11. Edge detection

11.1. Introduction

This laboratory presents the edge detection problem in digital images. Edge points are found where the image intensity encounters a steep variation along a specific direction '*x*' (Fig. 11.1). This intensity variation can be detected and quantified by finding the local maxima of the first order derivative of the image intensity (the gradient: $\nabla f=f'$) or by finding the zero crossings of the second order derivative of the image intensity (the laplacian: $\nabla^2 f=f''$).

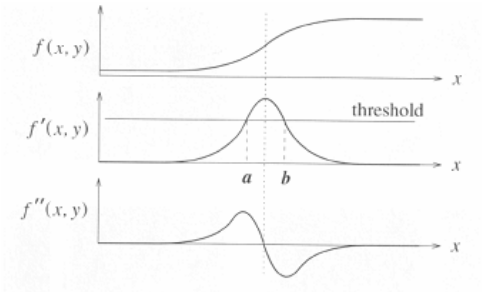

Fig. 11.1 Detection methods of the edge points (points were the image intensity suffers a steep variation).

Further on only the gradient based methods will be approached.

11.2. Computing the image gradient

The gradient in an image point is a vector heading the direction of the intensity variation around this point (Fig. 11.2). Its module is proportional with the speed of this variation (11.1). If the edge points are part of a contour (as in Fig. 11.2) the gradient will be perpendicular on the tangent to the contour at that point.

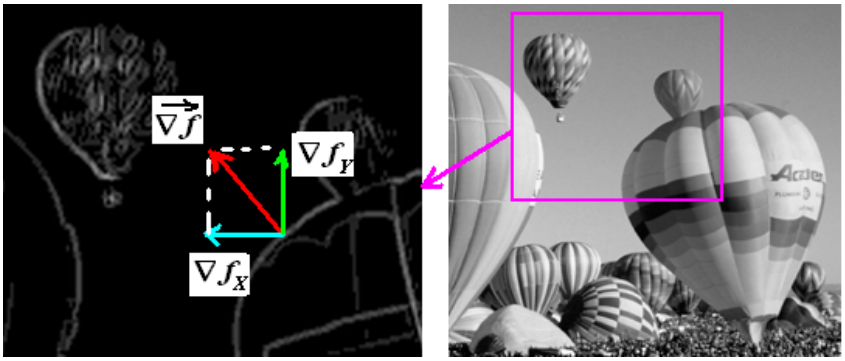

Fig. 11.2 Left: illustration of the image gradient (in an edge point) on the image of the gradient module.

The gradient of a two variables continuous function f is defined as:

$$\nabla f(x,y)=\begin{bmatrix}\nabla f_x\\ \nabla f_y\end{bmatrix}=\begin{bmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\end{bmatrix}=\begin{bmatrix}\lim\limits_{\Delta x\to 0}\frac{f(x+\Delta x,y)-f(x,y)}{\Delta x}\\ \lim\limits_{\Delta y\to 0}\frac{f(x,y+\Delta y)-f(x,y)}{\Delta y}\end{bmatrix} \tag{11.1}$$

For digital images, the gradient can be approximated by making Δx and Δy equal to 1 in (11.1):

$$\nabla f(x,y) = \begin{bmatrix} \nabla f_x \\ \nabla f_y \end{bmatrix} = \begin{bmatrix} f[x+1,y] - f[x,y] \\ f[x,y+1] - f[x,y] \end{bmatrix} \tag{11.2}$$

Other approximations of the two components of the gradient can be computed through the convolution of the image with the following kernels:

Prewitt:

$$\begin{aligned} \nabla f_x &= f(x,y) * \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \\ \nabla f_y &= f(x,y) * \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \end{aligned} \tag{11.3}$$

Sobel:

$$\begin{aligned} \nabla f_x &= f(x,y) * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \\ \nabla f_y &= f(x,y) * \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \end{aligned} \tag{11.4}$$

Roberts (cross):

$$\begin{aligned} \nabla f_x &= f(x,y) * \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \\ \nabla f_y &= f(x,y) * \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \end{aligned} \tag{11.5}$$

