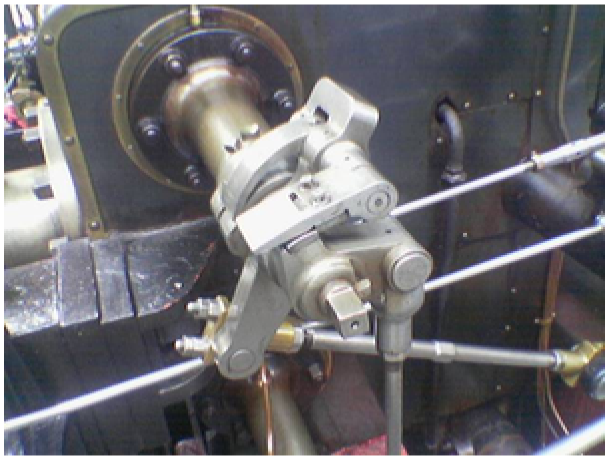
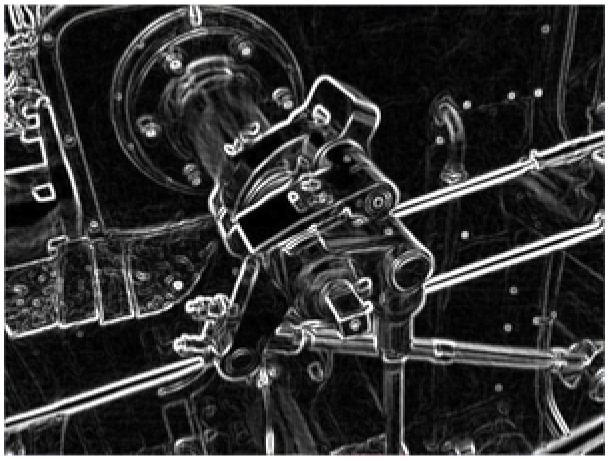
WIKIPEDIA

# Sobel operator

The **Sobel operator**, sometimes called the **Sobel–Feldman operator** or **Sobel filter**, is used in image processing and computer vision, particularly within edge detection algorithms where it creates an image emphasising edges. It is named after Irwin Sobel and Gary Feldman, colleagues at the Stanford Artificial Intelligence Laboratory (SAIL). Sobel and Feldman presented the idea of an "Isotropic 3 × 3 Image Gradient Operator" at a talk at SAIL in 1968.[1] 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–Feldman operator is either the corresponding gradient vector or the norm of this vector. The Sobel–Feldman operator is based on convolving the image with a small, separable, and integer-valued filter in the horizontal and vertical directions 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.


A color picture of an engine


The Sobel operator applied to that image

## Contents

## Formulation

The operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives – one for horizontal changes, and one for vertical. If we define $\mathbf{A}$ as the source image, and $\mathbf{G}_x$ and $\mathbf{G}_y$ are two images which at each point contain the horizontal and vertical derivative approximations respectively, the computations are as follows:[2]

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A}$$

where $*$ here denotes the 2-dimensional signal processing convolution operation.

Since the Sobel kernels can be decomposed as the products of an averaging and a differentiation kernel, they compute the gradient with smoothing. For example, $\mathbf{G}_x$ can be written as

$$\mathbf{G}_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \left( \begin{bmatrix} +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \right) \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix} * \left( \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \mathbf{A} \right)$$

The *x*-coordinate is defined here as increasing in the "right"-direction, and the *y*-coordinate is defined as increasing in the "down"-direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$\mathbf{G} = \sqrt{{\mathbf{G}_x}^2 + {\mathbf{G}_y}^2}$$

Using this information, we can also calculate the gradient's direction:

$$\mathbf{\Theta} = \operatorname{atan2}(\mathbf{G}_y, \mathbf{G}_x)$$

where, for example, $\mathbf{\Theta}$ is 0 for a vertical edge which is lighter on the right side (for $\operatorname{atan2}$ see atan2).

## More formally

Since the intensity function of a digital image is only known at discrete points, derivatives of this function cannot be defined unless we assume that there is an underlying differentiable intensity function that has been sampled at the image points. With some additional assumptions, the derivative of the continuous intensity function can be computed as a function on the sampled intensity function, i.e. the digital image. It turns out that the derivatives at any particular point are functions of the intensity values at virtually all image points. However, approximations of these derivative functions can be defined at lesser or larger degrees of accuracy.

