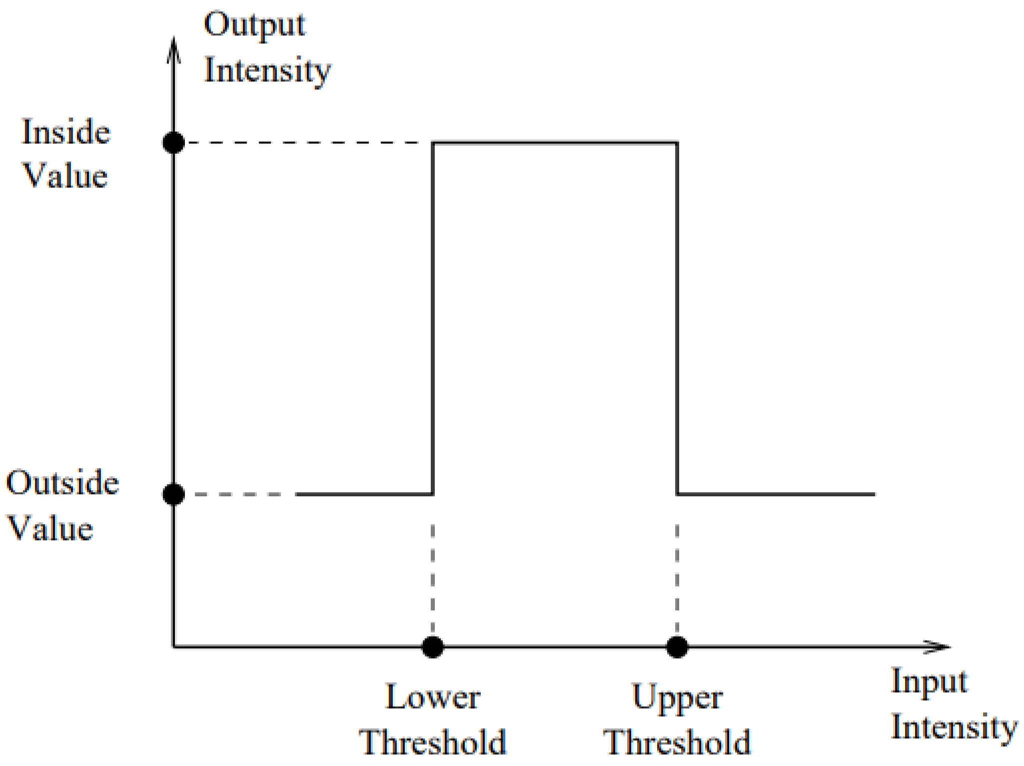
Filtering using ITK Toolkit

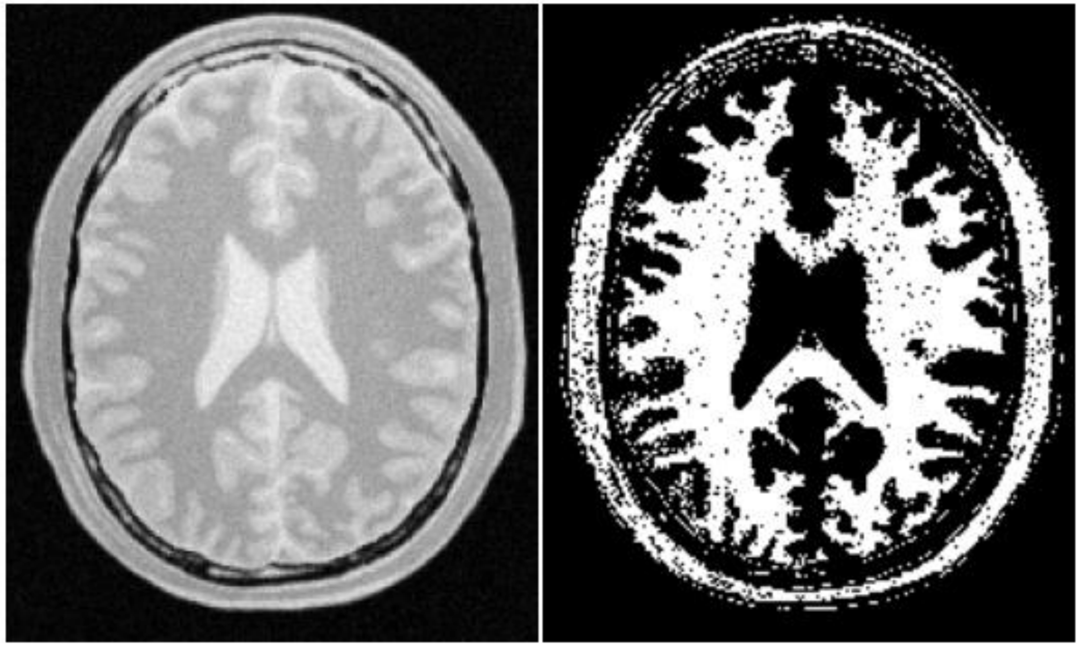
Kalinga Bandara - 200068R
Department of Electronic and Telecommunications Engineering
University of Moratuwa