As a vector, the gradient can be quantified by a magnitude (11.6) and a direction (11.7):

$$\left|\nabla f(x,y)\right| = \sqrt{\left(\nabla f_x(x,y)\right)^2 + \left(\nabla f_y(x,y)\right)^2} \tag{11.6}$$

$$\theta(x,y) = arctg\left(\frac{\nabla f_y(x,y)}{\nabla f_x(x,y)}\right) \tag{11.7}$$

11.3. The Canny edge detection method

The edge detection method proposed by Canny is based on the image gradient computation but in addition tries to:

- maximize the signal-to-noise ratio for a proper detection;
- find a good localization of the edge points;
- minimize the number of positive responses around a single edge (suppression of the gradient module non-maxims)

The steps of the Canny edge detection method are given bellow:

1. Noise filtering through a Gaussian kernel
2. Computing the gradient's module and direction
3. Non-maxima suppression of the gradient's module
4. Edge linking through adaptive hysteresis thresholding.

11.3.1. Noise filtering through a Gaussian kernel

The noise in the image is high frequency information which overlaps the original image signal. This introduces false edge points. The noise intrinsic to the image acquisition process can be modeled by a Gaussian distribution and can be suppressed by a Gaussian filter (see laboratory 10).

11.3.2. Computing the gradient's magnitude and direction

Computing the gradient's module and direction requires the allocation of two temporary image buffers (with the same size as the image) and the initialization of their elements according to equations (11.6) and (11.7) respectively, where the horizontal $\nabla f_x(x,y)$ and the vertical $\nabla f_y(x,y)$ components of the image gradient can be computed using the Prewitt operator (11.3) or the Sobel operator (11.4).

11.3.3. Non-maxima suppression of the gradient's module

Its purpose is the thinning of the edges by retaining only the edge points with the highest gradient module along the direction of the image intensity variation (along the direction of the gradient vector).

The first step consists in the quantization of the gradient directions, computed using (11.7) , in 4 regions shown in Fig. 11.3:

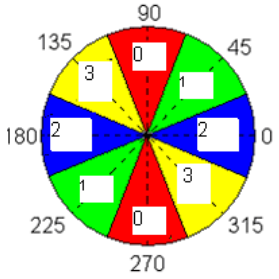

Fig. 11.3 Quantization of the gradient directions in the non-maxima suppression step

Supposing that, for example, the direction of the gradient in an image point is "1" (Fig. 11.4), the module of the gradient in point P is a local maximum if: $|\nabla P| > |\nabla I_6|$ and $|\nabla P| > |\nabla I_2|$. If these conditions are fulfilled, the point P is retained as an edge point, otherwise is rejected.

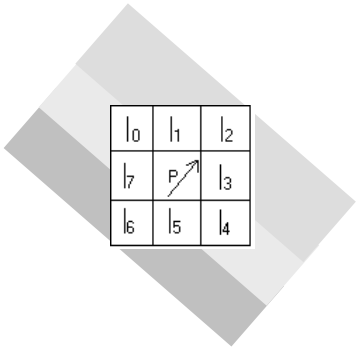

Fig. 11.4 Example for the non-maxima suppression.

11.3.4. Edge linking through adaptive hysteresis thresholding

After computing the image gradient and performing the non-maxima suppression procedure, an "image" is obtained in which the pixel values are equal with the gradient's modules in that pixel. Moreover, the thickness of the edge pixels (with non-zero module) has an ideal value of one pixel. In the following, the steps required to obtain the final edges are described:

11.3.4.1. Adaptive thresholding

Adaptive thresholding tries to extract a quite constant number edge points for a given image size. In this way, lighting and contrast variations are compensated (fixed threshold would extract either too much or too few edge points).

The parameter which is given to the threshold detection procedure is the ratio between the number of edge points and the number of points with non-zero gradient module:

$$NoEdgePixels = p * (NoPixels - ZeroGradientModulePixels) \quad (11.8)$$

Parameter p has usually values between 0.01 and 0.1.