The Sobel-Feldman operator represents a rather inaccurate approximation of the image gradient, but is still of sufficient quality to be of practical use in many applications. More precisely, it uses intensity values only in a 3×3 region around each image point to approximate the corresponding image gradient, and it uses only integer values for the coefficients which weight the image intensities to produce the gradient approximation.

## Extension to other dimensions

The Sobel–Feldman operator consists of two separable operations:[3]

- Smoothing perpendicular to the derivative direction with a triangle filter: $h(-1) = 1, h(0) = 2, h(1) = 1$
- Simple central difference in the derivative direction: $h'(-1) = 1, h'(0) = 0, h'(1) = -1$

Sobel–Feldman filters for image derivatives in different dimensions with $x, y, z, t \in \{0, -1, 1\}$ :

1D: $h'_x(x) = h'(x);$

2D: $h'_x(x, y) = h'(x)h(y)$

2D: $h'_y(x, y) = h(x)h'(y)$

3D: $h'_y(x, y, z) = h(x)h'(y)h(z)$

3D: $h'_z(x, y, z) = h(x)h(y)h'(z)$

4D: $h'_x(x, y, z, t) = h'(x)h(y)h(z)h(t)$

Thus as an example the 3D Sobel–Feldman kernel in *z*-direction:

$$h'_z(:,:,-1) = \begin{bmatrix} +1 & +2 & +1 \\ +2 & +4 & +2 \\ +1 & +2 & +1 \end{bmatrix} \quad h'_z(:,:,0) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad h'_z(:,:,1) = \begin{bmatrix} -1 & -2 & -1 \\ -2 & -4 & -2 \\ -1 & -2 & -1 \end{bmatrix}$$

## Technical details

As a consequence of its definition, the Sobel operator can be implemented by simple means in both hardware and software: only eight image points around a point are needed to compute the corresponding result and only integer arithmetic is needed to compute the gradient vector approximation. Furthermore, the two discrete filters described above are both separable:

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} [1 \quad 0 \quad -1] = \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 \\ 1 \end{bmatrix} [1 \quad -1] * [1 \quad 1]$$
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} [1 \quad 2 \quad 1] = \begin{bmatrix} 1 \\ 1 \end{bmatrix} * \begin{bmatrix} 1 \\ -1 \end{bmatrix} [1 \quad 1] * [1 \quad 1]$$

and the two derivatives $\mathbf{G}_x$ and $\mathbf{G}_y$ can therefore be computed as

$$\mathbf{G}_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * ([1 \quad 0 \quad -1] * \mathbf{A}) \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} * ([1 \quad 2 \quad 1] * \mathbf{A})$$

In certain implementations, this separable computation may be advantageous since it implies fewer arithmetic computations for each image point.

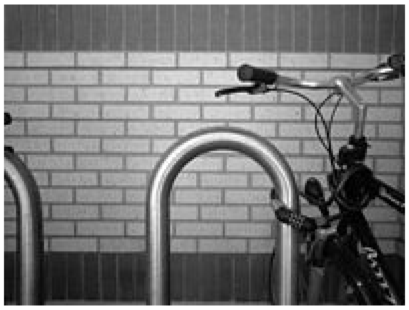
Applying convolution *K* to pixel group *P* can be represented in pseudocode as:

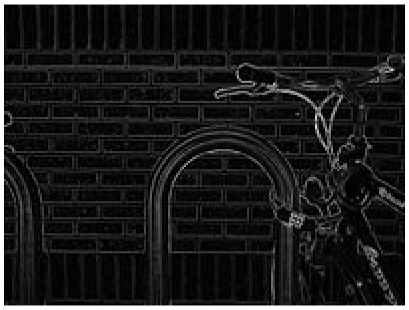
```
N(x,y) = Sum of { K(i,j).P(x-i,y-j)}, for i,j running from -1 to 1.
```

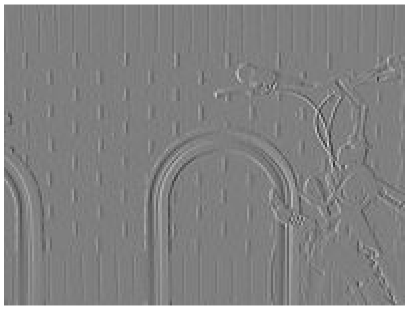
N(x,y) represents the new matrix resulted after applying the Convolution *K* to *P*, where *P* is pixel matrix.

