Neural-Networks-Based Edges Selector for Boundary Extraction Problems

Horacio M. González-Velasco, Carlos J. García-Orellana, Miguel Macías-Macías, F. Javier López-Aligué, and M. Isabel Acevedo-Sotoca

Departamento de Electrónica e Ingeniería Electromecánica
Universidad de Extremadura
Av. de Elvas, s/n. 06071 Badajoz - SPAIN
horacio@nernet.unex.es

Abstract In the present work, a neural-networks-based system is presented that makes it possible to reduce, when generating edge maps to be used in an object boundary detection problem, the number of edges that are not due to the object itself, but to the background. Starting from a conventional edge detection, the selection is carried out by a neural-networks-based classifier, which is trained using examples. As a test for the system, the application to bovine livestock images is presented, from which we want to extract the boundary of the animal.

1 Introduction

Edge detection is one of the fundamental operations in digital image analysis. This stems from the fact that edges characterize the boundaries of the image objects, and are therefore the base for subsequent segmentation systems. From the different boundary extraction techniques, those based on deformable models [1,2] and genetic algorithms search [3] stand out because of their success. In these methods, a parametric contour of the searched object is placed by optimizing a certain energy function, which is based on an edge map of the image [2,4,5].

There are three fundamental properties that a good edge detector must present [6]: to detect *all* the edges present in the image (absence of *false positives*), not to detect any edge that is not in the image (absence of *false negatives*) and, finally, that the position of the detected edge pixels is the correct one. Nevertheless, it is important, for the good operation of the above-mentioned boundary extractors, that the edge maps does not contain many more edges than those corresponding to the searched object. But this is not always possible, mainly in those images where backgrounds can not be controlled. Thus, in those cases it would be useful to have a system to *classify* the edges detected in the image, retaining only those which more likely correspond to the object itself, and discarding those corresponding to objects in the background.

In this work, a system for the edges selection, based on a multilayer perceptron neural network, is presented. This network is trained using a set of images for which the edges of the searched object are known. The selection system will

act over a previously obtained edge map, recognizing those edges that have their origin in the object we are looking for, according to the learned criterion. Although there exist many previous papers where NN techniques are applied to the problem of edge detection [7,8,9], they are in general focused on the enhancement step, trying to obtain a good detector in the previously mentioned sense. However, in this work it is not our interest to detect edges by means of the neural network, but to select those that serve for the subsequent processing step, once detected.

In the two following sections the proposed selection system is described, along with its integration within the global detector. After that, in section 4, results from a concrete application of the method are presented, while in section 5 there is a brief discussion about the method and its possible improvements.

2 System overview

In general, the selector proposed can be integrated in any kind of conventional edge detector, between enhancement and thresholding stages (see figure 1). However, since the direction of the line orthogonal to the edge is needed by the selector, the use of edge detectors based on the first derivative (gradient) is convenient. It must be followed by a nonmaximum suppressor, with which best-located one-pixel-width edges are obtained.

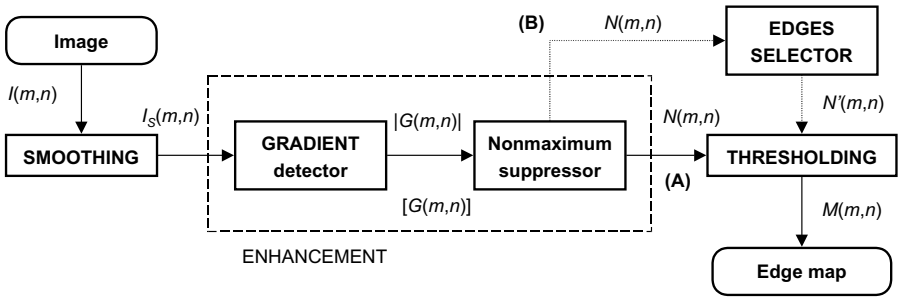


Figure 1. Block diagram representing the type of edge detector where the proposed selection system is integrated (path B in the figure)

If the edges selector is not used (path A in figure 1), the more strongly detected edges are obtained. However, in many applications this is not our interest, as in that described in section 4. When we use path B, high values are obtained, at the output of the selector, for the edges which are similar to those learned from the examples, whichever its strength. So, the edge map produced through the thresholding process will contain the edges which more likely correspond to the learned ones.