2.1 Thresholding

2.1.1 Binary Thresholding

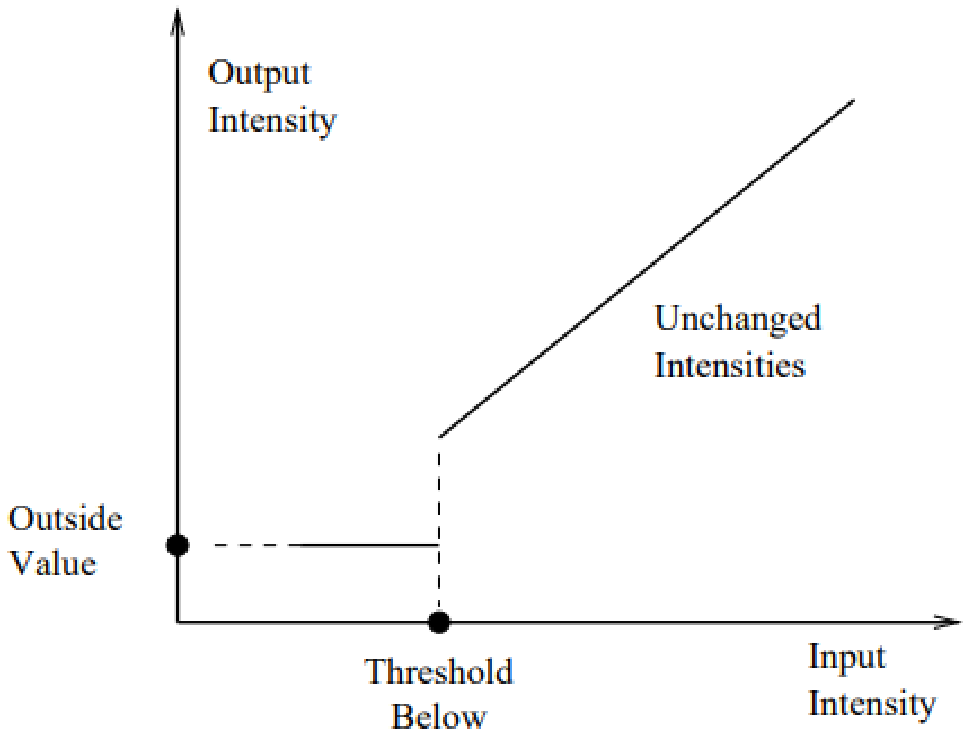
BinaryThresholdImageFilter

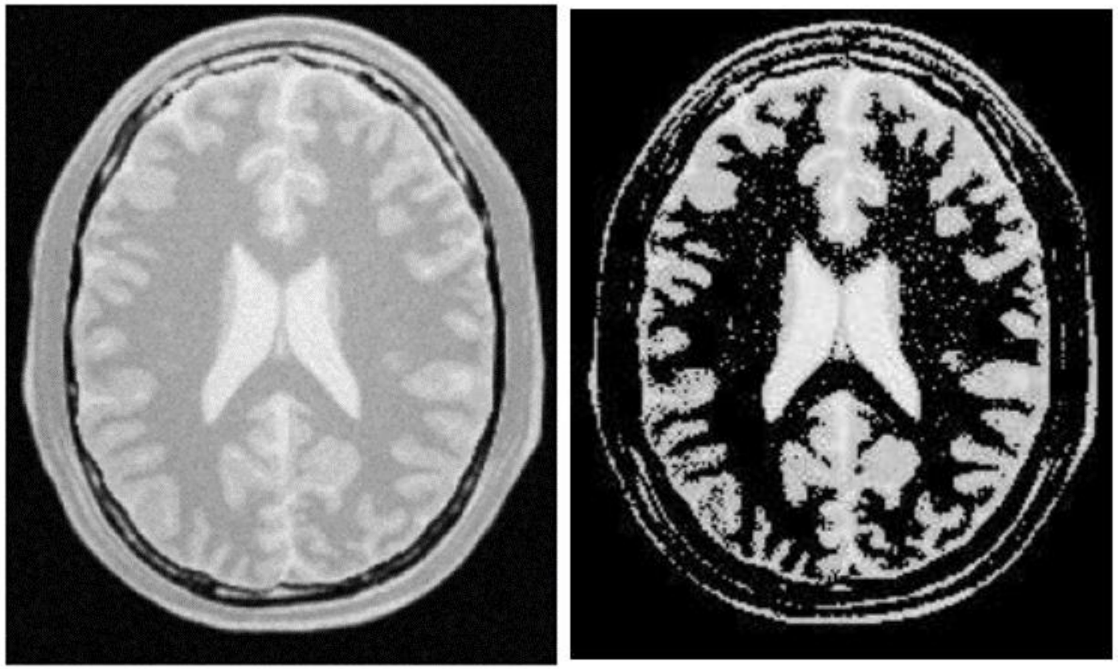




2.2.1 General Thresholding

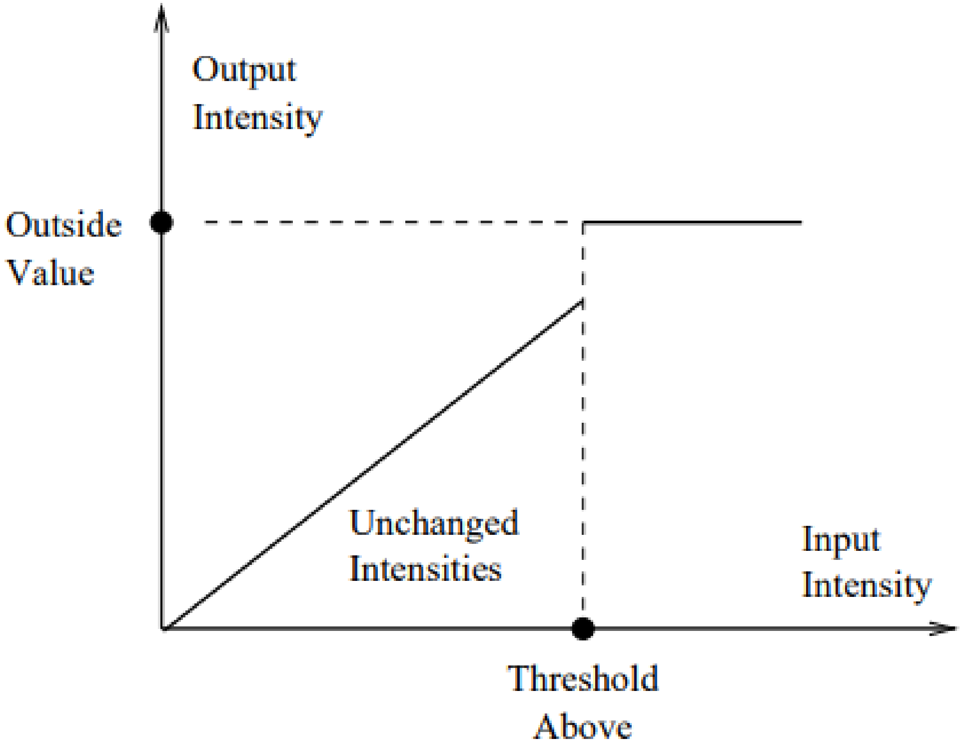
ThresholdImageFilter using threshold-below mode