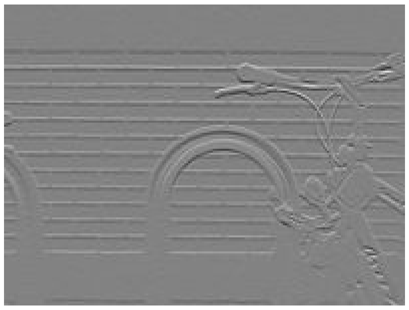
## Example

The result of the Sobel–Feldman operator is a 2-dimensional map of the gradient at each point. It can be processed and viewed as though it is itself an image, with the areas of high gradient (the likely edges) visible as white lines. The following images illustrate this, by showing the computation of the Sobel-Feldman operator on a simple image.

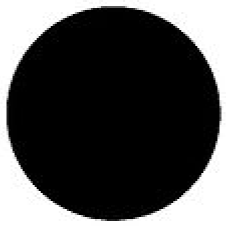

Grayscale test image of brick wall and bike rack

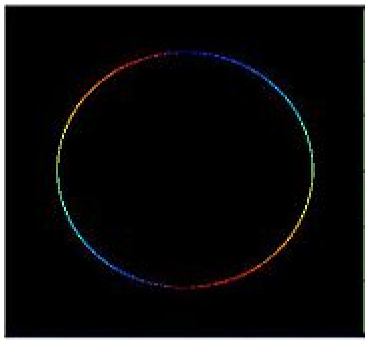

Normalized gradient magnitude from Sobel–Feldman operator


Normalized *x*-gradient from Sobel–Feldman operator


Normalized *y*-gradient from Sobel–Feldman operator

The images below illustrate the change in the direction of the gradient on a grayscale circle. When the sign of $\mathbf{G_x}$ and $\mathbf{G_y}$ are the same the gradient's angle is positive, and negative when different. In the example below the red and yellow colors on the edge of the circle indicate positive angles, and the blue and cyan colors indicate negative angles. The vertical edges on the left and right sides of the circle have an angle of 0 because there is no local change in $\mathbf{G_y}$. The horizontal edges at the top and bottom sides of the circle have angles of $-\frac{\pi}{2}$ and $\frac{\pi}{2}$ respectively because there is no local change in $\mathbf{G_x}$. The negative angle for top edge signifies the transition is from a bright to dark region, and the positive angle for the bottom edge signifies a transition from a dark to bright region. All other pixels are marked as black due to no local change in either $\mathbf{G_x}$ or $\mathbf{G_y}$, and thus the angle is not defined. Since the angle is a function of the ratio of $\mathbf{G_y}$ to $\mathbf{G_x}$ pixels with small rates of change can still have a large angle response. As a result noise can have a large angle response which is typically undesired. When using gradient angle information for image processing applications effort should be made to remove image noise to reduce this false response.


Grayscale image of a black circle with a white background.


The direction of the Sobel operator's gradient.

## Alternative operators

The Sobel–Feldman operator, while reducing artifacts associated with a pure central differences operator, does not exhibit a good rotational symmetry (about 1° of error). Scharr looked into optimizing this property by producing kernels optimized for specific given numeric precision (integer, float…) and dimensionalities (1D, 2D, 3D).[4][5] Optimized 3D filter kernels up to a size of 5 x 5 x 5 have been presented there, but the most frequently used, with an error of about 0.2° is:

$$h'_x(:,:) = \begin{bmatrix} +3 & 0 & -3 \\ +10 & 0 & -10 \\ +3 & 0 & -3 \end{bmatrix} \qquad h'_y(:,:) = \begin{bmatrix} +3 & +10 & +3 \\ 0 & 0 & 0 \\ -3 & -10 & -3 \end{bmatrix}$$

This factors similarly:

- $\begin{bmatrix} 3 & 10 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 3 \end{bmatrix}$

**Scharr operators** result from an optimization minimizing weighted mean squared angular error in the 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. The optimal 8 bit integer valued 3x3 filter stemming from Scharr's theory is

$$h'_x(:,:) = \begin{bmatrix} 47 & 0 & -47 \\ 162 & 0 & -162 \\ 47 & 0 & -47 \end{bmatrix} \qquad h'_y(:,:) = \begin{bmatrix} 47 & 162 & 47 \\ 0 & 0 & 0 \\ -47 & -162 & -47 \end{bmatrix}$$

A similar optimization strategy and resulting filters were also presented by Farid and Simoncelli.[6][7] They also investigate higher-order derivative schemes. In contrast to the work of Scharr, these filters are not enforced to be numerically consistent.

