New Application for the Edge Detection Algorithm

VLADISLAV SKORPIL, JIRI STASTNY
Department of Telecommunications
Department of Automation and Computer Science
Brno University of Technology
Purkynova 118, 612 00 BRNO, Czech Republic,
CZECH REPUBLIC

Abstract: The wavelet transform is a comparatively new and fast developing method for analysing signals. The main advantage of applying the wavelet transform to the detection of edges in an image is the possibility of choosing the size of the details that will be detected. The size of detected edges is set by the wavelet scale. In the case of the discrete wavelet transform the choice of the scale is performed by multiple signal passage through the wavelet filter. When processing a 2-D image, the wavelet analysis is performed separately for the horizontal and the vertical function. The vertical and the horizontal edges are thus detected separately. The wavelet transform will split the input signal into two components. One contains the low-frequency (LP) part of input signal, which corresponds to major changes in the function (individual objects in the image, etc.). The other part contains the high-frequency (HP) part of input signal, which corresponds to details in the function (noise, edges, etc.). This signal component is not processed on the next level of transformation.

Key-Words: - Wavelet transform, Edge detection, Image, Signal, Analysis

1 Introduction

For edge detection in an image, a computer application has been created that tests the two-dimensional scene. A real technological scene was simulated by digitising five selected objects. Two-dimensional images of three-dimensional objects were prepared for this purpose. The intention was to test such objects that resembled two-dimensional images of real objects. Objects similar in shape were also selected on purpose.

2 Edge Detection

Edge detection can thus be performed in two ways. Edges can be detected by an analysis using either the (HP) or (LP) output of the wavelet transform. In the image, the edges will show differently in each component.

In the (LP) function the edge will show by the function passing through zero. Thus it corresponds to the second derivative of the input function. In a discrete image, edge detection with the help of this signal component is performed on the basis of detecting the change in the signs of two consecutive values of the (LP) function.

The other possibility of detecting edges consists in making use of the (HP) function. In this case the edges will show as maxima (peaks, pulses). Detection is then based on recognizing the individual maxima of the function. Here it is necessary to choose a threshold that will determine the minimum size of a local maximum that can be regarded as an edge. An algorithm seeks to find such a group of three consecutive values of the function for which it holds that $x_{i-1} < x_i > x_{i+1}$ and simultaneously also $x_i >$ threshold.

2.1 Comparison of Results

The attached application contains the programmed CDF 1,1; CDF 1,3; CDF 1,5; CDF 2,2; CDF 2,4; CDF 2,6 and D4 wavelets. Either (LP) or (HP) function analysis can be chosen for edge detection. For the detection using the (HP) function the threshold value can be chosen. A significant improvement in the results of edge detection can be obtained via preprocessing the image by one of the four filters (or their combination). The attached program environment thus offers a comparatively great number of function combinations for image analysis. An example of a technological scene is shown in Fig.1. The results of individual wavelets are evaluated below.

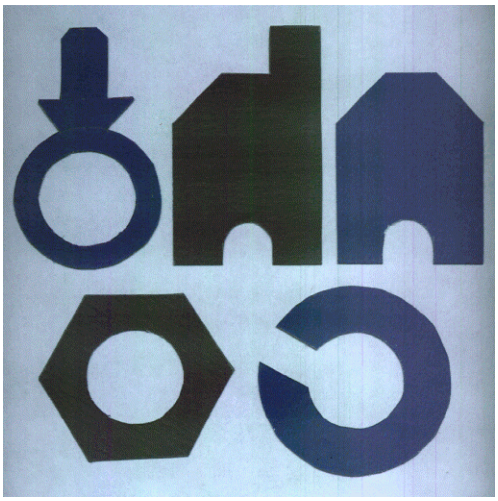

Fig.1 Example of a technological scene.

2.1.1 Edge detectors utilizing CDF1,X

They have a good edge resolution and low sensitivity to image noise. They can be applied to the attached technological scenes even without preprocessing (filtering). In the case of the CDF (1,5) filter there appears considerable edge duplication (Fig.2, Fig.3), which is very inconvenient for subsequent operations.