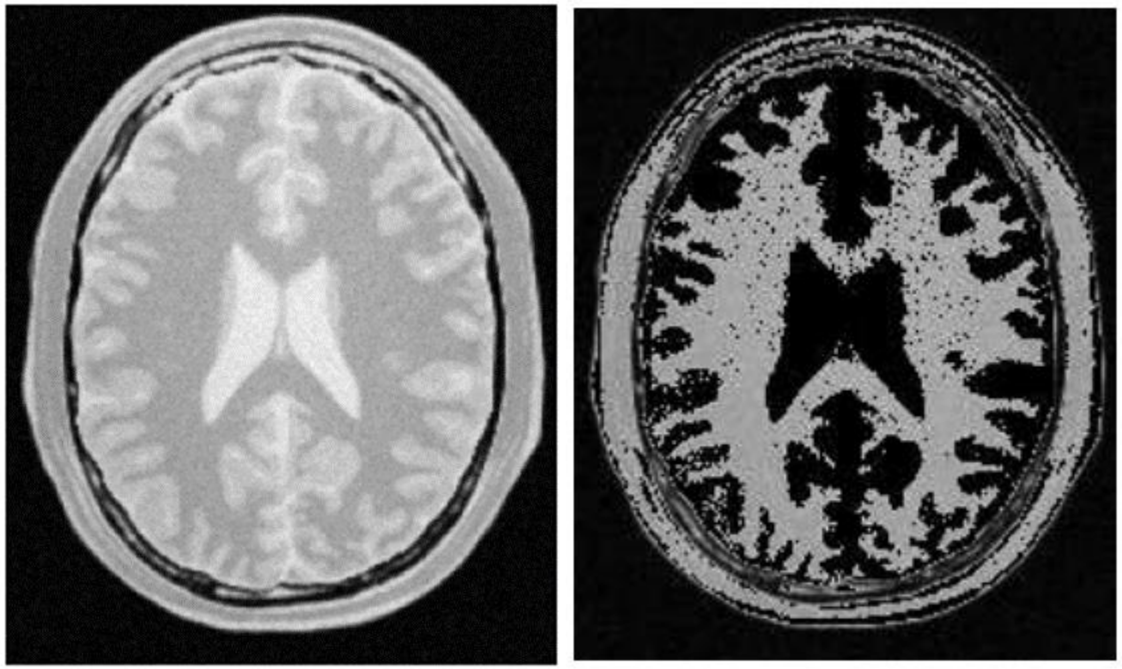




2.2.1 General Thresholding (continued)

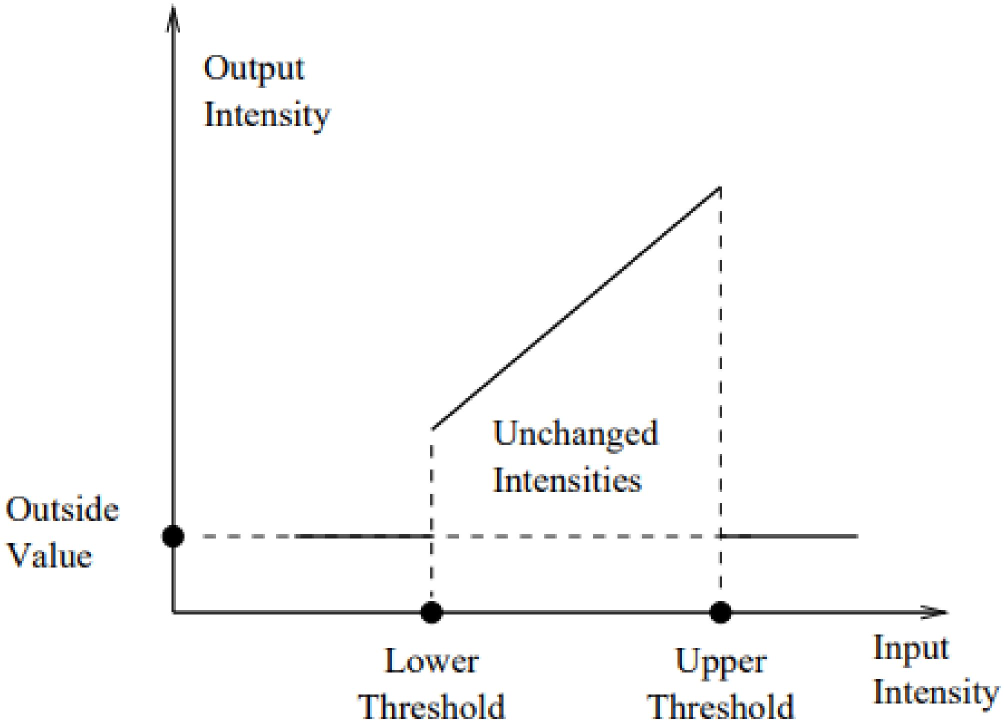
ThresholdImageFilter using threshold-above mode