The algorithm is the following:

1. The histogram of the gradient's magnitude image (values obtained after non-maxima suppression) is computed. These values will be scaled to fit within [0..255] range (by division with $4\sqrt{2}$ if the gradient was computed using the Sobel operator). The result is a histogram Hist[0..255]:

$$Hist[i] = No\ of\ pixels\ having\ the\ scaled\ gradient\ magnitude\ value\ i \quad (11.9)$$

2. The number of pixels with non-zero values which would not be edge points is computed:

$$NoNonEdge = (1-p) * (Height*Width - Hist[0]) \quad (11.10)$$

3. Starting with position 1 the values of the histogram are summed. When the sum exceeds the value *NoNonEdge*, then the index *i* reached in the counting process is the searched threshold. This technique, intuitively, will find the gradient magnitude value (*AdaptiveThresholding*) bellow which *NoNonEdge* pixels are found.

Pay attention to the pixels located at the image margins (where the image gradient was not computed)! Their values should be zero or should not be taken into account, because they can modify the value of the threshold.

11.3.4.2. Edge extension through hysteresis

Adaptive thresholding does not guarantees the completeness of the edges (shadowed parts of the objects or presence of noise can affect the edge detection process). The result will be an image with many fragmented edges.

Therefore an edge extension technique is needed. The edges obtained by adaptive thresholding are considered STRONG EDGES and we try to extend them with weaker edge points, which have not passed the thresholding with the initial value, but could be detected with a lower threshold.

Formally, two thresholds are defined:

$$Threshold\_high = AdaptiveThresholding \quad (11.11)$$

$$Threshold\_low = k * Threshold\_high \quad (11.12)$$

Where k<1 (for example, $k = 0.4$).

The image of the gradient module is scanned pixel by pixel. In the destination image the pixels with the gradient magnitude higher then *Threshold_high* are labeled as STRONG_EDGES (e.g. with the value 255). The pixels with the gradient magnitude between *Threshold_low* and *Threshold_high* are labeled as WEAK_EDGES (e.g. with the value 128).

The pixels with the gradient magnitude bellow *Threshold_low* are considered NON-EDGES and are rejected. The inverted result (negative) of this labeling is shown in Fig. 11.5-left:

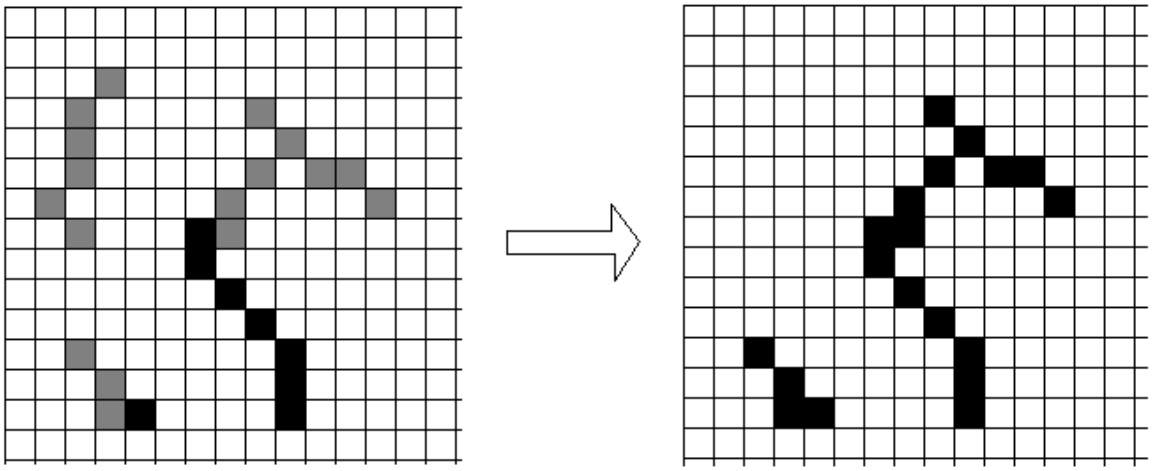

Fig. 11.5 Left: the image of the labeled strong and weak edges; Right: the result of the extension of the strong edges with connected WEAK edges.

