Edge Detection in Noisy Images Using the Support Vector Machines

Hilario Gómez-Moreno, Saturnino Maldonado-Bascón, Francisco López-Ferreras

Signal Theory and Communications Department. University of Alcalá
Crta. Madrid-Barcelona km. 33,600 D.P. 28871
Alcalá de Henares - Madrid (Spain)
`{hilario.gomez, saturnino.maldonado, francisco.lopez}@uah.es`

Abstract. In this paper, a new method for edge detection in presence of impulsive noise based into the use of Support Vector Machines (SVM) is presented. This method shows how the SVM can detect edge in an efficient way. The noisy images are processed in two ways, first reducing the noise by using the SVM regression and then performing the classification using the SVM classification. The results presented show that this method is better than the classical ones when the images are affected by impulsive noise and, besides, it is well suited when the images are not noisy.

1 Introduction

The edge detection methods are an important issue for a complete understanding of the image. The most usual classical methods search for several ways to perform an approximation to the local derivatives and they mark the edges by searching the maximum of these derivatives. The Sobel, Prewitt or Roberts filters are some of these approximations [1]. Other approaches are based in a previously smoothing action in order to improve the edge detection and to reduce the effect of noise (basically Gaussian noise) [2]. This pre-filtering process seems to be not adequate when the noise is impulsive and most edge detectors are very sensitive to the pulses of noise.

In this paper we present a way to detect the edges by using a new point of view. We do not try to approximate the derivative or to use other mathematics methods. The main idea in this work is to train the computer to recognize the presence of edges into an image. In order to perform this idea we use the Support Vector Machines (SVM) tool that is given good results in other classification problems.

The training is performed using a few own-created images that represents a clearly defined edges with the edges easily located.

The noise reduction process is performed using the SVM. In this case we use the information in a 3x3 neighborhood of each pixel to make a regression process and then replace the pixel value with the regressed value.

The results obtained with no noise are similar to those from previous methods and clearly superior when compared to some classical methods in noisy images.

2 SVM Classification and Regression

There are several ways to classify, Bayesian decision, neural networks or support vector machines, for example. In this work we use the SVM classifier since this method provides good results with a reduced set of data and then we do not require an intensive training like another methods. Thus the SVM gives us a simple way to obtain good classification results with a reduced knowledge of the problem. The principles of SVM have been developed by Vapnik [3] and they are very simple.

In the decision problem we have a number of vectors divided into two sets, and we must find the optimal decision frontier to divide the sets. The frontier chosen may be anyone that divides the sets but only one is the optimal election. This optimal election will be the one that maximizes the distance from the frontier to the data. In the two dimensional case, the frontier will be a line, in a multidimensional space the frontier will be an hyperplane. The decision function that we are searching has the next form,

$$f(\mathbf{x}) = \langle \mathbf{w} \cdot \mathbf{x} \rangle + b = \sum_{i=1}^{n} w_i x_i + b \quad . \tag{1}$$

In (1), **x** is a vector with n components that must be classified. We must find the vector **w** and the constant b that makes optimal the decision frontier. The basic classification process is made by obtaining the sign of the decision function applied to the given vector, a positive value represents the assignment to one class and a negative one represents the assignment to the another class.

Normally, we use another form of this decision function that includes the training input and output vectors information [4]. This new form is,

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i \langle \mathbf{x}_i \cdot \mathbf{x} \rangle + b \quad . \tag{2}$$

The *y* values that appear into this expression are +1 for positive classification training vectors and –1 for the negative training vectors. Besides, the inner product is performed between each training input and the vector that must be classified. Thus, we need a set of training data (**x**,**y**) in order to find the classification function and the α values that makes it optimal. The *l* value will be the number of vectors that in the training process contribute in a high quantity to form the decision frontier. The election of these vectors is made by looking at the α values, if the value is low the vector is not significant. The vectors elected are known as support vectors.

