Detection and localization of a 2-D contour with GHT

E. Hyrol and P. Stiasny

28.11.2016

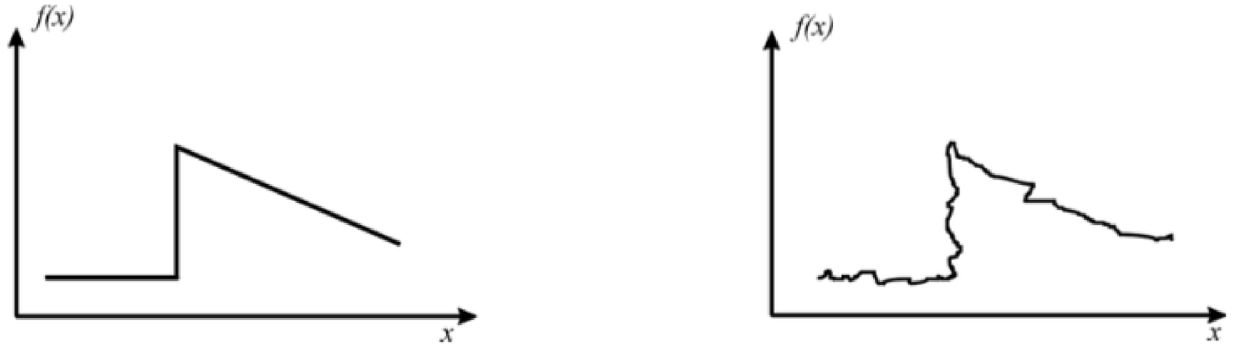
Abstract

1 Introduction

2 Theory

2.1 Image edge detection

A pre-processing step in edge detection: a smoothing operation in order to remove noise (spiky-like variations) from the image.



(a) Desired edge. (b) Real edge.

Figure 1: Example edges.

Basic types of image edge detectors:

- discrete image function gradients
- convolution kernels
- using parametric edge models
- mixed approaches

2.2 Sobel Edge Detection

The Sobel operator is used in image processing, particularly within edge detection algorithms. It falls into a group of convolution-based edge detectors. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. The Sobel operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation that it produces is relatively crude, in particular for high frequency variations in the image.

$$\Delta x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad \Delta y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Characteristics: different input-output image, e.g. maximum strength: 2040 for 8-bit input image, simple implementation, good results.

2.3 Scharr Edge detection

Scharr operator is similar to the Sober operator having the following kernel:

$$G_x = \begin{bmatrix} -3 & 0 & 3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{bmatrix}, \quad G_y = \begin{bmatrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{bmatrix}$$

Comparing with Sobel operators we see that they look similar, but Sobel weight matrix is isotropic and Scharr weight matrix is anisotropic.

Scharr operators result from an optimization minimizing weighted mean squared angular error in Fourier domain. This optimization is done under the condition that resulting filters are numerically consistent. Therefore they really are derivative kernels rather than merely keeping symmetry constraints.

2.4 Edge thinning

Threshold-based edge elimination. This simple edge thinning method is an edge elimination operator with a minimum threshold parameter θ. The threshold is either fixed or set adaptively (e.g. $\theta = \gamma S_{max}$, where $\gamma \in (0, 1)$).

$$s_{thin}(P) = \begin{cases} s(P), & \text{if } s(P) > \theta \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

No-maximum edge elimination. It depends on a check in the local neighborhood of given pixel P:

$$IF(s(P) \geq s(N_L)OR|r(P) - r(N_L)| \geq T) \; AND \; (s(P) \geq s(N_R)OR|r(P) - r(N_R)| \geq T) \tag{2}$$

$$THENs(P)_{THIN} = s(P) : ELSEs(p)_{THIN} = 0; \tag{3}$$

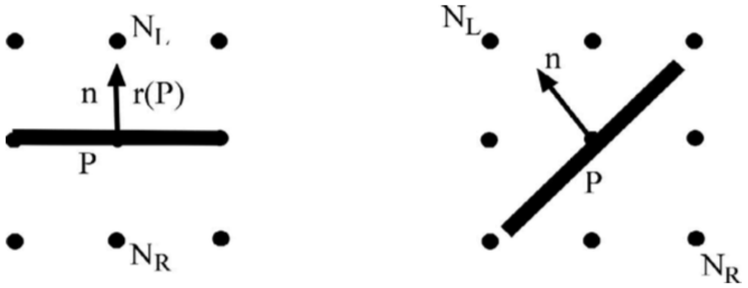


Figure 2: No-maximum edge elimination.

