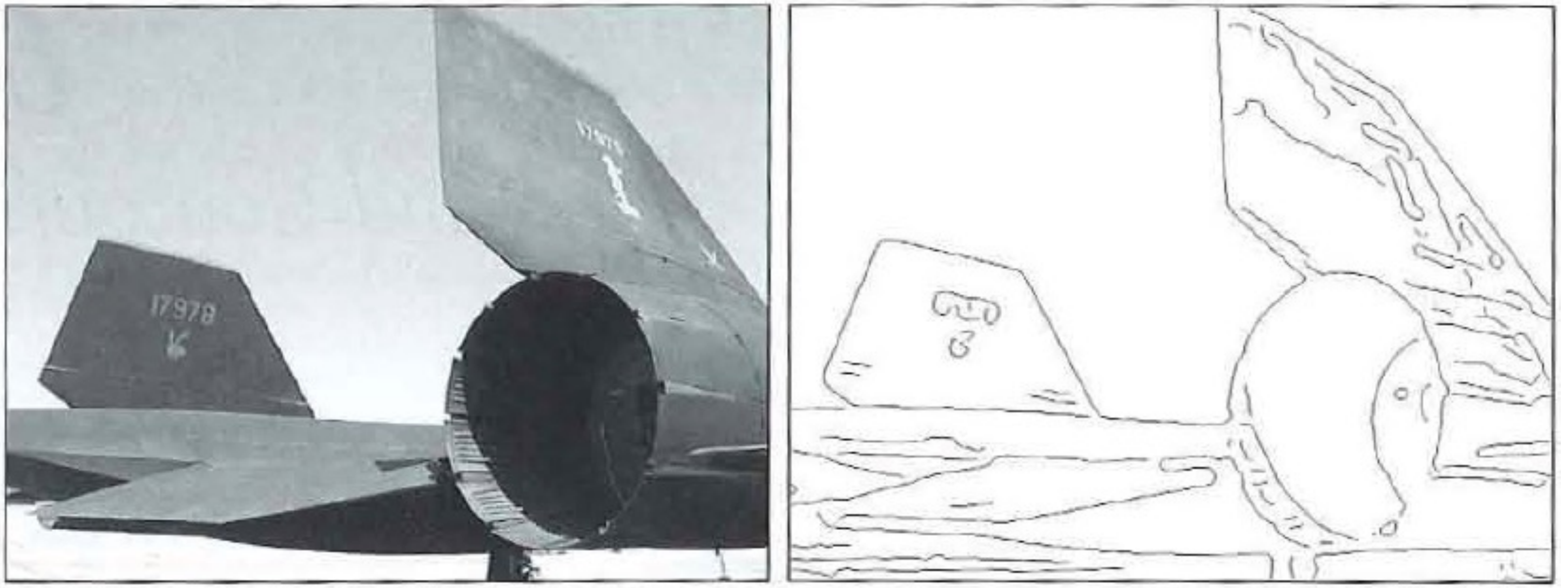
INF250

Filters: Edges and contours

Edges and contours

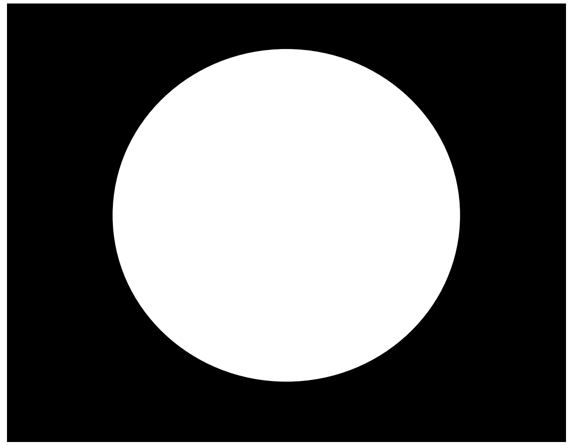
- Edges and contours are important for humans and animals vision
- A contour of an object can easily be used to reconstruct the object itself
- An edge can be described as image positions where the local intensity vary dramatically in relation to its surroundings

How can edges and contours be localised ?



(a) (b)

Figure 6.1 Edges play an important role in human vision. Original image (a) and edge image (b).



Gradient based edge detection

$$f'(x) = \frac{df}{dx}(x)$$

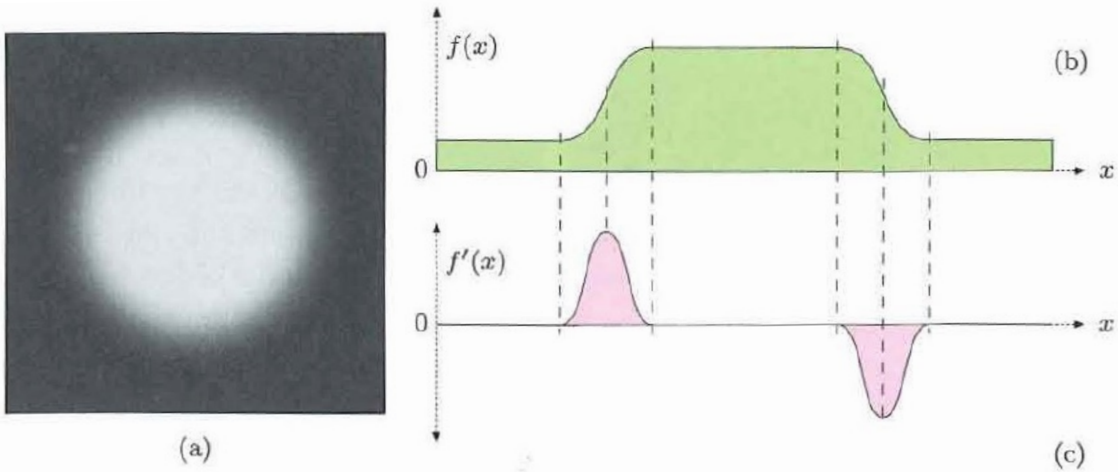


Figure 6.2 Sample image and first derivative in one dimension: original image (a), horizontal intensity profile $f(x)$ along the center image line (b), and first derivative $f'(x)$ (c).