Normally the data are not linearly separable and this scheme can not be used directly. To avoid this problem, the SVM can map the input data into a high dimensional feature space. The SVM constructs an optimal hyperplane in the high dimensional space and then returns to the original space transforming this hyperplane in a non-linear decision frontier. The non-linear expression for the classification function is given in (3) where *K* is the non-linear mapping function.

$$f(\mathbf{x}) = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{x}_i \cdot \mathbf{x}) + b \ . \tag{3}$$

The choice of this non-linear mapping function or kernel is very important in the performance of the SVM. The SVM applied uses the radial basis function to perform the mapping, since the others proved do not work appropriately. This function has the expression given in (4).

$$K(x, y) = \exp\left(-\gamma (x-y)^2\right) . \tag{4}$$

The γ parameter in (4) must be chosen to reflect the degree of generalization that is applied to the data used. The more data is obtained the less generalization needed in the SVM. A little γ reflects more generalization and a big one represents less generalization. Besides, when the input data is not normalized, this parameter performs a normalization task.

When some data into the sets can not be separated, the SVM can include a penalty term (C) that makes more or less important the mismatch classification. The more little is this parameter the more important is the misclassification error. This term and the kernel are the only parameters that must be chosen to obtain the SVM.

The classification scheme may be easily extended to the case of regression. In this case the idea is to train the SVM by using *y* values different from +1 and –1. The values used are the values known from the function to be obtained. Then, we search for an approximation function that fits approximately the known values [4].

3 Noise Reduction Training

In this section, we explain how the SVM regression may be used to reduce the impulse noise into the images in order to apply an edge detection algorithm without noise interferences.

The main idea is to replace the pixels into noisy image with a new value avoiding the noise. In order to obtain the new pixel value we use the SVM regression. The regression training process is shown in Fig. 1.

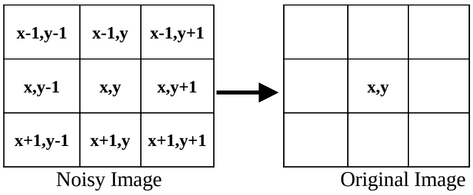


Fig. 1. Pixel regression

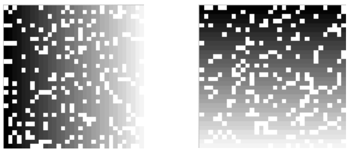


Fig. 2. Example of noisy training images

For each pixel (x,y) of the noisy image we form a vector containing the values of the pixels around it (including the pixel itself). This vector will be the input to the regression function. The output will be the value of the pixel (x,y) of the original image.

The next step is to find the images to be used into the training process. The images elected must be simple to avoid an excessive training time but significant enough to give good results. The option elected was to make controlled images given a gray scale and with random noise in a known position. In Fig. 2 an example is presented.

In this training images we must control the size and the noise ratio. The images in Fig. 2 have a 32x32 size and a 20 percent noise (20 percent pixels of the image are noise). If we use these two images in the training process the number of vectors processed will be 1953. The number of support vectors after the training process is 634 in this case.

4 Edge Detection Training

In this section we present a way to detect edges by using the SVM classification. In this case, the decision needed is between "the pixel is part of an edge" or "the pixel is not part of an edge". In order to obtain this decision we must extract the information needed from the images. In this work a vector is formed for each pixel, given the difference between this one and the pixels in a 3x3 neighborhood around it. This way an eight components vector is calculated at each pixel except for the border of the image, because in this case the differences can not be calculated. The vectors formed are used as inputs to the SVM training.

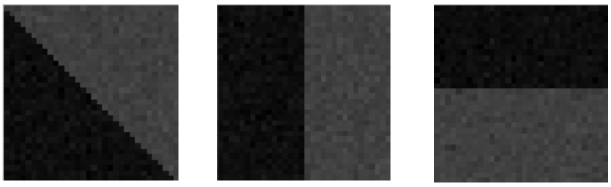


Fig. 3. Training images for edge detection

The images used to train the SVM are shown in Fig. 3. They are images created by trying to obtain a good model for the detection. The only edges used in the training are vertical, horizontal and diagonal ones and we expect that the other edges will be generalized by the SVM. The pixels considered as edges are those into each image that are in the border between bright and dark zones, i.e. the points in the dark zone near the bright one and vice versa.

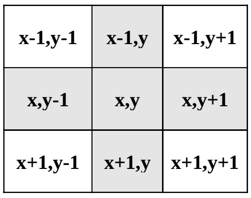
The dark and bright zones are not homogenous but the intensity at each pixel is a random value (gaussian values). The values at each zone never reach those of the other zone. By using these random values we try to simulate the no homogenous surface into a real image. The random nature of the training images makes random the SVM training, for example, the number of support vector may change every time we use different images. But the results obtained are very similar in all cases.

