Edge Detection

Leonhard Applis

TH Nürnberg

05.11.2018

Table of Contents

What makes an edge?

Figure: Felix

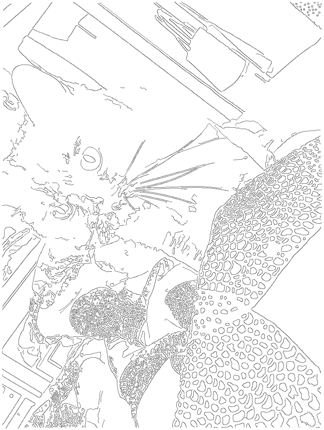


Figure: Felix's Edges

Problem I: Contrast

Figure: High Contrast Felix

Figure: Low Contrast Felix

Problem II: Smoothness

Figure: Smooth Felix

Problem III: Noise

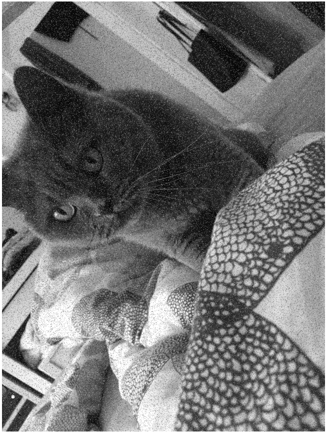


Figure: Salted Felix

Definition

In Image Processing, an edge can be defined as a set of contiguous pixel positions where an abrupt change of intensity, gray- or color-values occur. Edges represent boundaries between objects and background. Sometimes, the edge-pixel-sequence may be broken due to insufficient intensity difference.(Malay K. Pakhira)

Table of Contents

One dimensional approach

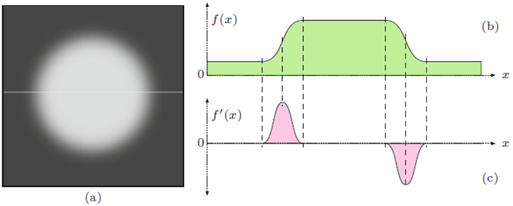


Figure: One dimensional image function and derivation

Only applyable with known, steady functions

Approximating discrete derivation

Problem: the image function is discrete, therefore we need to approximate the derivation

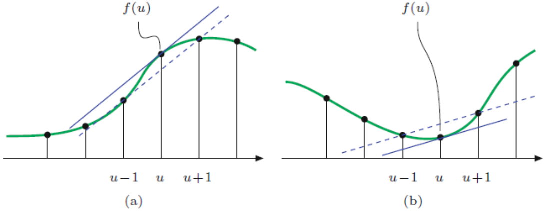


Figure: Approximation of the derivation for discrete imagefunctions

$$\frac{df}{dx}(u) \approx \frac{f(u+1) - f(u-1)}{(u+1) - (u-1)} = \frac{f(u+1) - f(u-1)}{2}$$

Two dimensional approach

If working with full images,we got two dimensions and therefore two partial derivations:

$$I_x = \frac{\partial I}{\partial x}(u,v), I_y = \frac{\partial I}{\partial y}(u,v)$$

the **gradient** at the point *(u,v)* is

$$\nabla I(u,v) = \begin{pmatrix} I_x(u,v) \\ I_y(u,v) \end{pmatrix}$$

And the **magnitude** is

$$|\nabla I| = \sqrt{I_x^2 + I_y^2}$$

Example

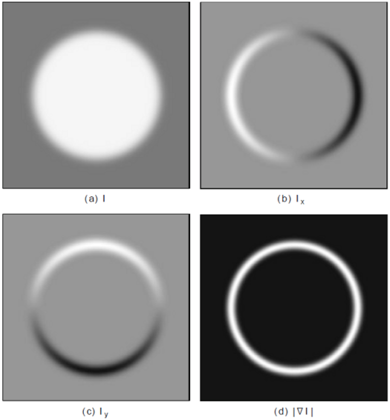
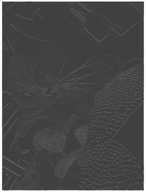


Figure: Visualisation of simple gradient-based edgedetection

Example with Felix

(a) I

(b) I_x



(c) I_y

(d) $|\nabla I|$

Figure: Simple Filters applied to Felix[1]

[1] All images have a 50% increased brightness

Implementation with filters

Expressing the gradient as a *linear filter* is simple:

$$I_x = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} \qquad I_y = \begin{bmatrix} -0.5 \\ 0 \\ 0.5 \end{bmatrix}$$