To make the training of the selector, K images $I_k(m,n)$ with $k \in \{1,\cdots,K\}$ (where $I_k(m,n) \in [0,1]$) are used, for which their corresponding reference edge

maps $R_k(m,n)$ are available. In those maps only the edges corresponding to the object can be found ($R_k(m,n) = 1$ for edge pixels, and 0 otherwise). Examples of these images are presented in figures 4 and 5.

3 Neural-networks-based selector

By means of the proposed selection system (followed by thresholding) we intend to classify edge pixels, previously detected in the enhancement step, into two classes: *correct* edges (i.e. corresponding to the searched object) and *erroneous* edges (which do not correspond to it). Thus, among the edges detected in the image, the system will *select* only those pixels classified as correct edges, rejecting those considered as incorrect.

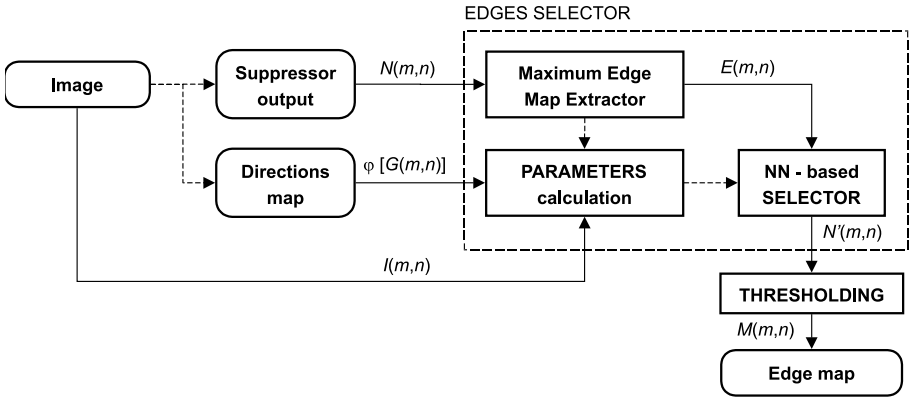


Figure 2. Diagram describing the selector system proposed, where, along with the nonmaximum suppressor output, the image itself and the directions of the gradient are represented as inputs, because they are needed to calculate parameters.

The classification process needs to be made through the following three steps, graphically represented in figure 2:

1. **Extraction of a maximum edge map:** In the first place, we have to determine what pixels have to be classified, for which the edge detection made in the previous stage is used. Since there could be, a priori, weak and strong edges in the contour we wish to extract, we are interested in classifying all those pixels that are detected as edge, whatever their strength. Thus, the pixels to be classified will be determined by thresholding the nonmaximum suppressor output image $N(m,n)$, using for the threshold a low value. This is called *maximum edge map* $E(m,n)$.
   In our application, we used a threshold low enough to permit at least 95% of the pixels of $N(m,n)$ without zero value to be considered as edges, rejecting

the 5 % of weaker detected ones, which likely correspond to noise introduced in the image capture process. In figures 4 and 5, examples of maximum edge maps (along with their corresponding original images) are presented. It must be mentioned that using the nonmaximum suppressor produces one-pixel-width edges, which reduces the number of pixels to be processed.

2. **NN processing:** Once the edge pixels to be classified are determined, a set of P parameters (see section 3.1) is calculated for each one, related to its position and the gray profile around it in the original image. Those parameters are used as inputs for a multilayer perceptron, with one hidden layer and two neurons at the output, which give us the membership to each class: $y_c(m,n)$ for correct edges and $y_e(m,n)$ for erroneous ones.