Deriving a discrete signal

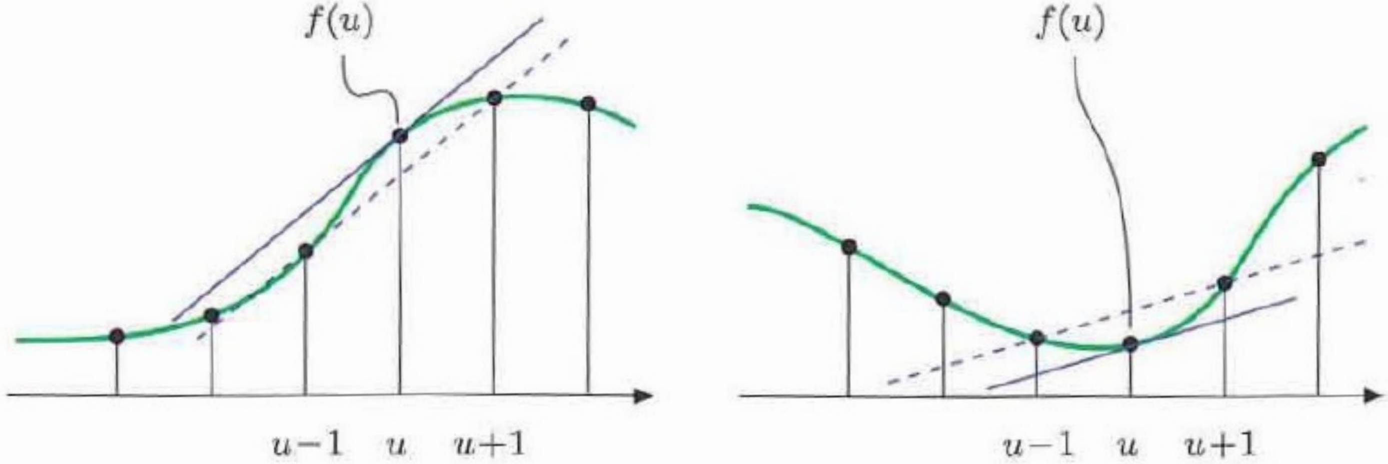


Figure 6.3 Estimating the first derivative of a discrete function. The slope of the straight (dashed) line between the neighboring function values $f(u-1)$ and $f(u+1)$ is taken as the estimate for the slope of the tangent (i.e., the first derivative) at $f(u)$.

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2} = 0.5 \cdot \big(f(u+1) - f(u-1)\big)$$

Partial derivation and gradient

$$\frac{\partial I}{\partial u}(u,v) \qquad \text{and} \qquad \frac{\partial I}{\partial v}(u,v)$$

$$\nabla I(u,v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u,v) \\ \frac{\partial I}{\partial v}(u,v) \end{bmatrix}$$

The magnitude of the gradient is invariant of image rotation, important for isotropic localization of edges.

$$|\nabla I|(u,v) = \sqrt{\left(\frac{\partial I}{\partial u}(u,v)\right)^2 + \left(\frac{\partial I}{\partial v}(u,v)\right)^2}$$

Simple edge operators
Linear filter and derivation

Horizontal and vertical component of the gradient can be obtained by the linear filters

$$H_x^D = \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} = 0.5 \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$$

$$H_y^D = \begin{bmatrix} -0.5 \\ \mathbf{0} \\ 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$

Gradient filters

Prewitt operator

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

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

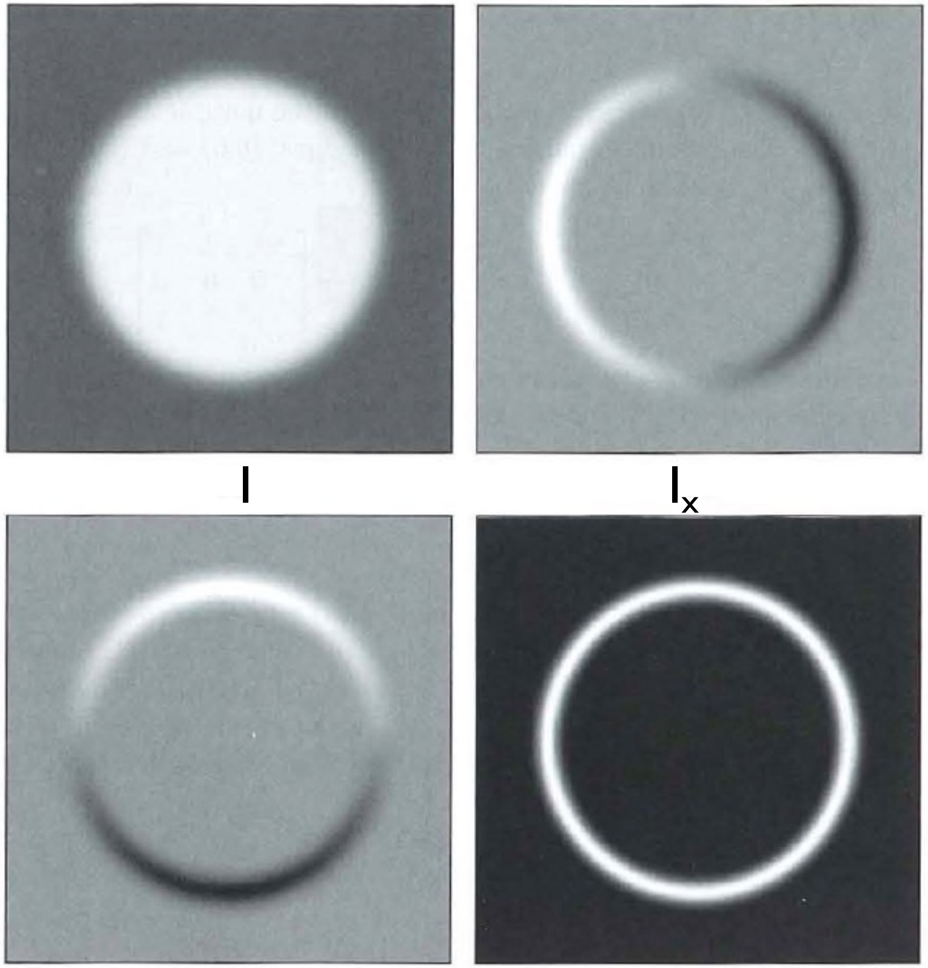
$$H_y^P = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix} * \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Gradient filters