Table of Contents

Prewitt Operator

Idea: Include neighboorhood

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\nabla I^P(u,v) \approx \frac{1}{6} \cdot \begin{pmatrix} (I * H_x^P)(u,v) \\ (I * H_y^P)(u,v) \end{pmatrix}$$

Sobel

Idea: Include neighboorhood but weight center more

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\nabla I^S(u,v) \approx \frac{1}{8} \cdot \begin{pmatrix} (I * H_x^S)(u,v) \\ (I * H_y^S)(u,v) \end{pmatrix}$$

Comparison with Felix

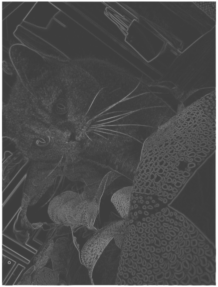


Figure: Simple edge filter

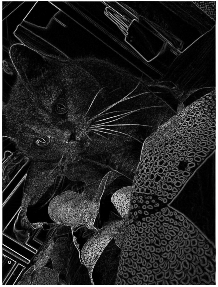


Figure: Prewitt Operator

Figure: Sobel Operator

Evaluations

general magnitude: $E(u,v) = \sqrt{I_x^2(u,v) + I_y^2(u,v)}$
holds for every Operator

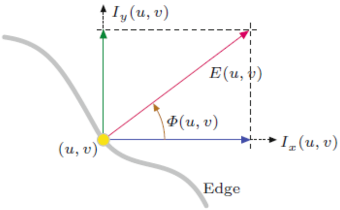


Figure: Visualisation of edge direction

$$\Phi(u,v) = \tan^{-1}\left(\frac{I_y(u,v)}{I_x(u,v)}\right) = \arctan(I_x(u,v), I_y(u,v))$$

Table of Contents

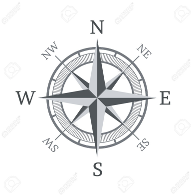
Compass Operators

Problem: The stronger a filter responses to edge-like structures, the more sensitive it is to orientation

$\Uparrow \textit{Orientation} \approx \Downarrow \textit{Edges} \Uparrow \textit{Edges} \approx \Downarrow \textit{Orientation}$

Solution: Apply 8 filters for each direction of the compass

Produce two pictures: One for the edge-strength taking the magnitude of the strongest activated filter,
one for the direction marking the strongest filter with colour



Extended Sobel Operator

$$H_0^{ES} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad H_1^{ES} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix},$$

$$H_2^{ES} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \qquad H_3^{ES} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix},$$

$$H_4^{ES} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \qquad H_5^{ES} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix},$$

$$H_6^{ES} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \qquad H_7^{ES} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}.$$

Figure: Extended Sobel Operator

Note: Its only required to apply 4 filters, as they are symetric

Extended Sobel Example

Figure: $\Phi^{ES}(Felix)$

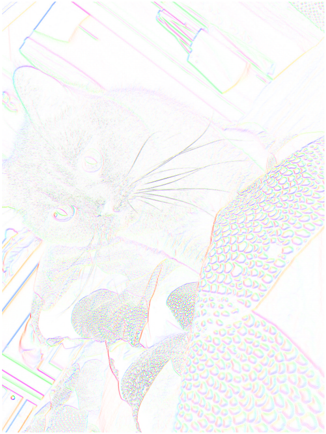


Figure: Colour encoding

Kirsch Operator

$$
H_0^{\mathrm{K}} = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix}, \quad
H_4^{\mathrm{K}} = \begin{bmatrix} 3 & 3 & -5 \\ 3 & 0 & -5 \\ 3 & 3 & -5 \end{bmatrix},
$$

$$
H_1^{\mathrm{K}} = \begin{bmatrix} -5 & -5 & 3 \\ -5 & 0 & 3 \\ 3 & 3 & 3 \end{bmatrix}, \quad
H_5^{\mathrm{K}} = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & -5 \\ 3 & -5 & -5 \end{bmatrix},
$$

$$
H_2^{\mathrm{K}} = \begin{bmatrix} -5 & -5 & -5 \\ 3 & 0 & 3 \\ 3 & 3 & 3 \end{bmatrix}, \quad
H_6^{\mathrm{K}} = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix},
$$

$$
H_3^{\mathrm{K}} = \begin{bmatrix} 3 & -5 & -5 \\ 3 & 0 & -5 \\ 3 & 3 & 3 \end{bmatrix}, \quad
H_7^{\mathrm{K}} = \begin{bmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{bmatrix}.
$$

