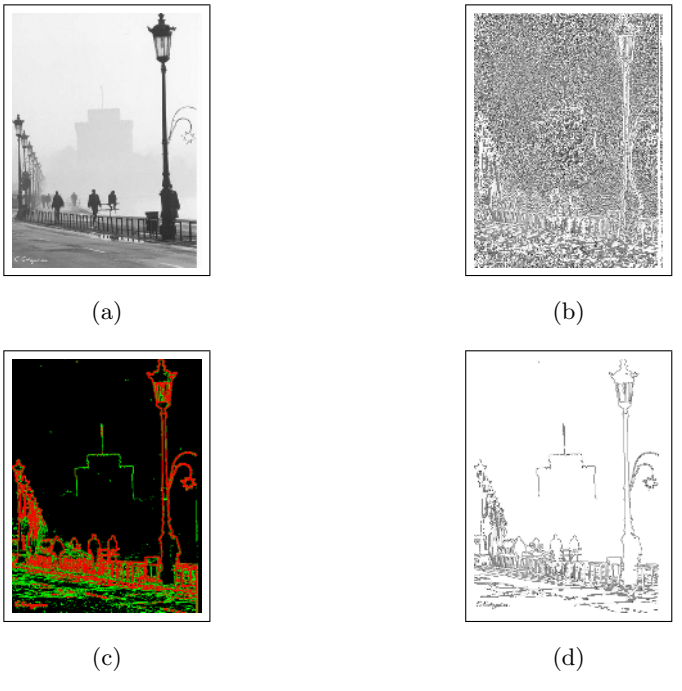
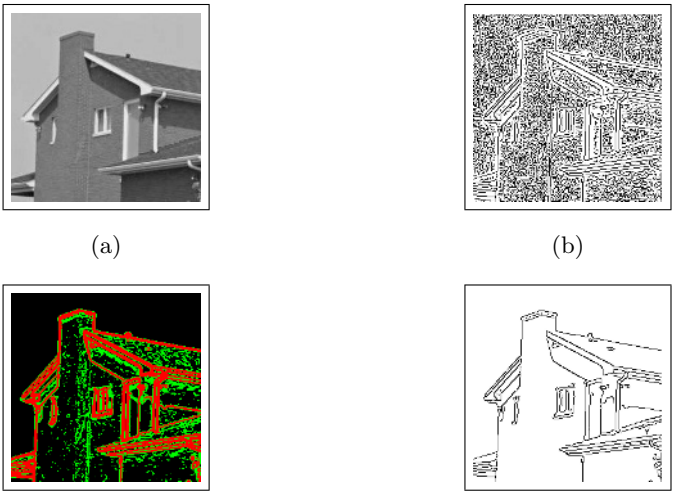


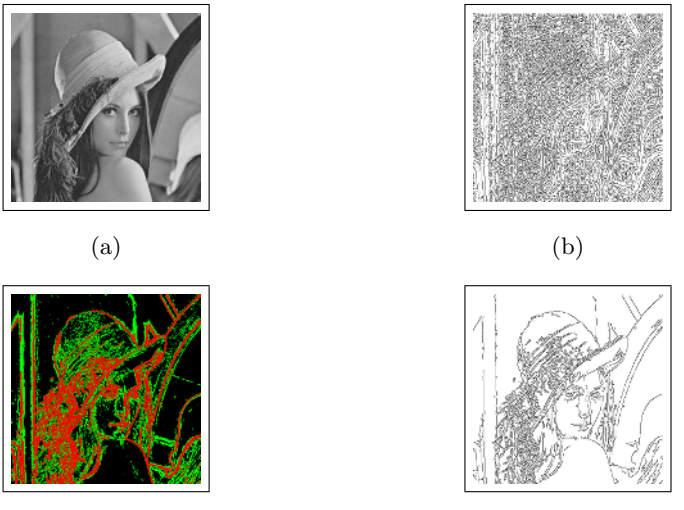
Fig. 6. (a) Original grayscale image, (b) Binary edge map M after smoothing differentation and non-maximum suppression, (c) Edge classes obtained by Kohonen SOM (d) Final edge map after automatic hysterisis thresholding



(a) (b)

(c) (d)