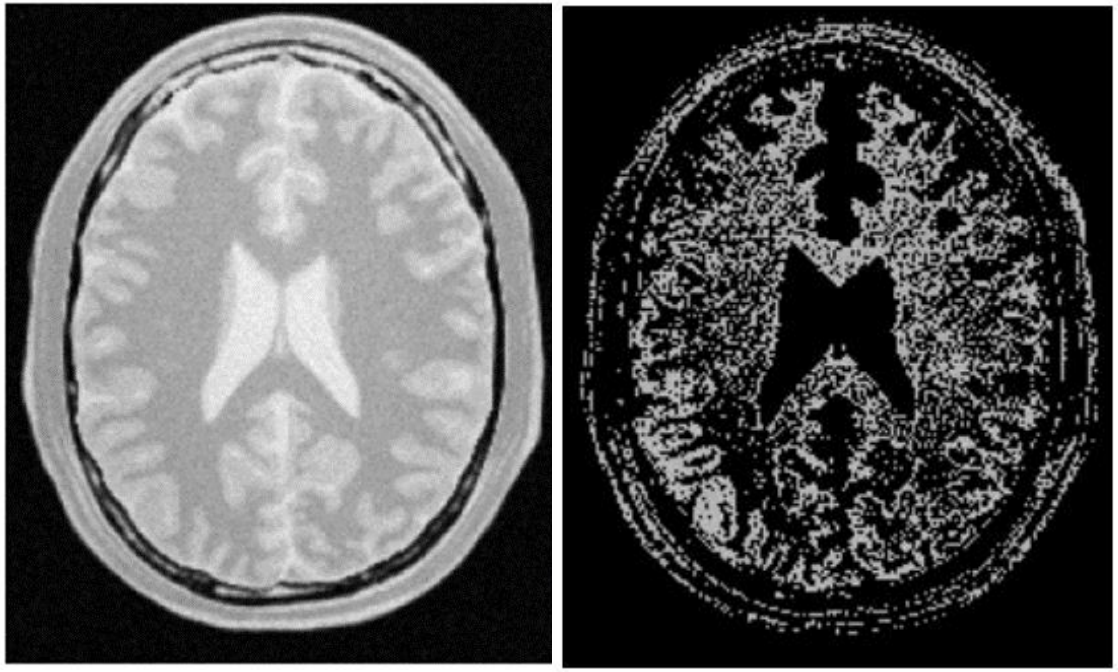




2.2.1 General Thresholding (continued)

ThresholdImageFilter using threshold-outside mode





2.2 Edge Detection

2.2.1 Canny Edge Detection

Four stage algorithm in Canny Edge Detection

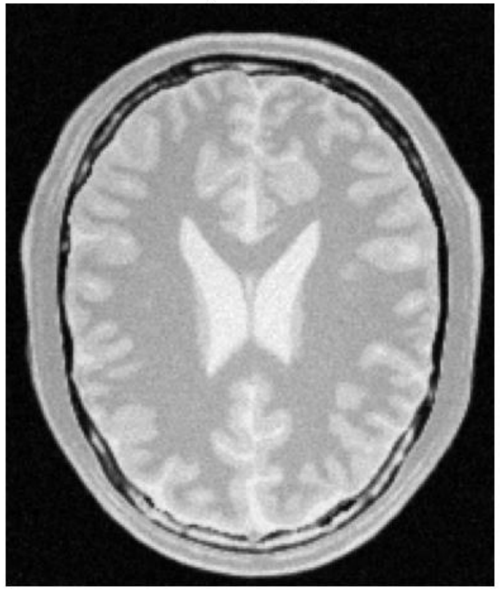
1. Filter the image with derivative of Gaussian.

2. Find magnitude and orientation of the gradient.

3. Non-maximum Suppression (Thinning wide ridges down to single pixel width)

4. Hysteresis thresholding (using two thresholds, high threshold to start the edge curves and low threshold to continue them)