Figure: Kirsch Operator

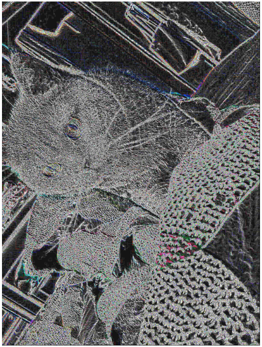


Figure: Felix with Kirsch

Canny-Edge Operator

1. Pre-Processing
2. Edge Localization
3. Edge Tracing and hysteresis threshholding

For complex applications the Canny-Edge-Algorithm is done with more than 3x3 Filters

1. Smooth Image with Gaussian-Filter
2. Apply simple filters (See 14)
3. Calculate magnitude

Also required: σ as Gauss-Radious and t_1, t_2 for thresholding edges

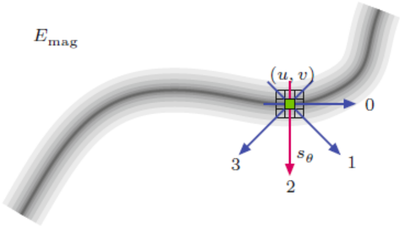


Figure: Edge-Localization

Operate over the magnitude-image:
Null every value that is not a local-maximum in any direction

1. Find points with $E \geq t_1$
2. Add Every Neighboor with $E \geq t_2$
3. Repeat 2 until image is processed

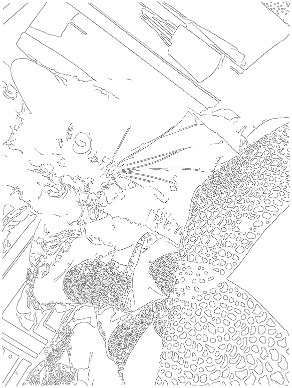


Figure: Canny-Edge Felix

Table of Contents

Laplacian-Filter
Concept

Idea: Alter *steep* imagepoints

$$\hat{f}(x) = f(x) + \omega \cdot f''(x)$$

Where ω is a weight for alternation,
$f(x)$ is the image function

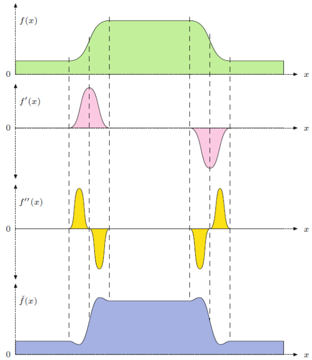


Figure: Applying the Laplacian-Filter

Laplacian-Filter
Implementation

Laplacian Operator: $(\nabla^2 f)(x,y) = \dfrac{\partial^2 f}{\partial^2 x}(x,y) + \dfrac{\partial^2 f}{\partial^2 y}(x,y)$

Expressed as filters:

$$\frac{\partial^2 f}{\partial^2 x}(x,y) \approx \begin{bmatrix} 1 & -2 & 1 \end{bmatrix}, \frac{\partial^2 f}{\partial^2 y}(x,y) \approx \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}$$

$$\Rightarrow H^L \approx \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Laplacian-Filter
Variants and Example

common variants:

$$H^L \approx \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$H^L \approx \begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Figure: Laplacian Filter applied to Felix

Unsharp Masking (USM)
Concept

1. produce a smoothed version of the image
2. subtract the smoothed version from original
   $\rightarrow$ **mask** of *sharp* points
3. add the mask with a weight α to original image

Any smoothing-filter can be used, but gaussian filters are common
Notable:

1. highly adaptable through exchangeable filters and α
2. Laplacian filters are a implementation of USM with very simple smoothing
3. vulnerable to *boost* noise, therefore often combined with a threshhold

Table of Contents

Summary

- Edges are found via gradient-based filters
- Most algorithms are simple
- Most algorithms require threshholds from user
- Best results are achieved when combined with smoothing
- Iteratively trying multiple variants can improve results
- Images with sharpened edges shouldn't be trusted 100%

Primary Sources

Digital Image Processing - An Algorithmic Introduction Using Java
2nd Edition
Wilhelm Burger, Mark J.Burge
Source for every non-felix picture.

User Guide and Documentation for HIPR
https://homepages.inf.ed.ac.uk/rbf/HIPR2/
Sometimes longer explanations and additional variants of the filters