A value that must be set in the training process is the mean difference between dark and bright zones, since this parameter controls the sensibility of the edge detector. A little difference makes the detector more sensible and a greater one reduces this sensibility.

5 Edge detection method

When we apply the trained SVM (3) to an image, a value for each pixel into the image is obtained. This value must be (ideally) a value positive or negative near 1. We can use the sign of these values to say when a pixel is an edge or not but, this way, a lost of information is produced. It is better to use the values obtained and say that there is a gradual change between “no edge” and “edge”. Then, the value obtained indicates a probability of being an edge or not.

After the process above we must decide the pixels that are considered as edges. This task can be simplified when we have the edges separated into vertical and horizontal directions. In this case we search for the pixels that are local maximum in both directions and the edge image is obtained. The method proposed is able to obtain these vertical and horizontal edges by using vectors formed only with the horizontal and vertical differences as input to the SVM (Fig. 4).



x-1,y-1	**x-1,y**	**x-1,y+1**
x,y-1	**x,y**	**x,y+1**
x+1,y-1	**x+1,y**	**x+1,y+1**

Fig. 4. Vectors for vertical and horizontal detection

Fig. 5 shows and example of the process. The original image "house" is a 256x256 eight bits gray-scaled image. The edge images presented are gray-scaled images where the values from SVM have been translated to gray values, 255 represents edge detection and 0 represents no edge detection. The intermediate values show a gradation between these extreme values. The parameters set for this edge detection were γ = 8e-4, C = 1000, and the mean difference between dark and bright zones in the training images was 50. Besides, in order to reduces the isolated points considered as edges we use only the values above a given threshold. In the examples the threshold is set at 32 in an eight bits gray scale.

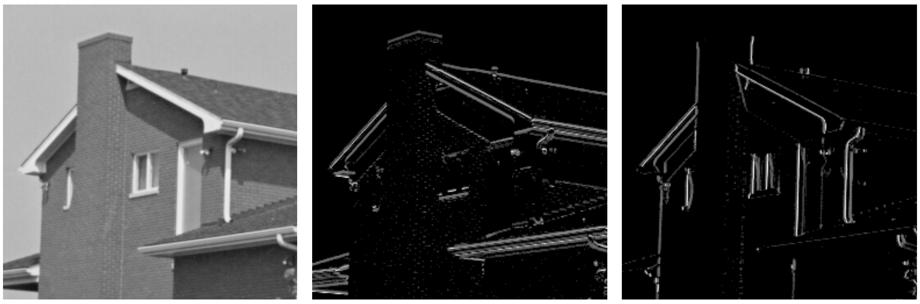


Fig. 5. Horizontal and vertical edge detection example.

Fig. 6 shows a comparison between the Canny edge detector and the detector proposed here. The image used as example presents a complex texture in the walls due to bricks. Thus, the parameters of the Canny detector have been set trying to reduce the false edge detection of this bricks.

We can see how the performance of the method proposed is similar to the Canny edge detector (considered as an standard) although the SVM method is not using any blurring filter like in the Canny edge detector.

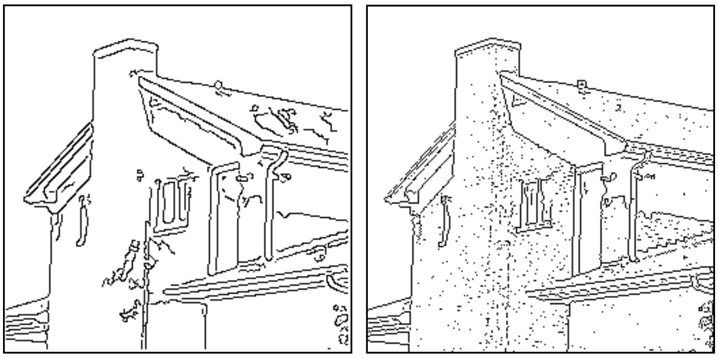


Fig. 6. Comparison of edge detection. (a) Canny edge detector. (b) SVM edge detector.