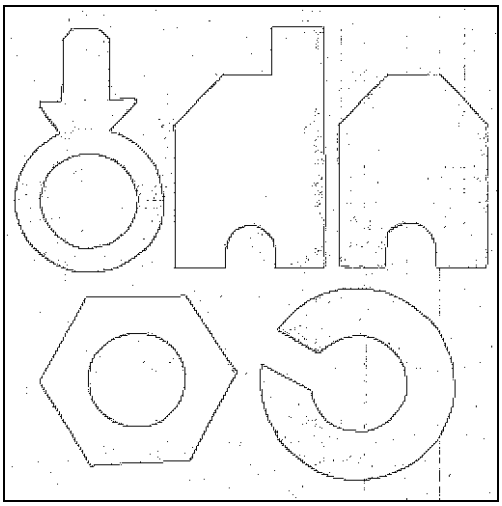

Fig.2 Edges detected by CDF (1,1).
No filtering prior to detection.

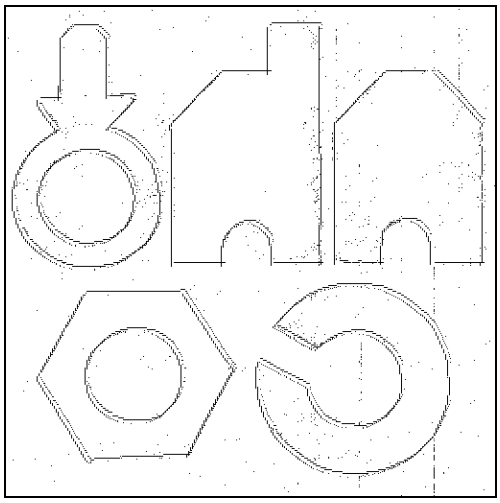

Fig.3 Edges detected by (1,5).
No filtering prior to detection.

2.1.2 Edge detectors utilizing CDF 2,X

In the case of these wavelets a greater sensitivity to image noise is apparent. For practical application it is therefore in most cases necessary to resort to preprocessing the image. Sometimes the edges of the object get diffused. Moreover, with CDF (2,4) and CDF (2,6) the edges get doubled (Fig.4, Fig.5).

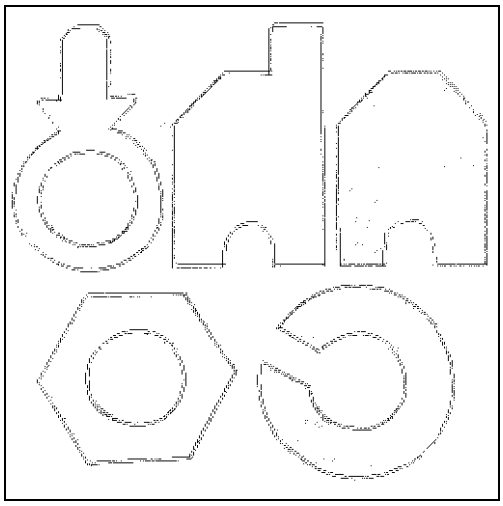

Fig.4 Edges detected by CDF (2,2).
Prior to detection, filtering via averaging
was used, with increased weight of the centre point.

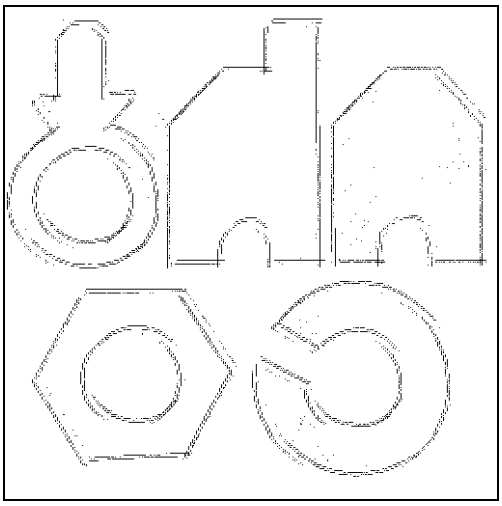

Fig.5 Edges detected by.
Prior to detection, filtering via averaging
was used, with increased weight of the centre point.

2.1.3 Edge detectors utilizing the D4 wavelet
These detectors exhibit the least noise resistance from all the detectors given so far. Three out of the four tested bitmaps with standards can be detected with a single threshold value for (LP). The edges are with slight doubling, which can still be eliminated in subsequent processing. Wavelets with a higher momentum (D6, …) were not implemented because of foreseeable greater doubling of edges.