Next step consists in the extension of the STRONG_EDGE points with neighboring WEAK_EDGE points, if they are parts of a connected component (see laboratory and lecture related to "Labeling") – as in Fig. 11.5. If a STRONG_EDGE point has WEAK_EDGE neighbor, the WEAK_EDGE neighbor is labeled as a STRONG_EDGE point. This STRONG_EDGE becomes a new source of edge extension. The process is repeated until the STRONG_EDGE points cannot be extended further by joining them with WEAK_EDGE points.

An efficient implementation of this step uses a queue to perform a breadth first search through WEAK_EDGE points connected to STRONG_EDGE points and mark them as STRONG_EDGE points. The algorithm would look like this:

1. Scan the image, top left to bottom right, pick the first STRONG_EDGE point encountered and push its coordinates in the queue.
2. While (queue is not empty)
    a) Extracts the first point from the queue
    b) Find all the WEAK_EDGE neighbors of the current point
    c) Label in the image all these neighbors as STRONG_EDGE points
    d) Push the image coordinates of these neighbors into the queue
    e) Continue to the next STRONG_EDGE point
3. Go to step 1 considering the next STRONG_EDGE point.
4. Eliminate the remaining WEAK_EDGE points from the image by turning them to NON_EDGE (0)

Final consideration: regarding the definition of the neighborhood used in the above algorithm, the common 4-type or 8-type neighborhood can be used, or a tolerance of 1 to 2 pixels can be considered. The reason is that, due to noise, the edges may be interrupted by small gaps.

11.4. Practical work

The practical activities related to this laboratory will be split in two:

11.4.1. Laboratory 1(first WEEK)

1. The horizontal ∇f_x and vertical ∇f_y components of the gradient through convolution with the kernels given in (11.3) ... (11.5) will be computed and the results will be shown in destination windows (the convolution operation was already implemented in laboratory 9).
2. The gradient magnitude (11.6) and direction (11.7) will be computed using the three operators (Sobel, Prewitt and Roberts) and the results of the gradient magnitude will be shown in a destination window.
3. The thresholding with an arbitrary and fixed threshold of the results obtained at point 2 will be shown in a destination window.
4. The steps 1 – 3 of the Canny edge detection algorithm will be implemented (step 1 – was already implemented in laboratory 10; step 2 – is the implementation from point 2 using the Sobel filters; step 3 – implement the non-maxima suppression operation). The results obtained after step 3 will be shown in a destination window. The results will be compared with the one obtained at point 2 after the simple use of the Sobel operator.
5. **Save your work. Use the same application in the next laboratories. At the end of the image processing laboratory you should present your own application with the implemented algorithms.**

11.4.2. Laboratory 2 (second WEEK)

1. Edge linking through adaptive hysteresis thresholding algorithm (step 4 of the Canny method) will be implemented. The intermediate results of the STRONG_EDGE and WEAK_EDGE points (after the hysteresis thresholding and before the edge extension step) and the final results (with the final edges) will be shown destination windows. The implementation will be experimented for different values of the parameters *p, k* and neighborhood types.
2. The final results of the implemented Canny edge detection method will be tested/experimented on different image types.
3. **Save your work. Use the same application in the next laboratories. At the end of the image processing laboratory you should present your own application with the implemented algorithms.**