Sobel operator – assigns higher weight to center line and column than the prewitt operator

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

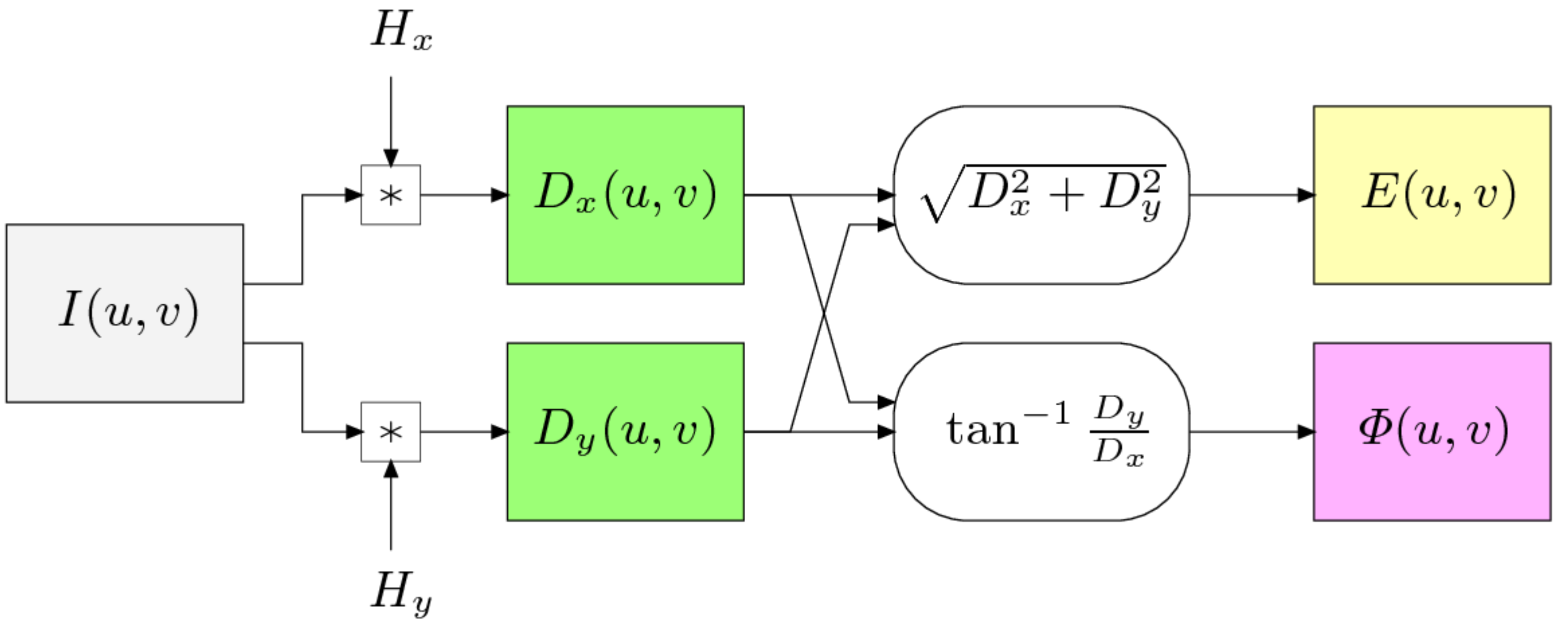
Gradient - from 1 to 2 dimensions



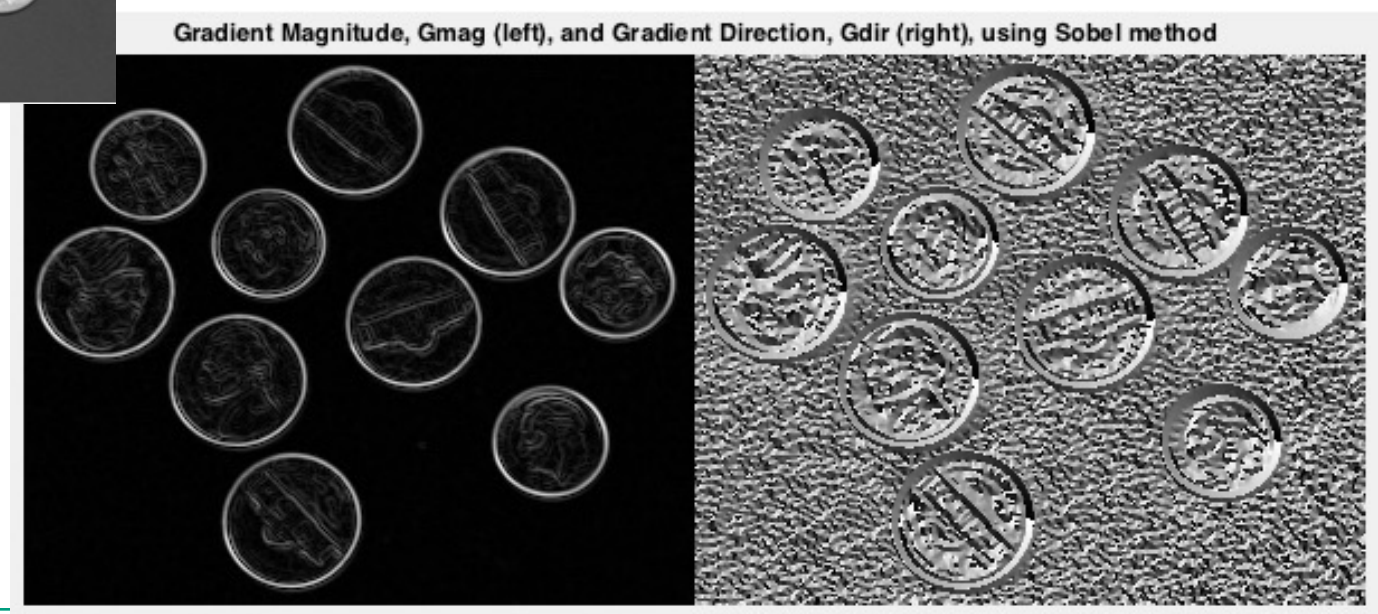
I $\qquad$ I_x

I_y $\qquad$ $|\nabla I|$

Gradientfiltere (SOBEL)