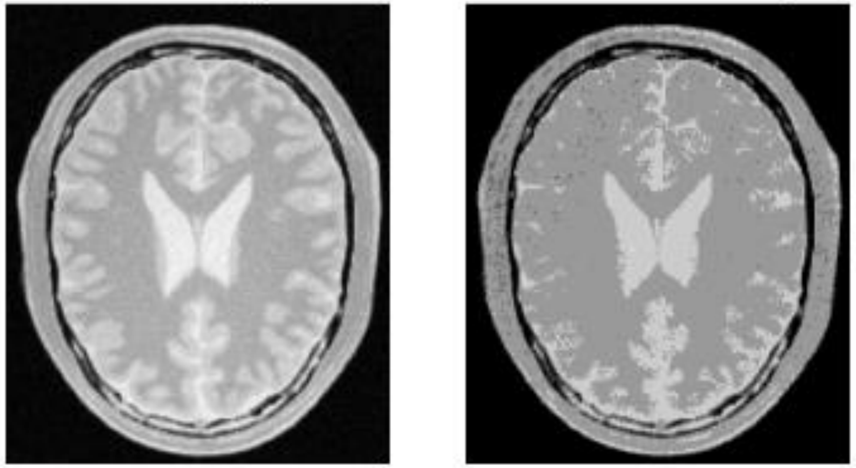
Original Image



Filtered Image

2.3 Casting and Intensity Mapping

2.3.1 Linear Mappings



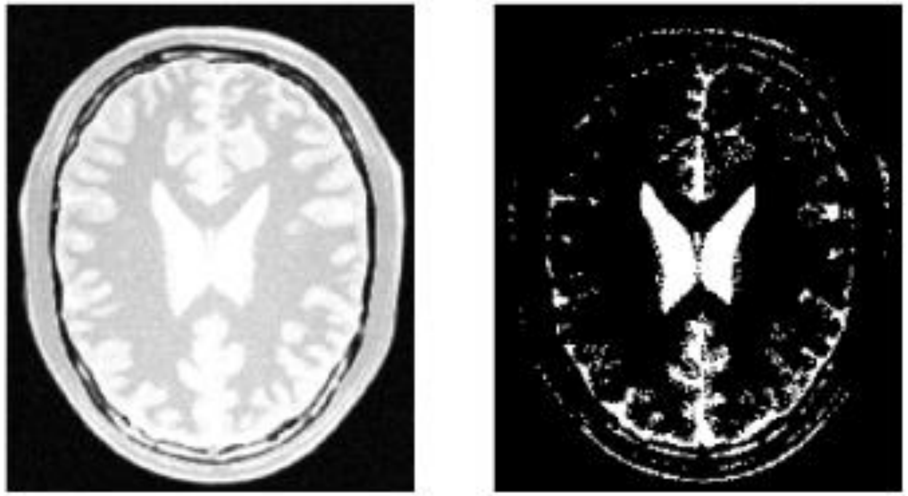
1. **CastImageFilter**: Casting every pixel to the type of the output image.

2. **RescaleIntensityImageFilter**: Linearly scaling the pixel values between given two values.

$$outputPixel = (inputPixel - inpMin) \times \frac{(outMax - outMin)}{(inpMax - inpMin)} + outMin$$

3. **ShiftScaleImageFilter**: A linear transformation

$$outputPixel = (inputPixel + Shift) \times Scale$$



4. **NormalizeImageFilter**: Transforming gray levels to have zero mean and unit variance

$$outputPixel = \frac{(inputPixel - mean)}{\sqrt{variance}}$$

2.3.2 Non-Linear Mappings

SigmoidImageFilter

$$I' = (Max - Min) \cdot \frac{1}{\left(1 + e^{-\left(\frac{I-\beta}{\alpha}\right)}\right)} + Min$$

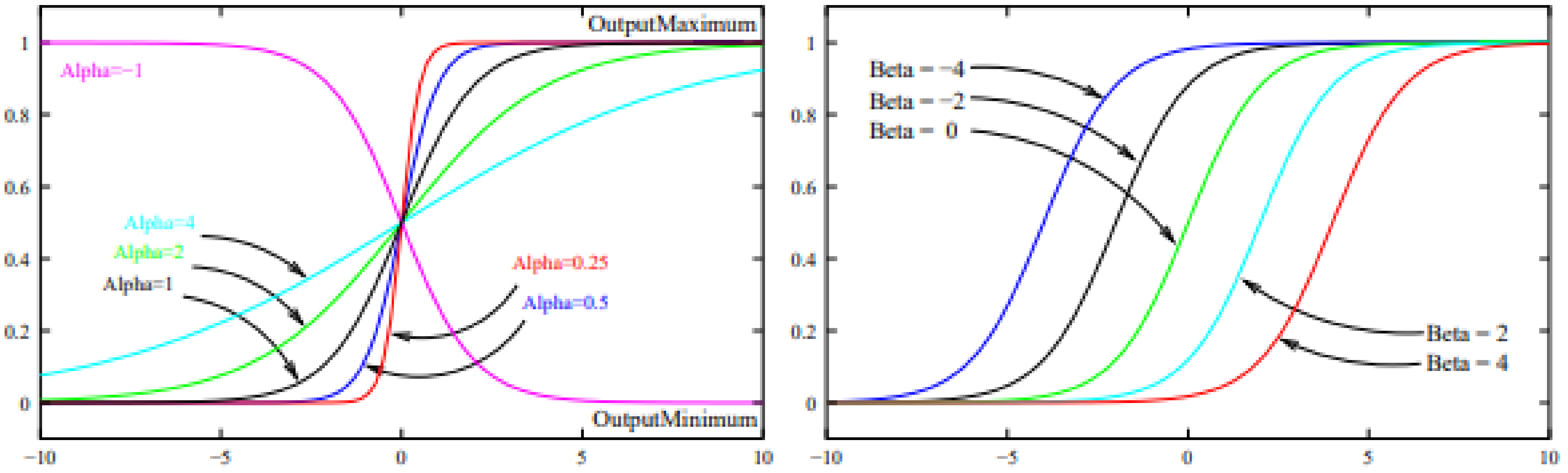
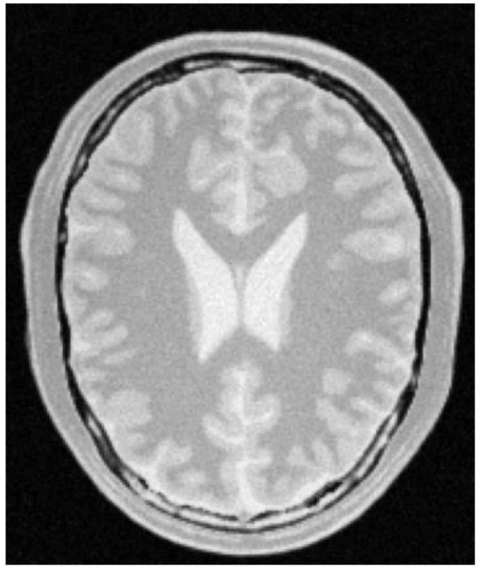
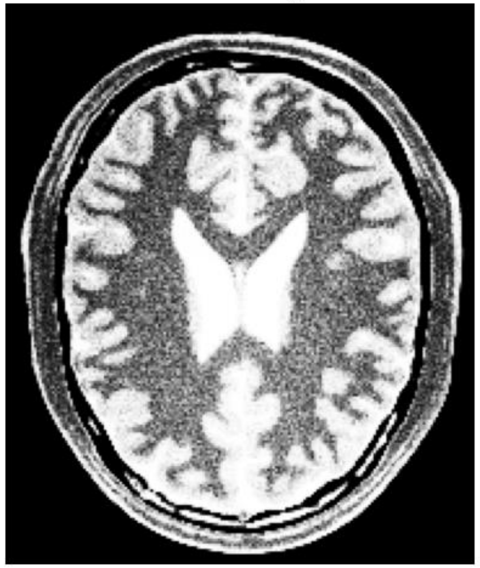