Edge modification.A local neighborhood-based modification of edge at pixel P:

- if P is the strongest edge element in the set: P, NL, NR , then:

$$s^{'}(P) = s(P) + \alpha(s(N_L) + s(N_R))$$

- if P is the weakest edge element in the above set then:

$$s^{'}(P) = s(P) - 2\alpha s(P)$$

- if one neighbour of P (denoted by P^+) is a stronger edge and another neighbour of P (denoted by P^-) is a weaker edge element then:

$$s^{'}(P) = s(P) - \alpha s(P^+) + \alpha s(P^-).$$

Several iterations over the whole image may be necessary.

Edge elimination with hysteresis threshold. This edge thinning method works with two edge strength thresholds: **the upper** θ_H and **the lower** θ_L. In the first run these edge pixels are individually marked as "good" that have higher strengths than the upper threshold. In the next run these "good" pixels are tried to be extended along a potential contour line in both directions (positive and negative line direction). For a single extension, the neighbour pixel needs to have a higher strength than the lower threshold.

Remark: Now the neighbours are not competing with each other and they are searched along the expected contour line (not across it).

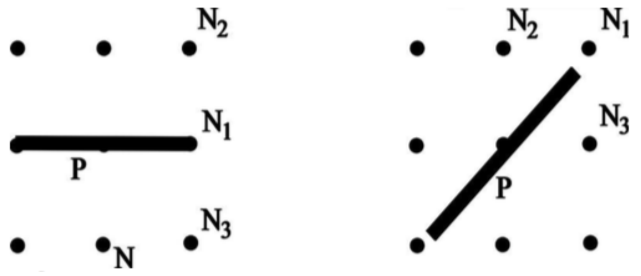
2.5 Edge chain following

Principle: searching for extension of current edge pixel $P = P_{-cur}$ by its successor edge $N = c(P_{cur})$. Two neighbour edge elements can be linked if the edge magnitude (strength) and direction differences are below certain thresholds and their magnitudes are relatively large:

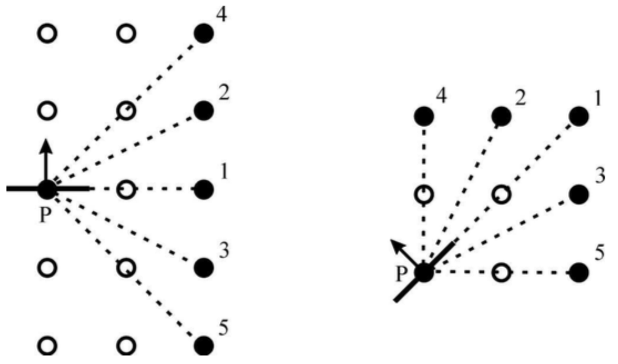
$$|s(P) - s(N)| \leq T_1 \tag{4}$$
$$|r(P) - r(N)| \; mod \; 2\pi \leq T_2 \tag{5}$$
$$|s(P)| > T, \; |s(N)| > T \tag{6}$$

Denote the 3 nearest pixel along the direction $r(P)$ as: N_1, N_2, N_3. The successor of P_{-cur}: an edge element Ni whose strength and orientation are most similar to P_{-cur}.

Successor candidates. Three candidates for a successor of pixel P:



Closing a gap for edge pixel P in directions (left) 0° and (right) 45°:



The hysteresis threshold method. Contrast (edge strength) may be different in different points of the contour. Careful thresholding of $M(x, y)$ is needed to remove weak edges while preserving the connectivity of the contours.

Hysteresis thresholding receives the output of the non-maxima suppression, $M_{NMS}(x, y)$. The algorithm uses 2 thresholds, T_{high} and T_{low}:

- A pixel (x, y) is called strong if $M_{NMS}(x, y) > T_{high}$.
- A pixel (x, y) is called weak if $M_{NMS}(x, y) \leq T_{low}$.