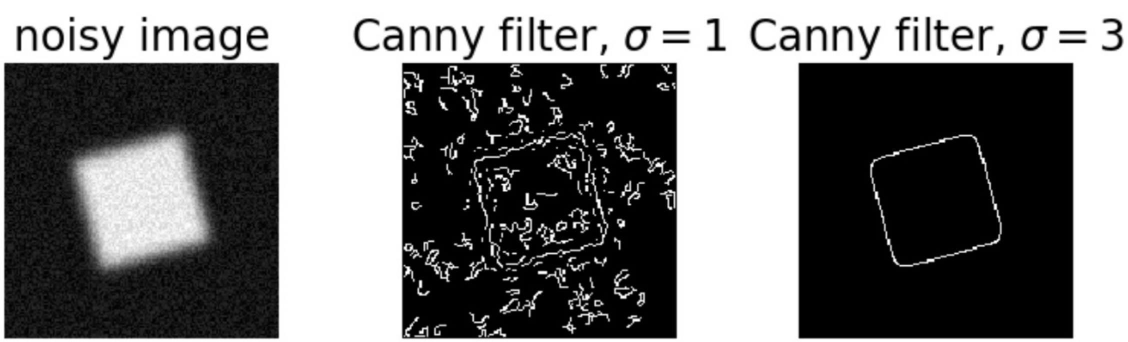


Gradientfilter, Orientation, Strength

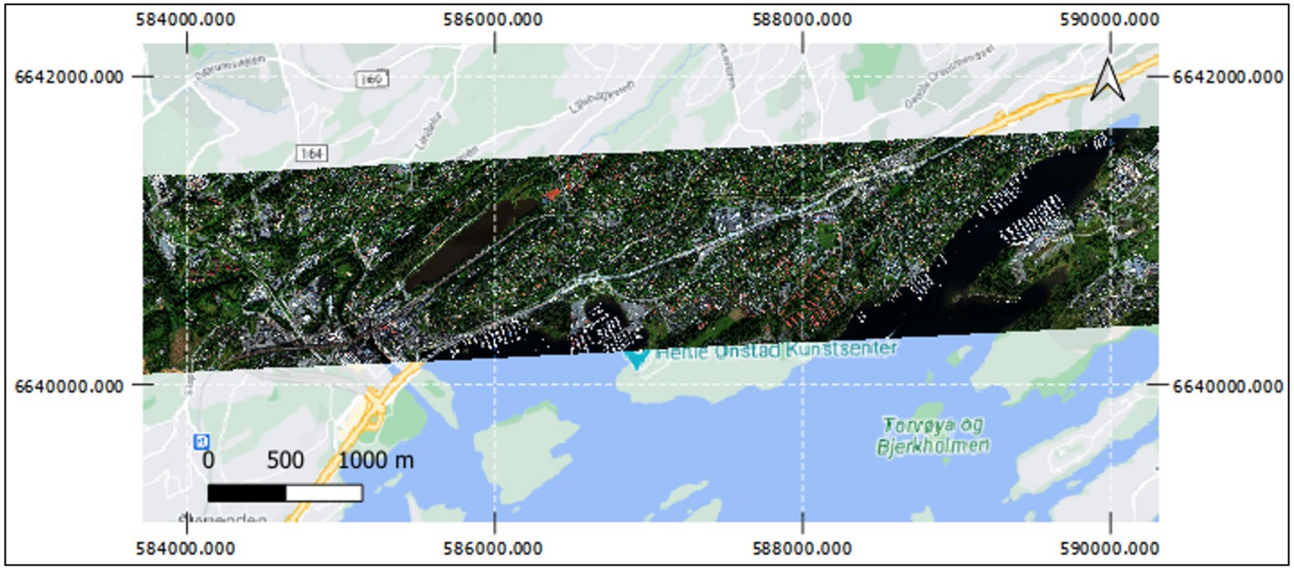


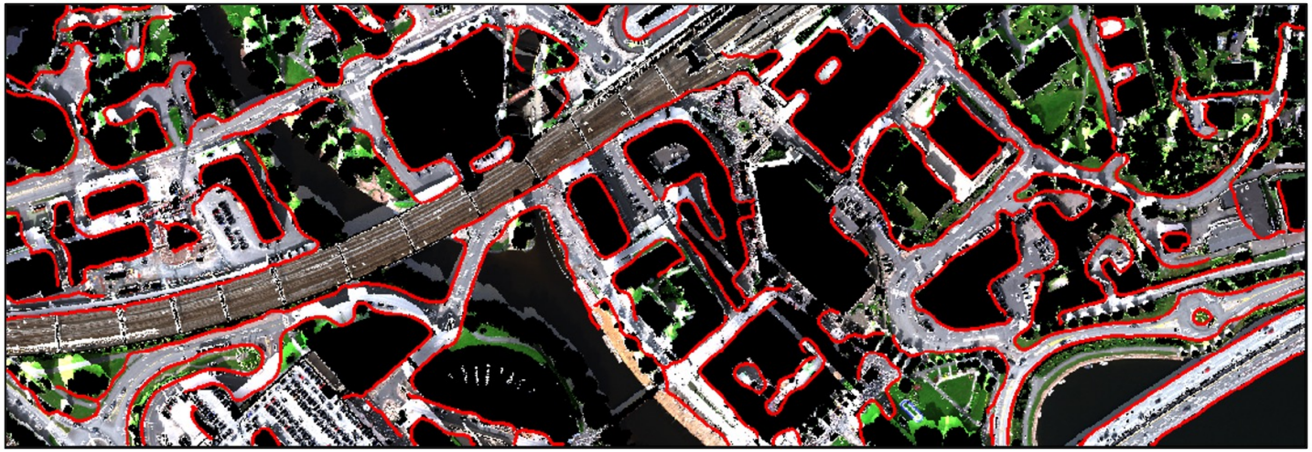
Canny filter

- Multi stage edge detector
- Based on the derivative of a Gaussian filtered image to compute the intensity of the gradients
- http://scikit-image.org/docs/dev/auto_examples/edges/plot_canny.html#sphx-glr-auto-examples-edges-plot-canny-py
- Two input parameters:
  - Sigma of gaussian smoothing
  - Hysterisis thresholding (min and max)