Figure 2.6: Effects of the various parameters in the SigmoidImageFilter. The alpha parameter defines the width of the intensity window. The beta parameter defines the center of the intensity window.

Original Image



Filtered Image



2.4 Gradients

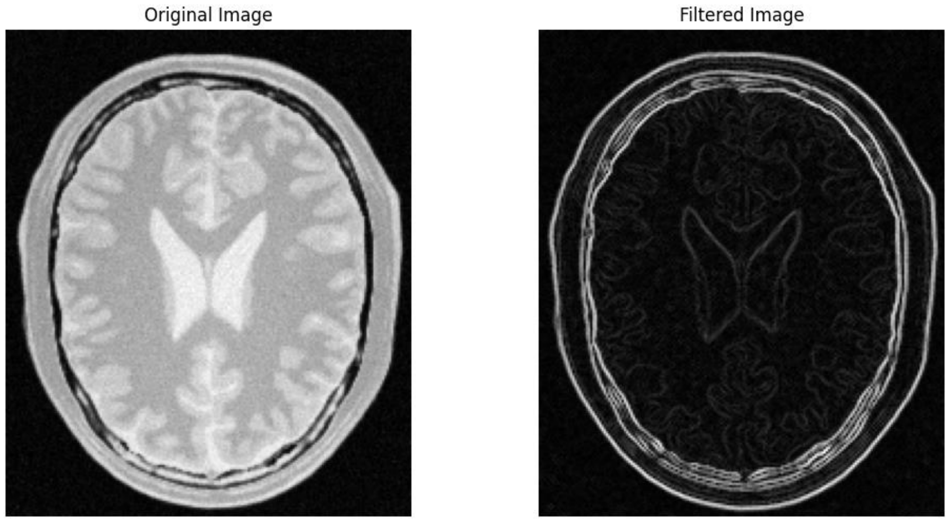
2.4.1 Gradient Magnitude

GradientMagnitudeFilter

For 2D case, gradient computation is equivalent to convolving the image with kernels of type

$$\begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$$

Then, adding the sum of squares and computing square root.



No smoothing is applied here. Hence, the filter is very sensitive to noise.

The dynamic range of the gradient magnitude is smaller than that input image.

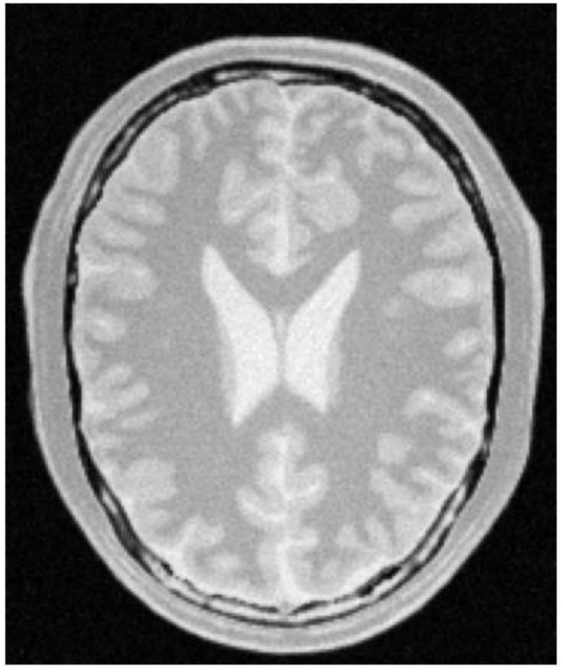
2.4.2 Gradient Magnitude with Smoothing

GradientMagnitudeRecursiveGaussianImageFilter

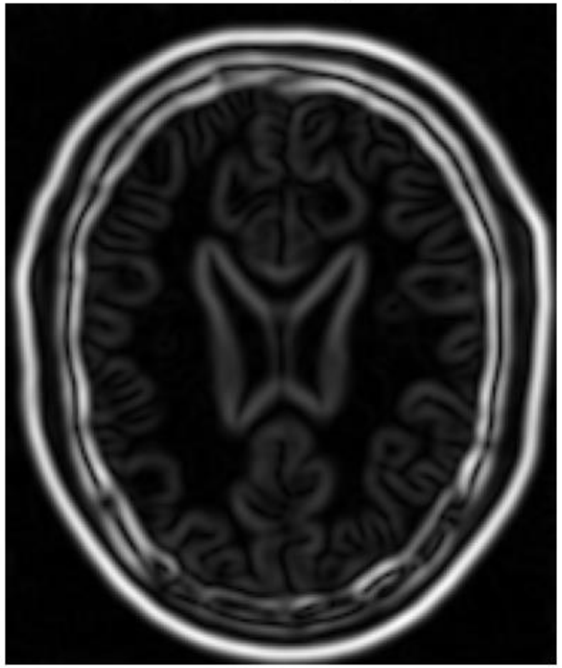
Internally, this is done by applying a IIR filter that convolves image with derivative of the Gaussian kernel.

Sigma should be appropriately selected to regulate the noise sensitivity.