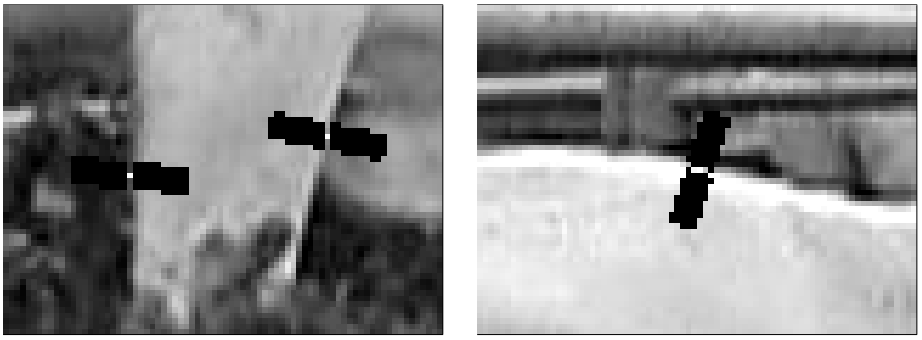
   The grey-level image $N'(m,n)$ has then a value proportional to $y_c(m,n) - y_e(m,n)$ for those pixels detected as edges, and 0 for the rest. Therefore, pixels with larger values correspond with correct edges, and vice versa, so by the thresholding process, the "more correct" edges will be obtained.
3. **Final thresholding:** This stage, placed outside the selector (see figures 1 and 2), is responsible for pointing out as edges those pixels of $N'(m,n)$ with larger intensity (correct edges). In this process it is very important the value of the threshold used, because it determines the number of edge pixels in the final image. So, a value such that the number of edge pixels in $M(m,n)$ is similar to the average number of edges in the reference images $R_k(m,n)$ is proposed.

Regarding the process described above, two aspects require further explanation: parameters used as input to the neural network and the training process.

3.1 Input parameters for the NN

If an object has, more or less, a uniform color (and, therefore, a gray-level), a good method to identify which their edges are consists of analyzing the gray-level profile in their orthogonal direction. For that reason, we propose to use, as inputs to the neural network, certain statistical parameters calculated over two rectangular windows in the direction orthogonal to the edge, located on each side of it, as shown in the examples of figure 3. Particularly, in the application of section 4, the mean and the variance of gray levels have been used. With the mean we have an idea of the gray values at each side of the pixel, while the variance will indicate the uniformity of the grey levels in the windows.

Furthermore, in those applications where the object of interest has a well-defined shape and a position that varies within certain limits, it is advantageous to use also as parameters the position of the pixel and the direction orthogonal to the edge, since finding edges of the object around certain positions and with a concrete direction should be expected. This is the case of the example in section 4.



(a) (b)

Figure 3. In this figure, examples of edge pixels (white) and the windows of 10×5 size in the direction orthogonal to the edge (black) are presented.

3.2 Training

In order to train the network we need prototypes for each of the two proposed clases. Considering that the set we are classifying is formed by those pixels detected as edges, the prototypes extraction process for each training image will consist of three steps:

- First, for the image $I_k(m,n)$ a maximum edge map $E_k(m,n)$ is obtained, using the method described above.
- Later, the set $P_{C,k}$ is defined as

$$P_{C,k} = \{(m_c, n_c) \,|\, E_k(m_c, n_c) = R_k(m_c, n_c) = 1\} \tag{1}$$

  so it contains the k-th image pixels detected as edges that belong to the considered object (prototypes for the class of correct edges). The set

$$P_{I,k} = \{(m_i, n_i) \,|\, E_k(m_i, n_i) = 1 \text{ y } R_k(m_i, n_i) = 0\} \tag{2}$$

  is also defined, which contains the pixels detected as edges that do not belong to the object (prototypes for the class of erroneous edges).
- Finally, for each pixel of both sets, the parameters are calculated, to be used in the training of the network.