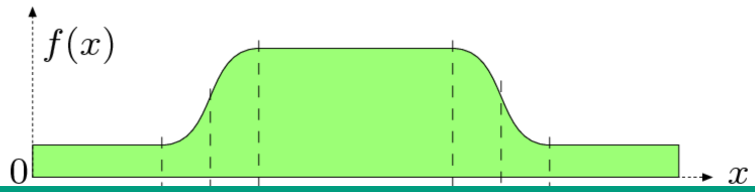


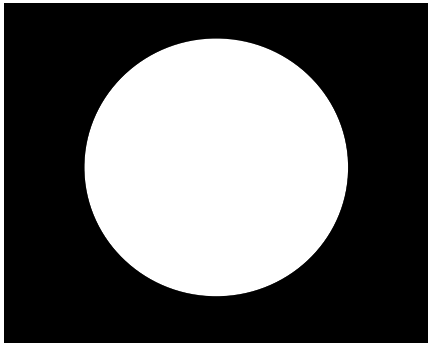
Canny filter for road detection



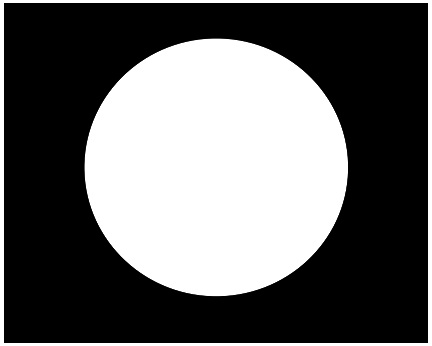


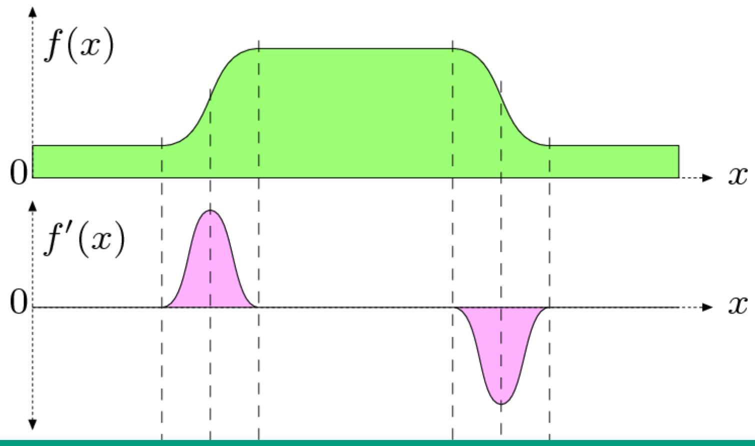
Laplace filter



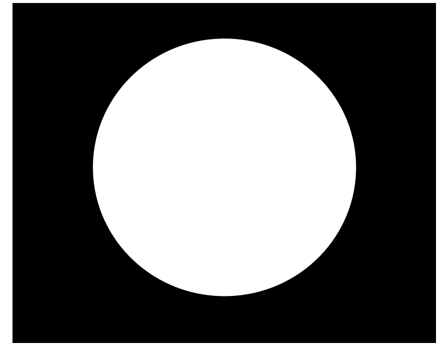


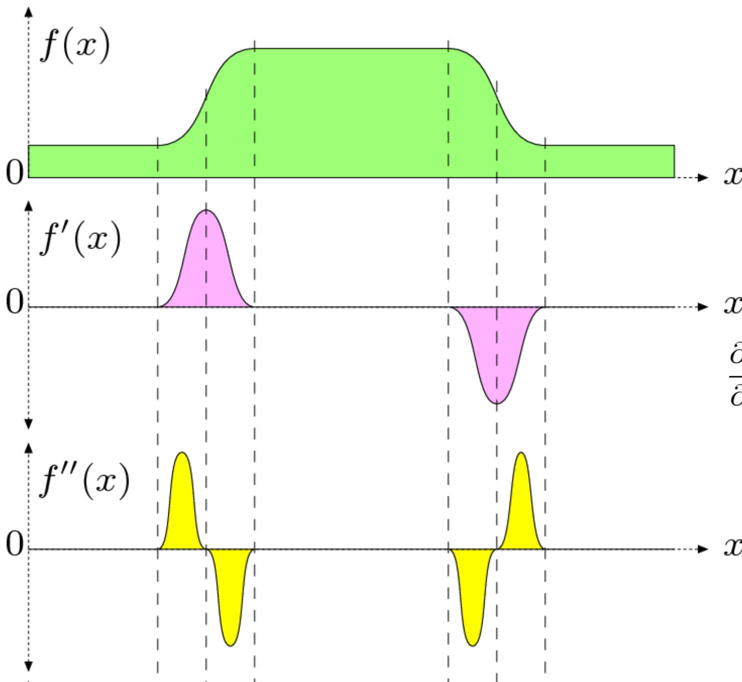
Laplace filter





Laplace filter

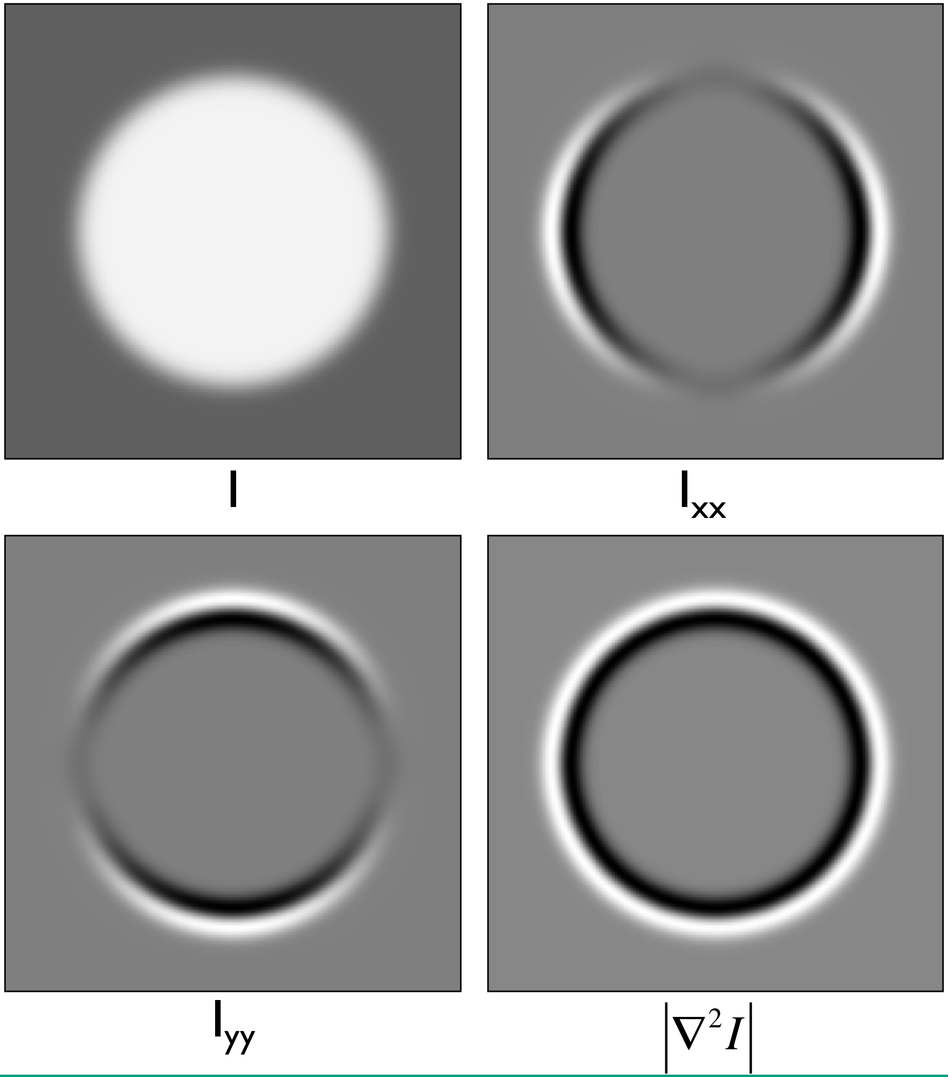