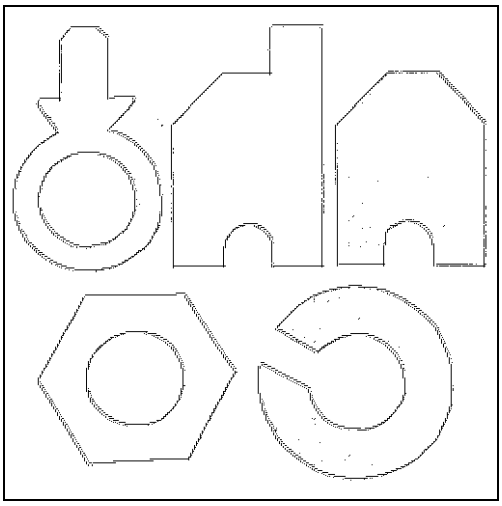

Fig.6 Edges detected by D4. No filtering prior to detection.

All the above detectors perform only one phase of the wavelet transform. In any further phase the accuracy of edge localization in the image gets reduced. This is due to halving the number of samples with each phase.

2.2 Joining the Edges

The output of an edge detector is a binary image. Green edges are depicted on a black ground. But not all the depicted edges must correspond to actual edges in the image. A spot can be denoted as an edge and have in fact nothing to do with any edge in the image. Or, vice versa, a spot that should be denoted as an edge is not denoted. These "false" detected edges are frequently due to the noise in the input image. For subsequent detection of individual objects in the image it is necessary to obtain from this image the closed boundaries of objects.

2.3 Searching for the Boundary

Boundaries are changed to the vector form by following the boundaries. In this form, every edge is defined by a list of points that belong to this boundary. In the program the MFC library is used, which offers the CList class, which is a concatenated list. Each item of the list contains the co-ordinates of one pixel. All boundaries are stored successively in an array (CArray class). Each item of the array is thus an object of class CList.

2.4 Description of algorithm

1. The image is scanned line by line, starting in the upper left corner. When a green pixel is found, an object derived from class CList is formed and the co-ordinates of the point found are entered into the subject. The colour of this point is then changed to red. The concatenated list holds a single item at this moment. This item is the first, and at the same time the last, item of the list. Searching for the boundary comes next.
2. Searching for the boundary is based on searching the 8-environment of a point of the last element of the list.

- If a green pixel has been found, its co-ordinates are added at the end of the list. The pixel is painted red and the algorithm returns to the beginning of step 2.
- If no green pixel has been found, the concatenated list is stored in the array of edges and the algorithm returns to step 1.

The algorithm described is greatly simplified. After every addition of a point to the list, it remembers the direction of the last shifting. When searching the environment, preference is given to the pixel that is the closest to the direction stored. The green pixel neighbouring on a red pixel (which already

belongs to a boundary) is not regarded as a new boundary. This provision is important in particular for edges several pixels wide.

After this phase, the boundaries are stored in vector form. Now we pass on the joining boundaries into concatenations.

2.5 Joining Edges into Concatenations

Prior to the joining of edges itself, edges whose length is one element are deleted. Frequently, these edges are caused by noise.

When calling the edge joining function the user sets several parameters:

- The minimum edge length. Edges shorter than this value (in pixels) will be deleted.
- The distance (in pixels) up to which the edges will be joined.
- Setting that closed edges alone will be preserved.

Edges are now joined using the following algorithm:

1. We set the value of δ at 2. This variable gives the distance up to which edges will be joined.
2. We test the mutual distance of all the beginnings and ends of all the stored edges. If for any two edges this distance is less than the value of δ, we supply the missing pixels and join them into one edge. The new edge will thus contain the points of the two edges and, on top of that, the points supplied. If the edge comes to be closed, it will no longer be joined with other edges.
3. Item 2. is repeated as long as there are edges in the array whose mutual distance is less than δ.
4. When there are only closed edges in the array, the algorithm comes to an end.
5. If the value of δ is less than the value set by the user, we increase the value of δ and return to step 2.

Finally we delete the edges that are shorter than the value set by the user. If the user has opted for preserving only the closed edges, we delete the non-closed edges.

3 Classification of Objects

The classification of objects is based on a comparison of momentum characteristics. The shape of every object is thus expressed by several (in our case seven) momenta. To calculate the momenta it is necessary to know the position of all pixels within a closed area (object). The task leads to filling the closed areas. This is why closed boundaries are required. An advantage of expressing objects by means of momenta lies in the independence from object rotation, its size or shifting.

3.1 Calculation of Object Momentum

General momentum of the order of p+q is defined as:

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^q f(x,y)dxdy \qquad (1)$$

where x, y are the co-ordinates of the points of the area.