- All other pixels are called candidate pixels.

The algorithm has the following steps:

1. In each position of (x, y), discard the pixel (x, y) if it is weak; output the pixel if it is strong.

2. If the pixel is a candidate, follow the chain of connected local maxima in both directions along the edge, as long as $M_{NMS}(x, y) > T_{low}$.

3. If the starting candidate pixel (x, y) is connected to a strong pixel, output this candidate pixel; otherwise, do not output the candidate pixel.

An example can be observed on Figure 3.

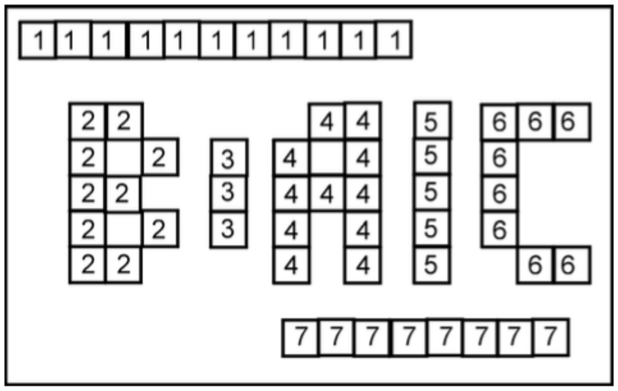


Figure 3: Edge chain labels

2.6 Contour detection by GHT

Parameters of the Hough space: $C = (x_C; y_C)$: location of the center mass, s - scale, α - contour orientation angle.

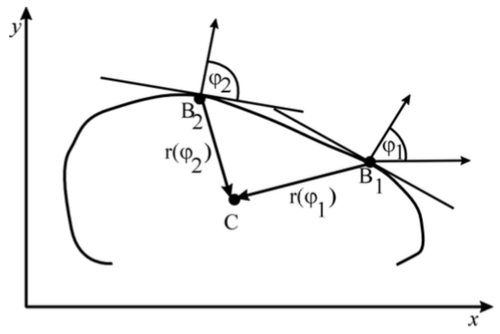
Model learning (design). For every pair of allowed discrete values s_d, α_g of scale and orientation, create a table $R(s_d, \alpha_g) = [r(\varphi), \varphi]$, where for each edge x_B with edge direction φ the pair: $[r(\varphi) = x_B - C, \varphi]$ is stored.

3 Project implementation

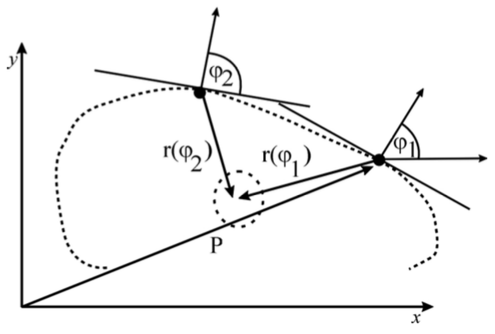
For implementation we chose the Python programming language with appropriate OpenCV bindings. OpenCV-Python provides good integration with the NumPy library for efficient matrix operations which we will use when necessary. We plan to first prepare a prototype using existing solutions found in OpenCV, then incrementally implement our own components.

The general algorithm we will implement is as follows:

1. Apply Gaussian filter to smooth the image in order to remove the noise (image).



(a) Model contour.



(b) Candidate contour.

Figure 4: Example of GHT contour detection.

2. Find the intensity gradients of the image using a convolution-based operator. We will use Sobel and Scharr operators. At each point in the image we will obtain the corresponding gradient vector(intensity gradients).

3. No-Maximum suppression will be applied because after applying gradient calculation, the edge extracted from the gradient value is still quite blurred. Non-maximum suppression will help to suppress all the gradient values to 0 except the local maximal, which indicates location with the sharpest change of intensity value.(filtered gradients)

4. Perform edge chain detection by the hysteresis method.(edge pixels)

5. Use the Generalized Hough Transform for contour localization. We will first use a picture of a bottle for model learning. We will then apply GHT to find contours of said bottle.