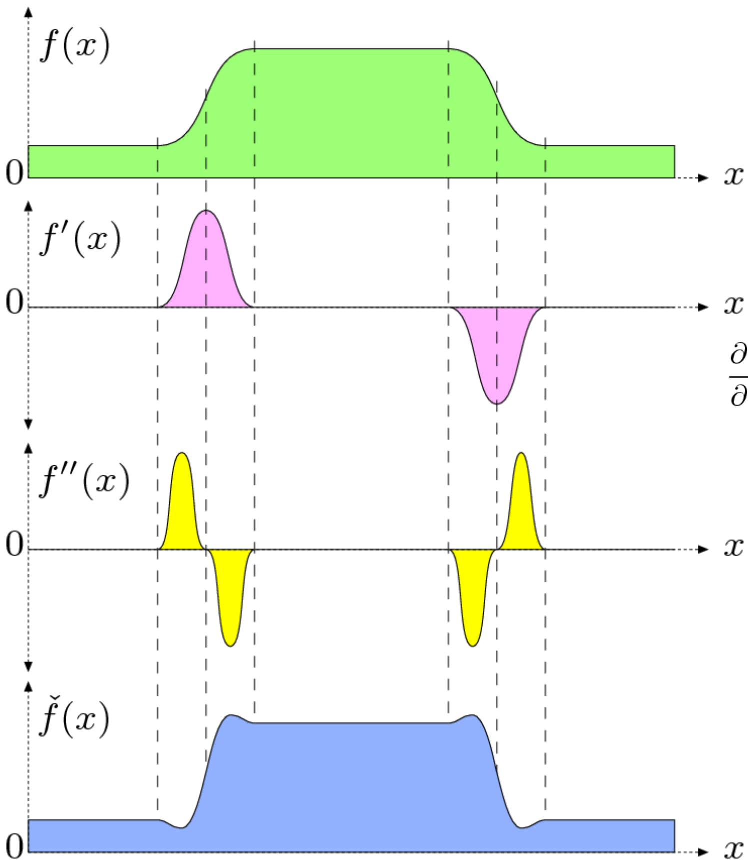
$$\frac{\partial^2 f}{\partial^2 x} \equiv H_x^L = \begin{bmatrix} 1 & -\mathbf{2} & 1 \end{bmatrix} \quad \text{and} \quad \frac{\partial^2 f}{\partial^2 y} \equiv H_y^L = \begin{bmatrix} 1 \\ -\mathbf{2} \\ 1 \end{bmatrix}$$

$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -\mathbf{4} & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Laplace – edge sharpening