Fig. 7. (a) Original grayscale image, (b) Binary edge map M after smoothing differentation and non-maximum suppression, (c) Edge classes obtained by Kohonen SOM (d) Final edge map after automatic hysterisis thresholding



(a) (b)

(c) (d)

Fig. 8. (a) Original grayscale image, (b) Binary edge map M after smoothing differentation and non-maximum suppression, (c) Edge classes obtained by Kohonen SOM (d) Final edge map after automatic hysterisis thresholding

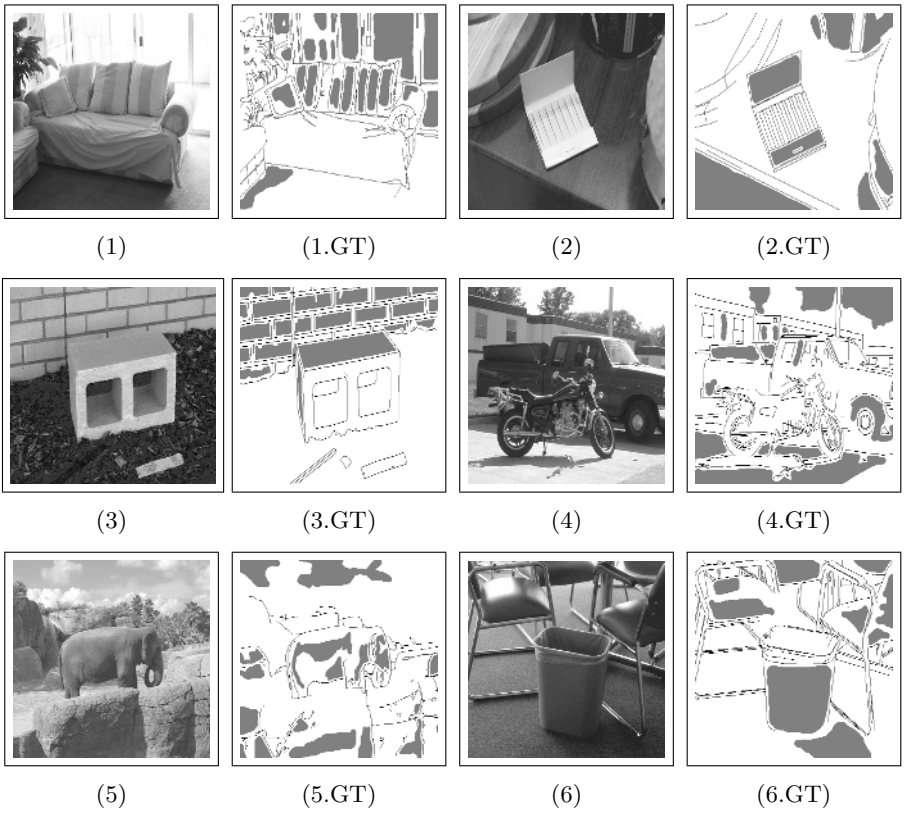


Fig. 9. Images used for pixel-based evaluation

The calculation of these values is performed as follows: if a detector reports an edge pixel within a specified tolerance T_{match} of an edge in the GT, then it is counted as a true positive (TP) and the matched pixel in the GT is marked so that it cannot be used in another match. The Tmatch threshold for tolerance in matching a detected edge pixel to GT allows detected edges to match the GT even if displaced by a small distance [7]. In our test we used a value of Tmatch = 1. If a detector reports an edge pixel in a GT no-edge region, then it is counted as a false positive (FP). Edge pixels reported in a dont care region do not count as TPs or FPs. Background pixels that match pixels in a GT no-edge region are counted as true negatives (TN). Background pixels that match GT edge pixels are counted as false negatives (FN).

For the pixel-based comparison, these similarity measures were used:

– The percentage correct classification (PCC):

$$PCC = \frac{TP + TN}{TP + TN + FP + FN} \tag{12}$$

– The Jaccard Coefficient:

$$Jaccard = \frac{TP}{TP + FP + FN} \tag{13}$$

– The Dice Coefficient:

$$Dice = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{14}$$

These three measures yield different properties: The PCC measure describes the proportion of matches to the total number of (pixels). Jaccard measure is an overlap ratio which excludes all non-occurrences, and, thereby, disregards the information on matches between background pixels. The Dice measure is similar to Jaccord but it gives more weight to occurrences of edge pixels (TPs).