Once obtained the prototypes, the whole set is split into three subsets (at the level of images, to obtain a better generalization): training, validation and test, and the process starts training one perceptron with very few neurons in the hidden layer. The training is carried out using the training subset and the RPROP algorithm described in [10], and stops when the number of misclassifications obtained over the validation set goes through a minimum. Later, we repeat this process by increasing the network size and we choose the new configuration as optimal if the number of misclassifications over the validation set is lower than the previous one.

4 Experiments and results

The previously described technique was applied to the generation of edge maps for bovine livestock images (figure 4, first image), with the final aim of finding the boundary of the animal through a genetic search, using the method described in [11]. To this end we had 45 images for which their corresponding references were generated. In the first place, the training process and its results will be described, and later the results obtained when applying the edge detector to the set of test images, with and without selector, are presented.

4.1 Training

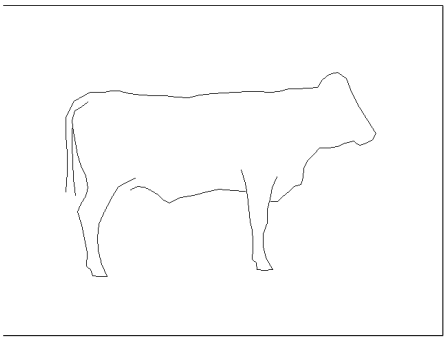
As stated before, the 45 images were divided into three subsets: 27 were used for training, 9 for validation and 9 for test. From all the generated prototypes, the training of the network was carried out using 60.000 from the training subset and 20.000 from the validation and tests subsets, uniformly distributed between classes. Using this sets, networks were trained with a number of neurons in the hidden layer between 20 and 55, obtaining the best results with 46 neurons. For this configuration, 89.5%, 87.5% and 88.1% success rate for correct edges were obtained, and also 87.0%, 85.3% and 85.1% for erroneous edges, considering the training, validation and test sets, respectively.

4.2 Edge detection

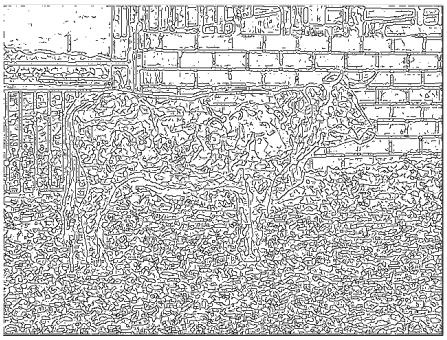
After training, the edge detector was applied, with and without selector, to the 9 images of the test subset. To determine the effectiveness of both configurations quantitatively, a figure of merit was proposed, based on the one defined by Pratt [6] for conventional edge detectors. That figure of merit considers the three aspects that a good edge detector must observe, cited in section 1. However we used $R_k(m, n)$ images as references, which do not contain all the edges of the image, but only those corresponding to the searched object. Thus, concepts of false positive and false negative are modified: now a detected edge which do not correspond to the object is a false positive (though the edge exists in the image) and a non-detected edge of the object (which generally exists) is a false negative. Therefore, this figure of merit has a large value when many edges are detected from the boundary of the object and few from the rest of the image, resulting 1 for the reference.

The results obtained for test images are shown in table 1. It can be seen that the obtained value for the system with selector is significantly larger in all cases, which indicates a great improvement with regard to the system without selector. In figures 4 and 5, two examples are presented, where the outputs of the detector for both cases, along with the original image, are shown. As can be seen, while in the results without selector many edges are observed due to the objects in the background, they disappear almost completely when the selector is applied, obtaining then a more suitable edge map for the genetic search.