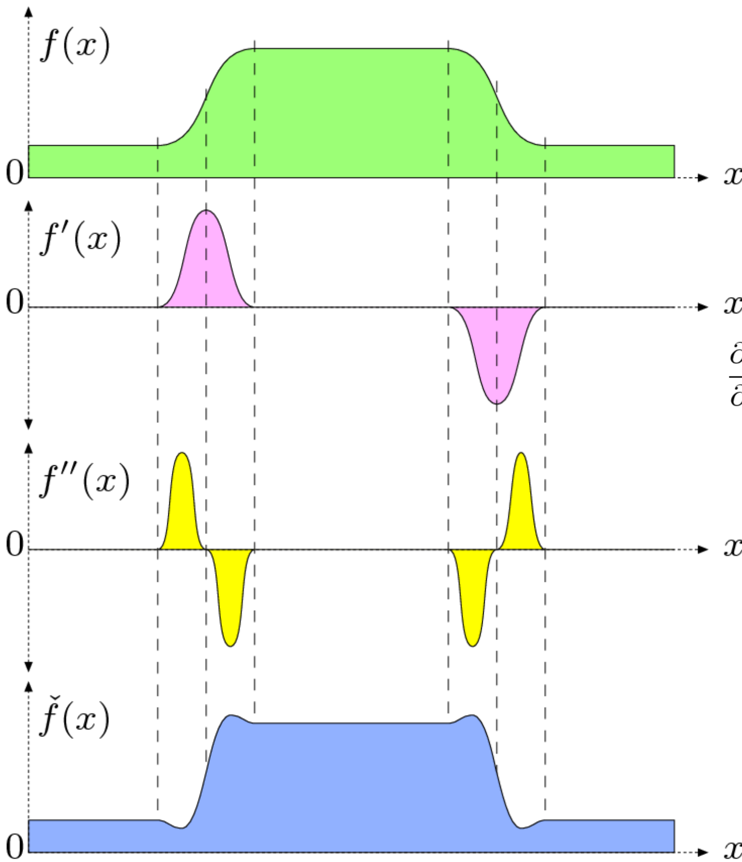
Laplace filter



$$\frac{\partial^2 f}{\partial^2 x} \equiv H_x^L = \begin{bmatrix} 1 & \mathbf{-2} & 1 \end{bmatrix} \quad \text{and} \quad \frac{\partial^2 f}{\partial^2 y} \equiv H_y^L = \begin{bmatrix} 1 \\ \mathbf{-2} \\ 1 \end{bmatrix}$$

$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & \mathbf{-4} & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Laplace filter



$$\frac{\partial^2 f}{\partial^2 x} \equiv H_x^L = \begin{bmatrix} 1 & -\mathbf{2} & 1 \end{bmatrix} \quad \text{and} \quad \frac{\partial^2 f}{\partial^2 y} \equiv H_y^L = \begin{bmatrix} 1 \\ -\mathbf{2} \\ 1 \end{bmatrix}$$

$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -\mathbf{4} & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$\check{f}(x) = f(x) - w \cdot f''(x)$$

Low pass filter

- Filter ⇔ Convolution kernel
- Uniform weight convolution
  - The simples filter (kernel)

$$\begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix} = \frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$ = ”Lowpassfilter”

Lowpass – passing low frequencies, coarse structures, long wavelenghts

High pass filter

- Passing high frequencies / fine structures / short wavelengths
- High pass filtrered image: Use a low pass filter and subtract result from the original image

$H = I - L$

- H = high pass filtered image, I = original image, L = low pass filtrered image

$h \otimes I = i \otimes I - l \otimes I$

- h, l and i are high pass, low pass and identity filter

High pass filter cont.

$$i = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = identity$$

$$h = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} * \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = highpassfilter$$

$$h = \frac{1}{9} * \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} = highpassfilter$$

Making the image sharper

- Sharping kernel kan be made by
  - Blending / mixing / overlaying an original image with a high pass filtered image (weighted)

$$S = (1-f)*I + f*H$$

- S is the result, f is a factor (0,1), I is the original image, H is the high pass filtered image

$$s \otimes I = (1-f)*i \otimes I + f*h \otimes I$$

$$s = (1-f)*i + f*h$$

Making the image sharper cont.

$$s = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} + f\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Producing

$$s = \frac{1}{9}\begin{bmatrix} -f & -f & -f \\ -f & (9-f) & -f \\ -f & -f & -f \end{bmatrix}$$

S=s $\otimes$I

where s is the sharping filter, I is the original image and S is the result

Unsharpen Mask (USM)

- USM is a technique to increase the sharpness using edge detections
- Popular within astronomy, digital printing, web publications etc
- USM was used in classical photography where sharp photos were constructed from a combination of the original photo and a smoothed (unsharp) photo
- The human vision uses the same principle

USM algorithm

- The mask M is generated by subtracting a smoothed version of the image from the original image (I)

$$M \leftarrow I - (I * \tilde{H}) = I - \tilde{I}$$

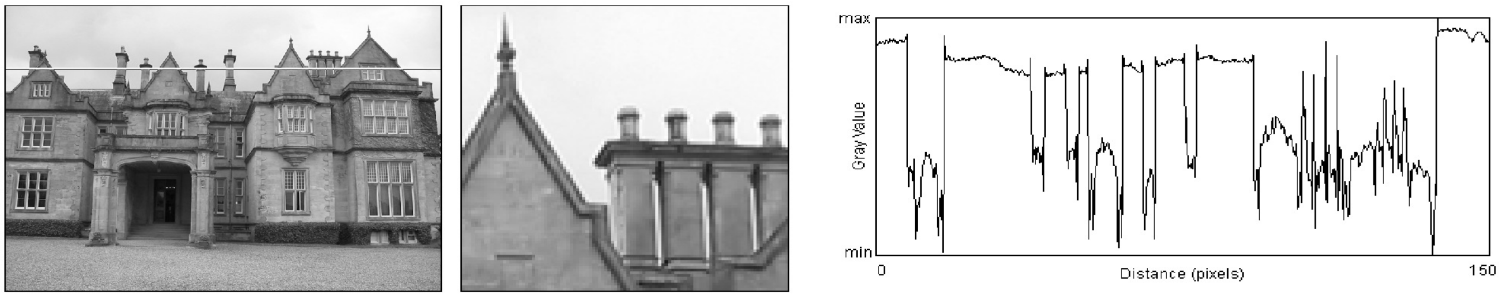
where the kernel of the smoothing filter is assumed to normalized