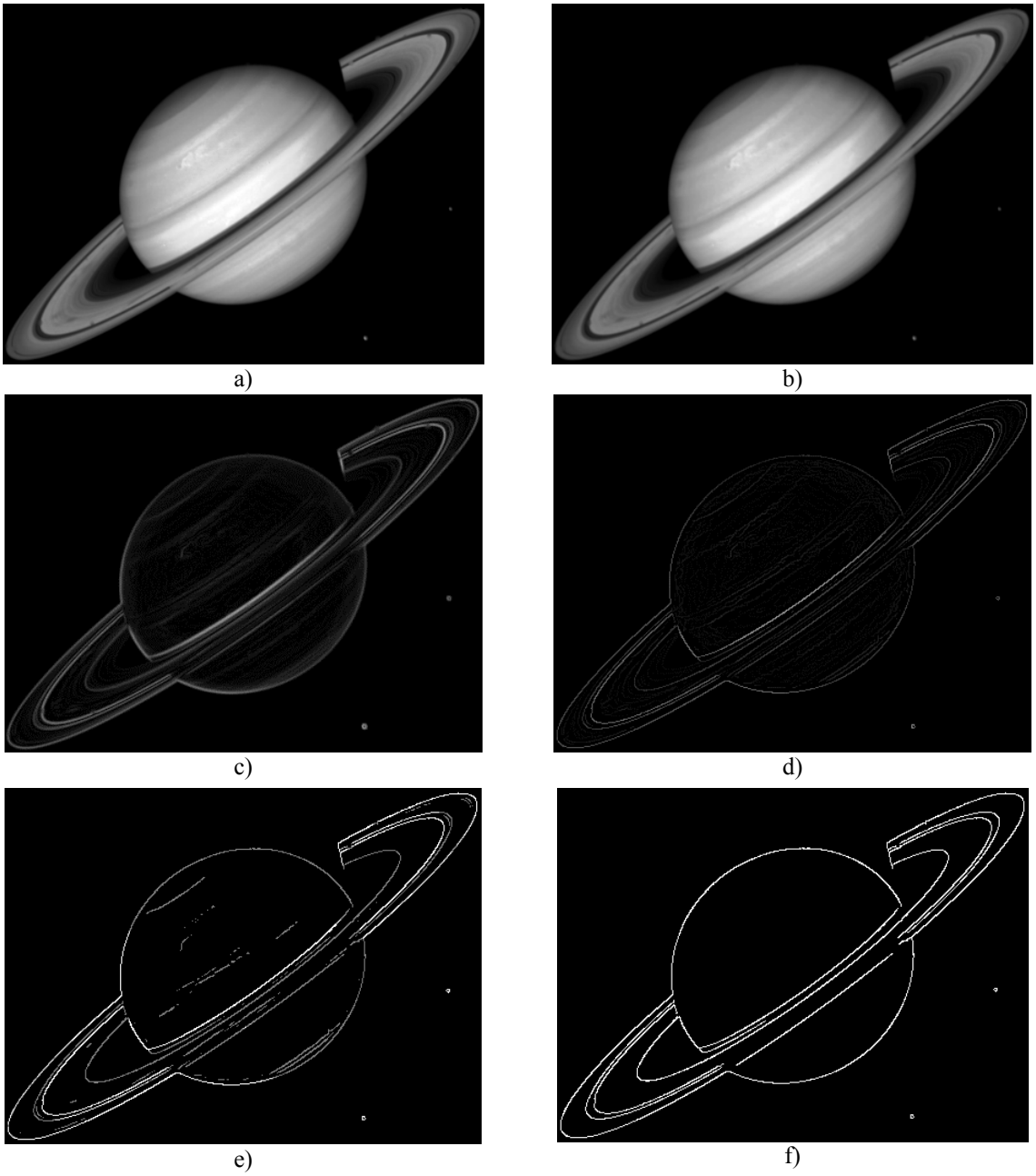

Fig. 11.6 Canny edge detection - sample results: a) initial image; b) after Gaussian filtering ($\sigma = 0.5$); c) normalized gradient magnitude (using Sobel operators); d) after non-maxima suppression; e) after adaptive thresholding (p = 0.1); f) final edges after edge extension and weak-edge removal

References

[1] E.Trucco, A.Verri, *Introductory Techniques for 3-D Computer Vision,* Prentice Hall, 2001
[2] R.C.Gonzales, R.E.Woods, *Digital Image Processing, 2-nd Edition,* Prentice Hall, 2002