For the digital image we can write:

$$m_{pq} = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} i^p j^q f(i,j) \qquad (2)$$

where i, j are the co-ordinates of the points of the area. This momentum is dependent on the size, shifting and rotation. A momentum that is independent of position and rotation can be obtained as:

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-x_t)^p (y-y_t)^q f(x,y) \qquad (3)$$

where x_1, y_1 are the co-ordinates of the centre of gravity:

$$x_t = \frac{m_{10}}{m_{00}}\ ,\ y_t = \frac{m_{01}}{m_{00}} \qquad (4)$$

where m_{00} in the case of binary image expresses its size. The independence of momenta from the change in scale can be secured by normalized central momenta:

$$\theta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\gamma}}\ ,\quad \gamma = TRUNC\left(\frac{p+q}{2}+1\right) \qquad (5)$$

where TRUNC represents the whole part x.

3.2 Comparison of Objects

First we create a file that will contain the momenta of all the known objects. To store these model momenta it is convenient to choose the best possible preprocessing and detection method in order that the stored momentum characteristics best correspond to actual momenta.

Object detection is based on comparing the calculated momenta with model momenta from the file. This comparison takes the form of seeking the least quadratic deviation:

$$\delta = \sum_{i=1}^{7} (\varphi_i - \varphi_{\bar{i}})^2 \qquad (6)$$

where φ_i… is one of the seven momenta of the object

... is one of the seven momenta of an object in the file

4 Conclusion

The paper describes the application of the wavelet transform to edge detection in images. The principles and algorithms given above have been used in an application that was developed at Brno University of Technology. The application has been programmed in the Microsoft Visual C++ 6.0 environment. In the development, the Win32 API and MFC (Microsoft Foundation Class) libraries were made use of. In the application, any of the seven implemented wavelet types can be used for edge detection.

In the application for the identification of a 2D technological scene the CDF 1,1; CDF 1,3; CDF 1,5; CDF 2,2; CDF 2,4; CDF 2,6 and D4 wavelets were programmed. The "Lifting Scheme" algorithm was used for the wavelet transform. The edge detectors that use CDF 1,X have a good edge resolution and low sensitivity to noise in the image. For the given technological scene these wavelets can be used even without preprocessing (filtering), with the exception of the CDF 1,5 filter, which produces double edges (see Fig.2, Fig.3), which is very inconvenient in subsequent operations. The detectors that use CDF 2,X exhibit greater sensitivity to noise in the image. For practical application it is thus in most cases necessary to preprocess the image. Using these detectors, the boundaries of objects get often diffused. Moreover, in the case of CDF 2,4 and CDF 2,6 the edges get doubled (see Fig.4, Fig.5). Edge detectors making use of the D4 wavelet exhibit the least sensitivity to noise. They resolve most of the objects of the given technological scene with a single threshold value for (HP). Subtle doubling of the boundary can be witnessed (see Fig.6), which can still be eliminated by subsequent processing. In the application, edge detection using the gradient can also be used. Tests have shown that the detection using the wavelet transform will surpass the classical gradient methods.

Acknowledgement

This research was supported by the grants: No 102/03/0434 Limits for broad-band signal transmission on the twisted pairs and other system co-existence The Grant Agency of the Czech Republic (GACR)
No CZ 400011(CEZ 262200011) Research of communication systems and technologies (Research design)
No IS 432 124 (2124/2003/F1) Modernizing and innovation of telecommunication services education (grant of the Czech Ministry of Education, Youth and Sports)

References:

[1] Pratt, W.K. Digital Image Processing. John Wiley & Sons, 1978.
[2] Pavlidis, T. Algorithms for Graphics and Image Processing . Bell Laboratories Computer Science Press, 1982.
[3] Huang, T.S. Picture Processing and Digital Filtering. In: Topics in Applied Physics vol.6. Springer Verlag, 1976.
[4] Uytterhoeven G., Wulpen F., Jansen M., Roos D., Bultheel A.: WAILI-Wavelets with Integer Lifting. Internet sources, 2000.
[5] Uytterhoeven G., Roos D., Bultheel A.: Wavelet Transforms Using the Lifting Scheme. . Internet sources, 2001.
[6] Daubechies I., Sweldens W.: Factoring Wavelet Transforms IntoLifting Steps. . Internet sources, 2001.
[7] Calderbank A., R., Daubechies I., Sweldens W.: Wavelet Transforms that Map Integers to Integers. Internet sources, 2000.
[8] Gaps A.: An Introduction to Wavelets. . Internet sources, 2000.
[9] Ziou D., Tabbone S.: Edge Detection Techniques. . Internet sources, 2001.