- To obtain the sharpened image the mask is added to the original image, weighted by a factor a, which controls the amount of sharpening

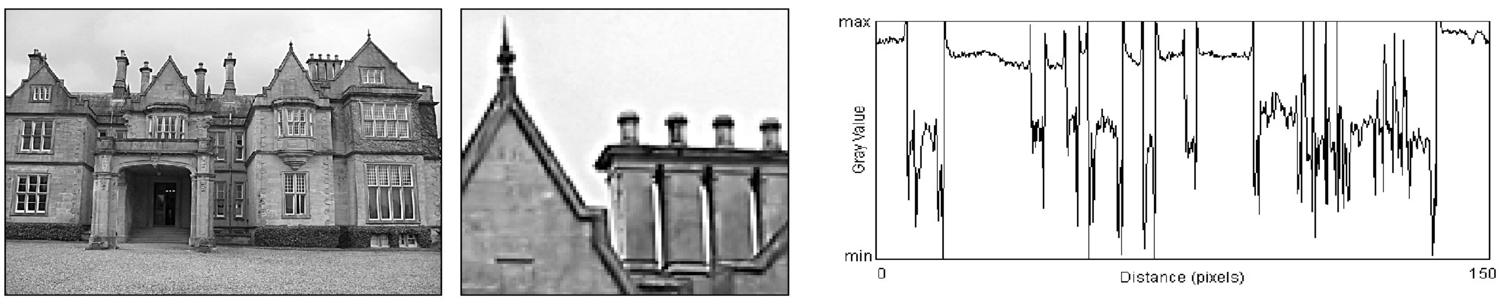
$$\breve{I} \leftarrow I + a \cdot M = I + a \cdot \left(I - \tilde{I}\right) = \left(1 + a\right) \cdot I - a \cdot \tilde{I}$$

Smoothing filter used in USM

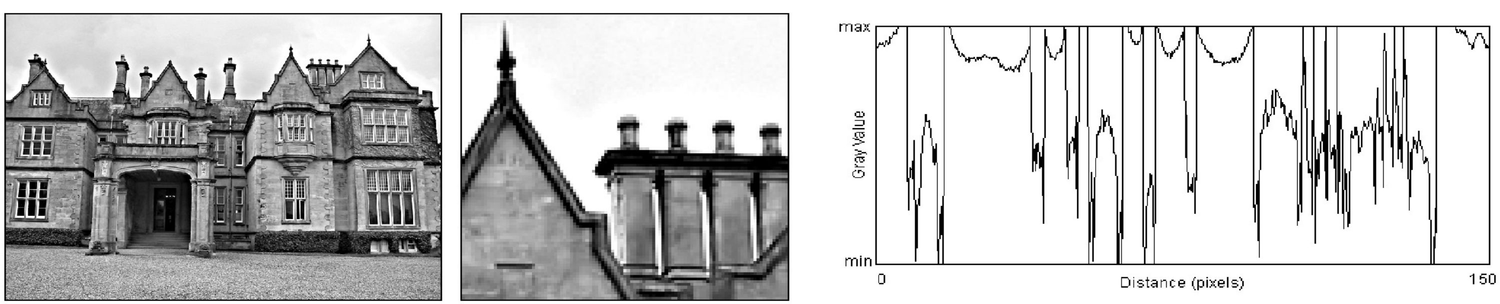
Any filter can be used but Gaussian filters with variable radius are most common



(a) Original (b) (c)



(d) $\sigma = 2.5$ (e) (f)



(g) $\sigma = 10.0$ (h) (i)

```python
import numpy as np
import matplotlib.pyplot as plt
from skimage import filters

from skimage import data, img_as_float
from skimage.data import astronaut
from skimage.color import rgb2gray

astro = rgb2gray(img_as_float(data.astronaut()))

astro = astro[30:180,150:300]
plt.imshow(astro,'gray')
```

```python
#sobel and prewitt filter
astro_sobel = filters.sobel(astro)
plt.imshow(astro_sobel,'gray')

astro_sobel_h = filters.sobel_h(astro)
plt.imshow(astro_sobel_h,'gray')

astro_sobel_v = filters.sobel_v(astro)
plt.imshow(astro_sobel_v,'gray')

astro_prewitt = filters.prewitt(astro)
plt.imshow(astro_prewitt,'gray')

from skimage import feature
edges1 = feature.canny(astro, sigma=1)
plt.imshow(edges1,'gray')

edges2 = feature.canny(astro, sigma=3)
plt.imshow(edges2,'gray')
```

```python
# laplace filter
from skimage.filters import laplace
astro_lap = laplace(astro)
plt.imshow(astro_lap,'gray',vmin=0., vmax=0.2)
astro_sharp = astro-2*astro_lap
plt.imshow(astro_sharp, 'gray', vmin=0.4, vmax=0.9)
```

```python
# high pass filter
from skimage.filters import gaussian
plt.imshow(astro,'gray')
gaussastro = gaussian(astro, sigma=5)
plt.imshow(gaussastro,'gray')
astro_high = astro-(gaussastro)
plt.imshow(astro_high,'gray', vmin=0., vmax=0.2)


# unsharpening mask to sharpen the image
amount = 2
usm_astro = astro + amount*(astro-gaussastro)
plt.imshow(usm_astro, 'gray', vmin=0.2, vmax=0.9)

plt.imshow(astro,'gray', vmin=0.2, vmax=0.9)

plt.hist(astro.ravel(),256,[0,1], color='black')
plt.show()

plt.hist(gaussastro.ravel(),256,[0,1], color='black')
plt.show()
```