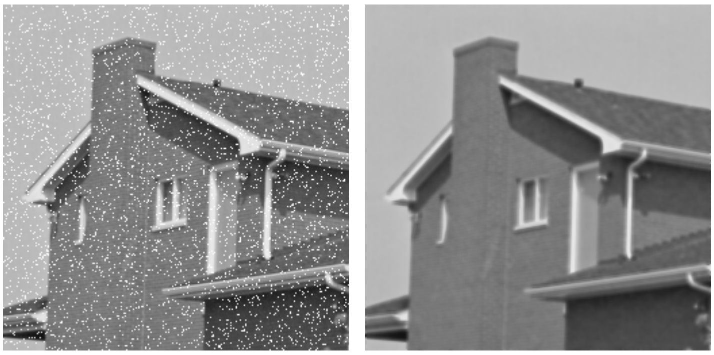


Fig. 7. Example of noise reduction. (a) Noisy image (b) No noise image

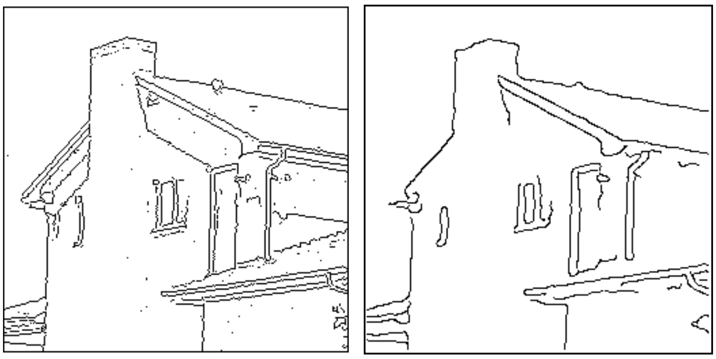


Fig. 8. Comparison of edge detectors. (a) SVM detector (b) Canny edge detector.

6 Results

The results presented here have been obtained by using the LIBSVM [5] as implementation for the SVM. The programs used were written in C++ and compiled using the Visual C++ 6.0 compiler. The computer used has a Pentium III processor with 128 Mbytes RAM.

First we show how the noise reduction process performs. The training process uses images like in Fig. 2, 32x32 gray-scaled images with a 20 % impulsive noise. In this case the impulsive noise has white pulses only but the method may be applied in a similar way to “salt&pepper” noise. The γ parameter of the kernel is elected to obtain the best results, in this case the value elected is γ = 3e-5. The C parameter must be elected to obtain accurate results with a reduced number of support vectors. In this work the parameter has a value C = 1000. A greater value increases the training time and a more little value decreases the accuracy of the process. The training process takes about 10 minutes in the case presented. The noise reduction process can be viewed in Fig. 7. Fig. 7-a shows a corrupted image with impulsive noise of 10 % and Fig. 7-b shows the image after the noise reduction process explained here.

The example shows that this reduction process is not well suited for denoising since the recovered image is blurred. However, the impulsive noise disappears from the image and the edge detection process can be performed. Besides, the blurring is useful in the edge detection since the textures into the images are more uniform after blurring and the false detection is reduced.

The next step is to perform the edge detection like previously explained. The kernel parameter in this case was γ = 8e-4 and the parameter C = 1000 like in the previous case. The threshold used in this case is 16. In Fig. 8 is shown a comparison between the SVM method and the Canny edge detector. The parameters of the Canny edge detector have been set in order to reduce to the maximum the detection of the impulsive noise like edges. It is clearly that the SVM method is superior in this case. The only drawback of the method proposed is that the execution takes over 15 seconds while the Canny method takes 1 second.

7 Conclusions

This work shows how the SVM performs the edge detection in presence of impulsive noise in an efficient way and given good results.

The comparison between the edge detection method proposed here with other known methods shows that the SVM method is superior when the impulsive noise is present and it is similar (although slightly inferior) when the image is not noisy.

We think that the method may be improved in execution speed and performance by optimizing the C++ programs and by using more differences between adjacent pixels to make edge detection, e.g. using 5x5 windows around a pixel.

References

1. A.K. Jain. "Fundamentals of Digital Image Processing". Englewood Cliffs, NJ, Prentice Hall, 1989.
2. J.F. Canny. “A computational Approach to Edge Detection”. IEEE Trans. On Pattern Analysis and Machine Intelligence, vol. 8, pp. 679-698. 1986.
3. V. Vapnik. "The Nature of Statistical Learning Theory". New York, Springer-Verlag, 1995.
4. N. Cristianini and J. Shawe-Taylor. "An introduction to Support Vector Machines and other kernel-based methods". Cambridge, Cambridge University Press, 2000.
5. C.C. Chang and C.J. Lin. "Libsvm: Introduction and benchmarks". http://www.csie.ntu.edu.tw/~cjlin/papers.