The problem of derivative filter design has been revisited e.g. by Kroon.[8]

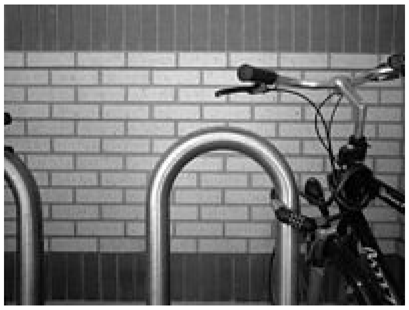
Derivative filters based on arbitrary cubic splines was presented by Hast.[9] He showed how first and second order derivatives can be computed correctly using cubic or trigonometric splines by a double filtering approach giving filters of length 7.

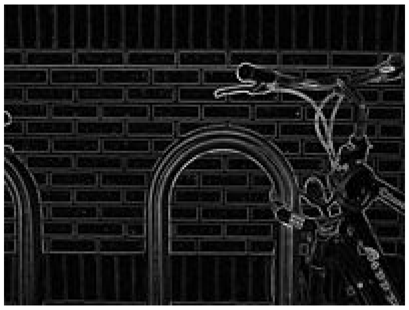
Another similar operator that was originally generated from the Sobel operator is the Kayyali operator,[10] a perfect rotational symmetry based convolution filter 3x3.

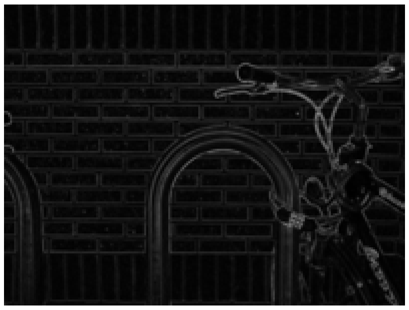
Orientation-optimal derivative kernels drastically reduce systematic estimation errors in optical flow estimation. Larger schemes with even higher accuracy and optimized filter families for extended optical flow estimation have been presented in subsequent work by Scharr.[11] Second order derivative filter sets have been investigated for transparent motion estimation.[12] It has been observed that the larger the resulting kernels are, the better they approximate derivative-of-Gaussian filters.

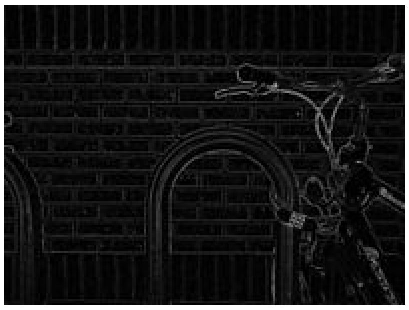
## Example comparisons

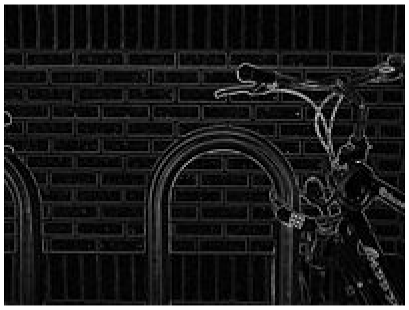
Here, four different gradient operators are used to estimate the magnitude of the gradient of the test image.


Grayscale test image of brick wall and bike rack


Gradient magnitude from Sobel–Feldman operator


Gradient magnitude from Scharr operator


Gradient magnitude from Roberts Cross operator


Gradient magnitude from Prewitt operator

## MATLAB implementation

```
clc
clear all
close all

test_img = imread('gantrycrane.png');
gray_img = rgb2gray(test_img);
sobel_img = sobel(gray_img);
```

```
figure
imshow(test_img)

figure
imshow(gray_img)

figure
imshow(sobel_img)

function output_image = sobel(A)
Gx = [-1 0 1; -2 0 2; -1 0 1]
Gy = [-1 -2 -1; 0 0 0; 1 2 1]

rows = size(A, 1)
columns = size(A, 2)
mag = zeros(size(A));

A = double(A);

for i=1:rows-2
    for j=1:columns-2
        S1 = sum(sum(Gx.*A(i:i+2, j:j+2)));
        S2 = sum(sum(Gy.*A(i:i+2, j:j+2)));

        mag(i + 1, j + 1) = sqrt(S1.^2 + S2.^2);
    end
end

threshold = 70 % varies for application [0–255]
output_image = max(mag, threshold);
output_image(output_image == round(threshold)) = 0;
end
```

The above MATLAB code will use an image which is packaged with MATLAB itself ('gantrycrane.png') to demo the Sobel filter. There will be outputs with original image, grayscale version of original image and the output from the Sobel filter.