Original Image



Filtered Image

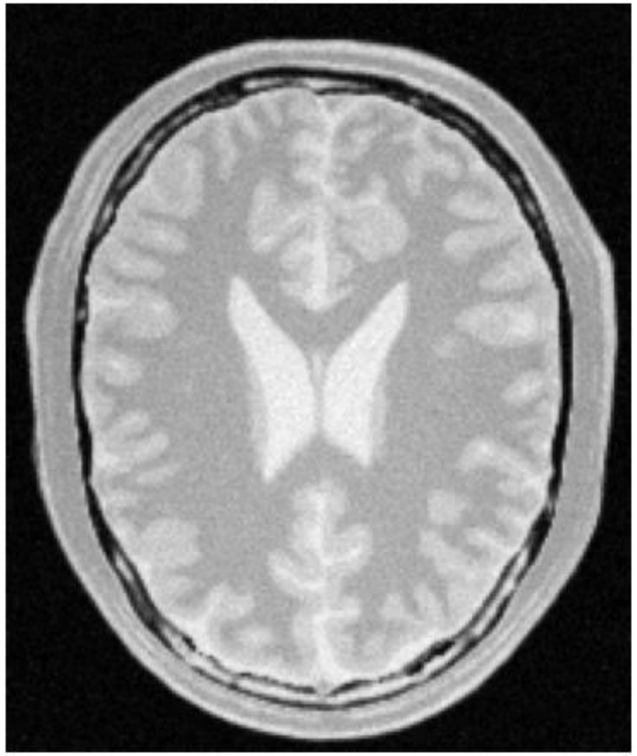


2.4.3 Derivative without Smoothing

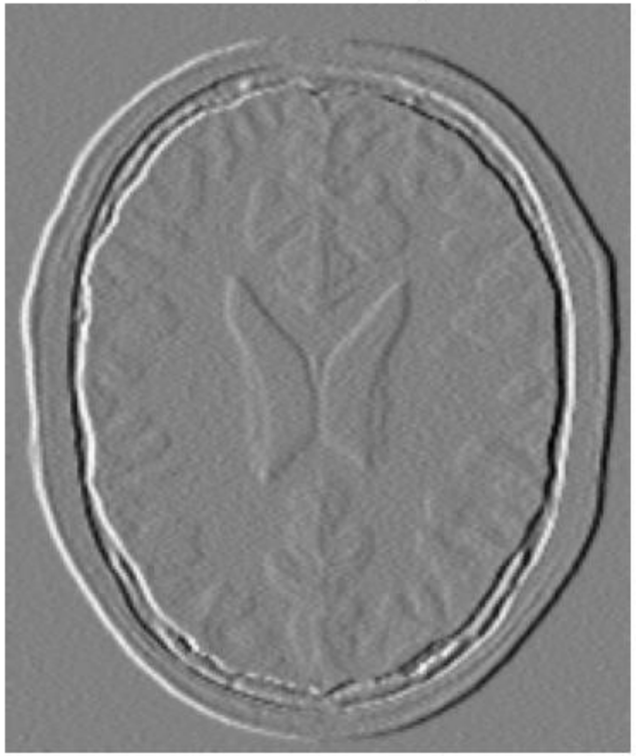
DerivativeImageFilter

This filter is used to compute the partial derivative of an image over a particular direction.