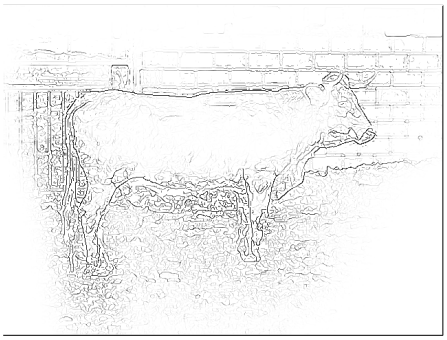
Original image L17



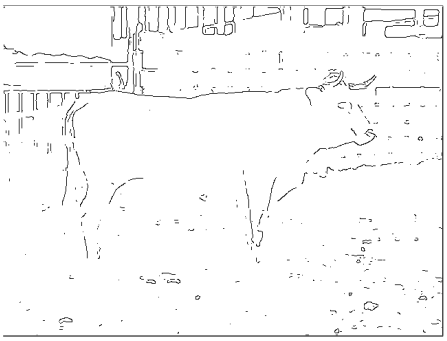
Reference R_{17}



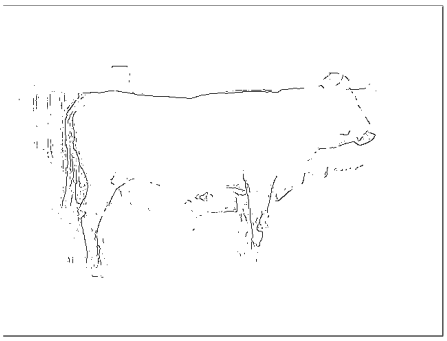
Maximum edge map E_{17}



Selector output N'_{17}



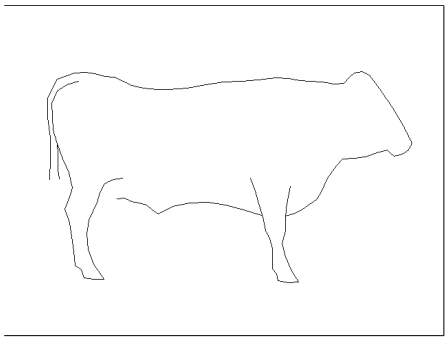
Result without selector



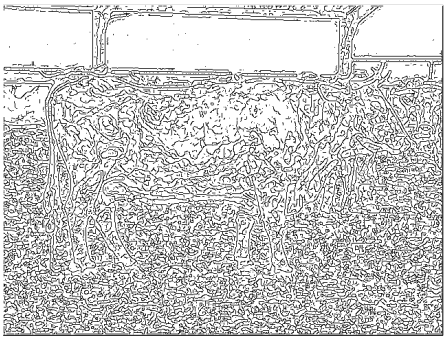
Result with selector

Figure 4. In this figure, an example of the images used and generated by the selector system are shown, along with the final results, with and without selector. Apart from the first one, the images has been inverted to improve presentation (contours must be white and background black).

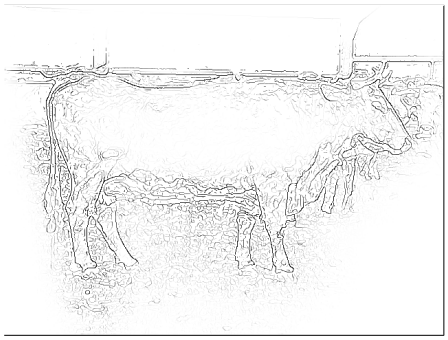
Original image L26



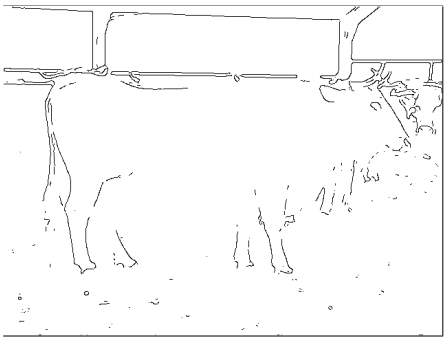
Reference R_{26}



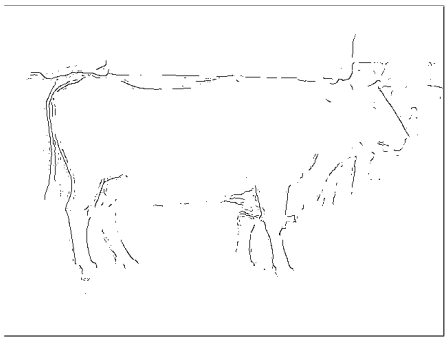
Maximum edge map E_{26}



Selector output N'_{26}



Result without selector



Result with selector

Figure 5. Another example of the images used and generated by the selector system is presented, along with the final results, with and without selector. Apart from the first one, the images has been inverted to improve presentation (contours must be white and background black).