## See also

- Digital image processing
- Feature detection (computer vision)
- Feature extraction
- Discrete Laplace operator
- Prewitt operator

## References

1. Irwin Sobel, 2014, *History and Definition of the Sobel Operator* (https://www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator)
2. Feature Detectors – Sobel Edge Detector (http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm)
3. K. Engel (2006). *Real-time volume graphics*. pp. 112–114.
4. Scharr, Hanno, 2000, Dissertation (in German), *Optimal Operators in Digital Image Processing* (http://nbn-resolving.de/urn/resolver.pl?urn=urn:nbn:de:bsz:16-opus-9622).
5. B. Jähne, H. Scharr, and S. Körkel. Principles of filter design. In Handbook of Computer Vision and Applications. Academic Press, 1999.

6. H. Farid and E. P. Simoncelli, *Optimally Rotation-Equivariant Directional Derivative Kernels* (http://www.cs.dartmouth.edu/~farid/downloads/publications/caip97.pdf), Int'l Conf Computer Analysis of Images and Patterns, pp. 207–214, Sep 1997.
7. H. Farid and E. P. Simoncelli, *Differentiation of discrete multi-dimensional signals* (http://www.cns.nyu.edu/pub/lcv/farid03-reprint.pdf), IEEE Trans Image Processing, vol.13(4), pp. 496–508, Apr 2004.
8. D. Kroon, 2009, Short Paper University Twente, *Numerical Optimization of Kernel-Based Image Derivatives* (http://www.k-zone.nl/Kroon_DerivativePaper.pdf).
9. A. Hast., "Simple filter design for first and second order derivatives by a double filtering approach" (http://www.sciencedirect.com/science/article/pii/S0167865514000282), Pattern Recognition Letters, Vol. 42, no.1 June, pp. 65–71. 2014.
10. Dim, Jules R.; Takamura, Tamio (2013-12-11). "Alternative Approach for Satellite Cloud Classification: Edge Gradient Application" (https://doi.org/10.1155%2F2013%2F584816). *Advances in Meteorology*. **2013**: 1–8. doi:10.1155/2013/584816 (https://doi.org/10.1155%2F2013%2F584816). ISSN 1687-9309 (https://www.worldcat.org/issn/1687-9309).
11. Scharr, Hanno (2007). "Optimal Filters for Extended Optical Flow". *Complex Motion*. Lecture Notes in Computer Science. Vol. 3417. Berlin, Heidelberg: Springer Berlin Heidelberg. pp. 14–29. doi:10.1007/978-3-540-69866-1_2 (https://doi.org/10.1007%2F978-3-540-69866-1_2). ISBN 978-3-540-69864-7.
12. Scharr, Hanno, *OPTIMAL SECOND ORDER DERIVATIVE FILTER FAMILIES FOR TRANSPARENT MOTION ESTIMATION* (http://www.eurasip.org/Proceedings/Eusipco/Eusipco2007/Papers/a2l-g03.pdf) 15th European Signal Processing Conference (EUSIPCO 2007), Poznan, Poland, September 3–7, 2007.

## External links

- Sobel edge detection in OpenCV (http://docs.opencv.org/doc/tutorials/imgproc/imgtrans/sobel_derivatives/sobel_derivatives.html)
- Sobel Filter (https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.sobel.html), in the SciPy Python Library
- Bibliographic citations for Irwin Sobel (http://www.informatik.uni-trier.de/~ley/db/indices/a-tree/s/Sobel:Irwin.html) in DBLP
- Sobel edge detection example using computer algorithms (http://www.mathworks.com/discovery/edge-detection.html)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Sobel_operator&oldid=1120354351"

**This page was last edited on 6 November 2022, at 15:26 (UTC).**