From each test image we extract three edge maps. The first one is obtained using our method. The second and third, using canny edge detection with parameters selected with the process described in [6], with EGT estimated with ROC analysis and best parameter selection (PS) using ROC analysis and Chi-Square test respectively.

In table 1 we present the results of pixel-based comparison to the ground truth images. Larger values of *PCC*, *Jaccard* coefficient and *Dice* coefficient indicate greater similarity to the GT images. From the comparison of the measurments we conclude that for this GT dataset, the method proposed in this paper performs better compared to the method of detector parameter selection proposed in [6].

Table 1. Results of pixel-based evaluation. Larger values indicate better performance.

GT evaluation results				
		Our method	PS: ROC analysis	PS: Chi Square test
Image 1	PCC	0.520055	0.477554	0.484051
	Jaccard	0.111810	0.033120	0.045144
	Dice	0.201133	0.064117	0.086389
Image 2	PCC	0.509625	0,496870	0,497209
	Jaccard	0.065781	0,041458	0,042076
	Dice	0,123443	0,079608	0,080754
Image 3	PCC	0,528503	0,518710	0,520773
	Jaccard	0,127268	0,108964	0,112783
	Dice	0,225799	0,196515	0,202704
Image 4	PCC	0,533181	0,511978	0,511269
	Jaccard	0,156842	0,118467	0,117178
	Dice	0,271156	0,211838	0,209775
Image 5	PCC	0,520935	0,504178	0,510735
	Jaccard	0,092413	0,059962	0,072394
	Dice	0,169191	0,113140	0,135014
Image 6	PCC	0,523620	0,511709	0,515407
	Jaccard	0,120254	0,098250	0,105069
	Dice	0,214691	0,178922	0,190159

References

1. J. Canny: A computational approach to edge detection, IEEE Transactions on pattern analysis and machine intelligence 8 (6) (1986) 679–698.
2. Lily Rui Liang, Carl G. Looney: Competitive fuzzy edge detection. Applied Soft Computing 3 (2003) 123-137
3. D. Marr, E. Hildreth: Theory of edge detection, Proc. Roy. Soc. London B-207 (1980) 187–217.
4. T. Kohonen, Self-Organizing Maps, 2nd edition, Springer, Berlin, 1997.
5. P.J. Toivanen, J. Ansamaki, J.P.S. Parkkinen, J. Mielikainen: Edge detection in multispectral images using the self-organizing map. Pattern Recognition Letters 24 (2003) 2987-2994
6. Yitzhak Yitzhaky, Eli Peli: A Method for Objective Edge Detection Evaluation and Detector Parameter Selection. IEEE Transactions on pattern analysis and machine intelligence 25 (8) (2003) 1027–1033
7. Kevin Bowyerm,Christine Kranenburg : Edge Detector Evaluation Using Empirical ROC Curves. Computer Vision and Image Understanding 84(2001)77-103
8. Todd Law, Hidenori Itoh, Hirohisa Seki: Image Filtering, Edge Detection and Edge Tracing Using Fuzzy Reasoning. IEEE Transactions on pattern analysis and machine intelligence 18 (5) (1996) 481–491
9. William K. Pratt: Digital Image Processing, Third Edition, John Wiley & Sons, Inc. (2001)
10. Armando J. Pinho: Modeling Non-Linear Edge Detectors Using Artificial Neural Networks. Proc. of the 15th Annual Int. Conf. of the IEEE Eng. in Medicine and Biology Soc.,San Diego, CA, U.S.A. October 1993, pp 306-307.
11. S. Weller: Artificial Neural Net Learns the Sobel Operators (and More), Applications of Artificial Neural Networks II , SPIE Proceedings Vol SPIE-1469 pp 69–76, Aug 1991.
12. J.C. Bezdek and D. Kerr: Training Edge Detecting Neural networks with Model-Based Examples, Proc 3rd International Confererence on Fuzzy Systems, FUZZ-IEEE'94, Orlando, Florida, USA. pp 894–901, June 26 - 29, 1994.