Table 1. Results for the edge detector, with and without selector, when applying to images in the test subset, showing the percentage of improvement.

Image	***Without sel.***	***With sel.***	**% improv.**
L01	0.387	0.440	13.6
L04	0.285	0.375	31.4
L05	0.231	0.329	42.5
L12	0.184	0.372	101.9
L17	0.188	0.489	160.3
L18	0.186	0.391	110.2
L26	0.238	0.398	66.9
L34	0.247	0.308	24.7
L41	0.218	0.367	68.3

5 Conclusions and future research

In this work, a selection system has been presented that, starting from a conventional edge detection, eliminates the influence of background edges partially, which is particularly important when the edge map is used to extract the boundary of an object, mainly if deformable models or genetic algorithms techniques are applied.

The requirement of needing reference images can be seen as a serious drawback for the use of this technique, because its generation could be hard and tedious. Nevertheless, if we are using parametric shape models for the subsequent boundary extraction, as PDM [3,12], the labelling of the points (which could be obtained with semi-automatic methods) can be used to define the reference images too, linking the points with lines.

We are now trying to improve the selector in two aspects. On one hand, other parameters are being considered to characterize the windows, obtained through parameter extraction techniques, based on PCA and ICA, applied to windows data. On the other hand, we are considering to apply our technique to color images, which have more information to identify whether an edge corresponds to the searched object or not.

Acknowledgements

This work has been supported in part by the Junta de Extremadura through project 2PR01A007. We are also indebted to the staff of the *Centro de Selección y Reproducción Animal* (CENSYRA) of Badajoz for their technical assistance with the cattle and the photographs.

References

1. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. Journal of Computer Vision **1** (1988) 321–331
2. Cohen, L.D., Cohen, I.: Finite-element methods for active contour models and balloons for 2-d and 3-d images. IEEE Tran. on Pattern Analysis and Machine Intelligence **15** (1993) 1131–1147
3. Hill, A., Taylor, C.J.: Model-based image interpretation using genetic algorithms. Image and Vision Computing **10** (1992) 295–300
4. Toet, A., Hajema, W.P.: Genetic contour matching. Pattern Recognition Letters **16** (1994) 849–856
5. Toet, A.: Target detection and recognition through contour matching. Technical report, CALMA (Combinatorial Algorithms for Military Applications) project (1994)
6. Parker, J.R.: Algorithms for image processing and computer vision. John Wiley (1996)
7. Srinivasan, V.: Edge detection using neural networks. Pattern Recognition **27** (1994) 1653–1662
8. Wong, H.S., Caelli, T., Guan, L.: A model-based neural network for edge characterization. Pattern Recognition **33** (2000) 427–444
9. Suzuki, K., Horiba, I., Sugie, N.: Neural edge detector – a good mimic of conventional one yet robuster against noise. Lecture Notes in Computer Science **2085** (2001) 303–310
10. Riedmiller, M., Braun, L.: A direct adaptive method for faster backpropagation learning: the RPROP algorithm. In: Proc. IEEE International Conference on Neural Networks. (1993) 586–591
11. González, H.M., García, C.J., Macías, M., Acevedo, M.I.: GA techniques applied to contour search in images of bovine livestock. Lecture Notes in Computer Science **2084** (2001) 482–489
12. Cootes, T.F., Hill, A., Taylor, C.J., Haslam, J.: The use of active shape models for locating structures in medical images. Image and Vision Computing **12** (1994) 355–366