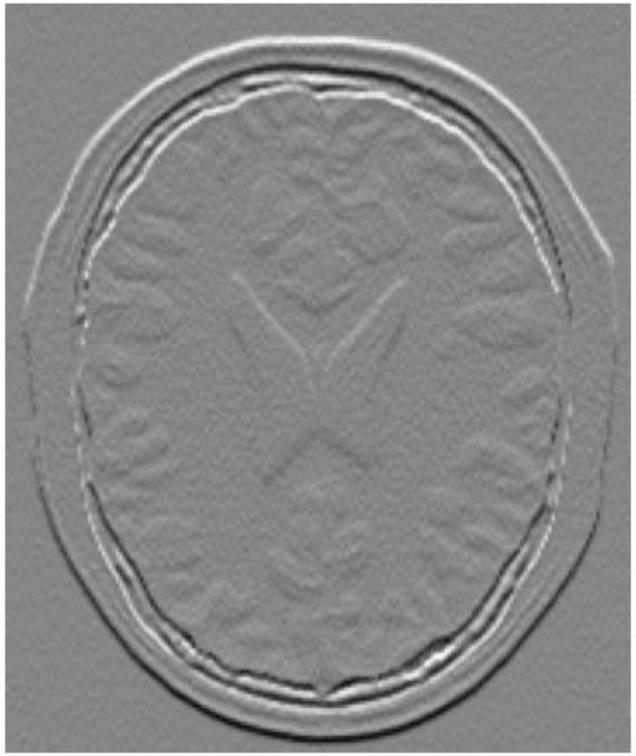
Original Image



First-order Derivative along x-direction



First-order Derivative along y-direction

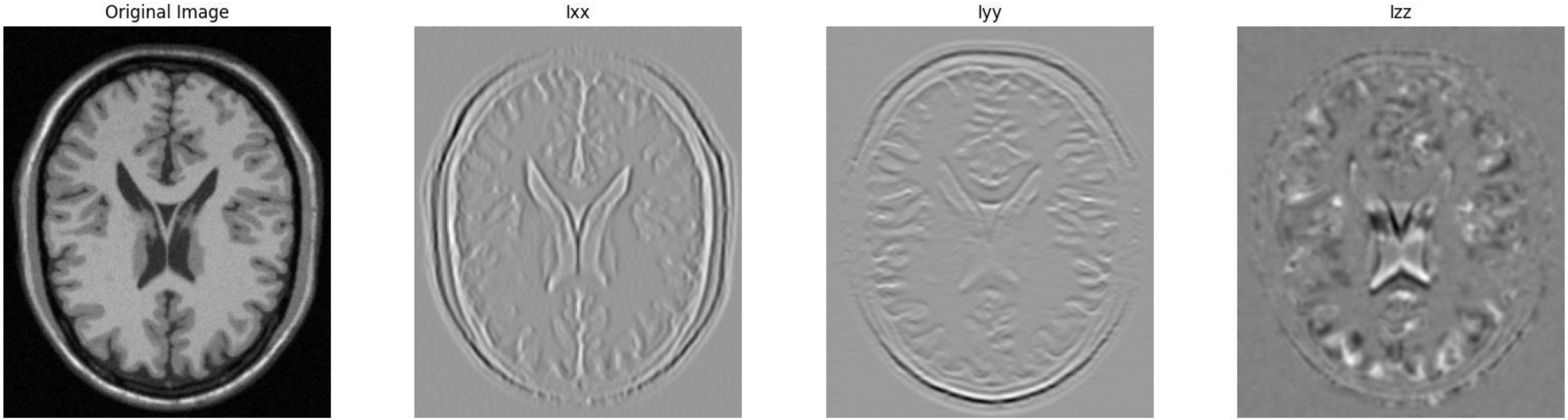


2.5 Second Order Derivatives

2.5.1 Second Order Derivatives

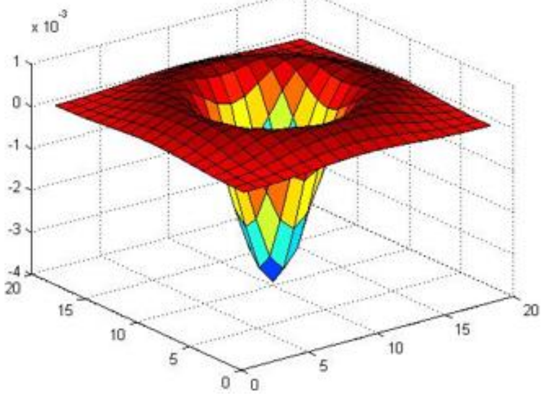
SecondDerivativeRecursiveGaussianImageFilter

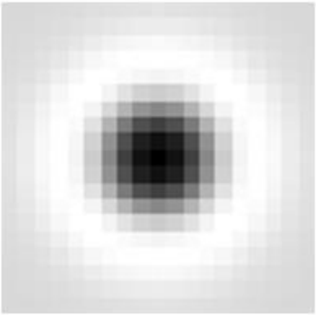
This filter computes second derivatives of an image over given axes.

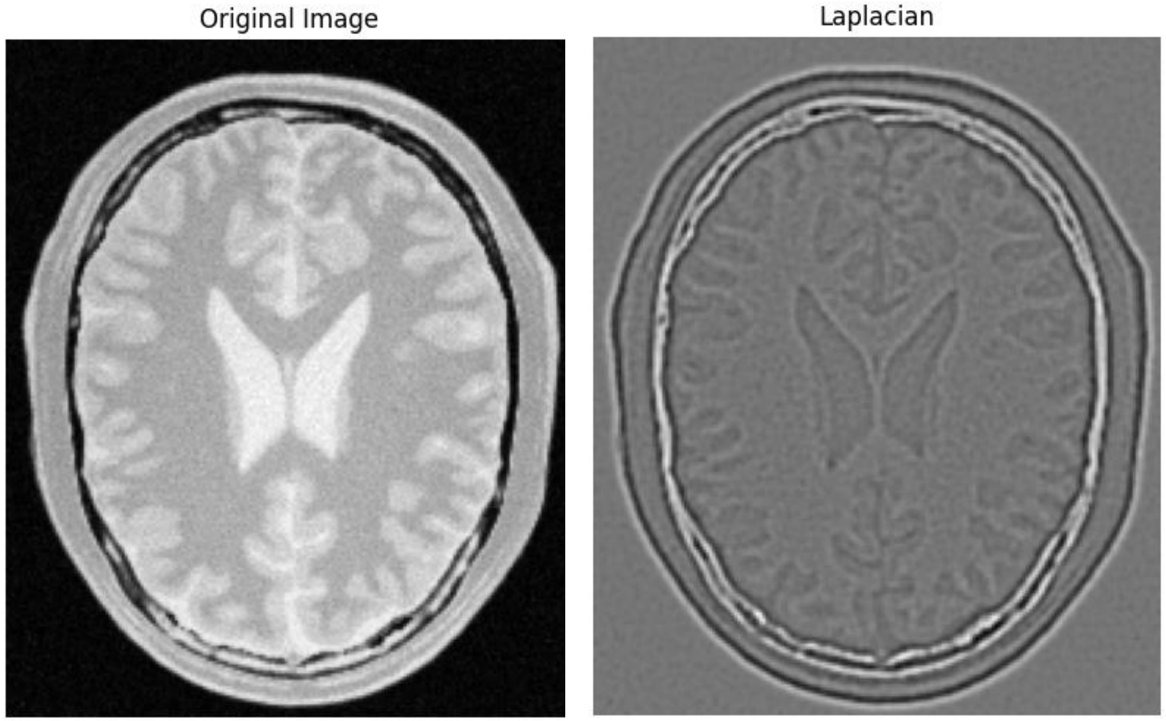


2.5.2 Laplacian Filters

The Laplacian of an image highlights regions of rapid intensity change







$$\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$$

The ITK toolkit offers two ways for getting this.

1. Using **RecursiveGaussianImageFilter** to add separate filters.
2. Using **LaplacianRecursiveGaussianImageFilter** directly.

Thank You!