4 Edge Detection

A simple idea of an edge of a 1D-function $f(x)$ is the place where it changes most rapidly, i.e. where $f'(x) \to \mathsf{max}$. This in turn leads to the idea of the second derivative being zero, i.e. $f''(x) \to 0$. However, since we numerically have to deal with sampled functions, the location of the zero of the second derivative of a function may not be exactly at one of the sample points, so we need to use the interpolation ideas that we already saw in section 3.

4.1 Differentiation of Sampled Functions

4.1.1 Differentiation Based on the Sampling Theorem

We already saw that a function may be exactly reconstructed from its samples (assuming that the function is periodic, band-limted and that sampling is above the Nyquist rate). This means that we can then differentiate the result :

$$h(t) = \sum_{n=-\infty}^{\infty} h_n \mathrm{Sinc}(\omega_c t - n\pi) \quad \Rightarrow \quad h''(t) = \sum_{n=-\infty}^{\infty} h_n \mathrm{Sinc}''(\omega_c t - n\pi) \tag{26}$$

The second derivative of a Sinc funcion is

$$\mathrm{Sinc}(t) := \frac{\sin(t)}{t} \quad \Rightarrow \quad \mathrm{Sinc}\text{''}(t) = -\frac{2\cos(t)}{t^2} + \frac{2\sin(t)}{t^3} - \frac{\sin(t)}{t} \tag{27}$$

Here is a plot of it : Note

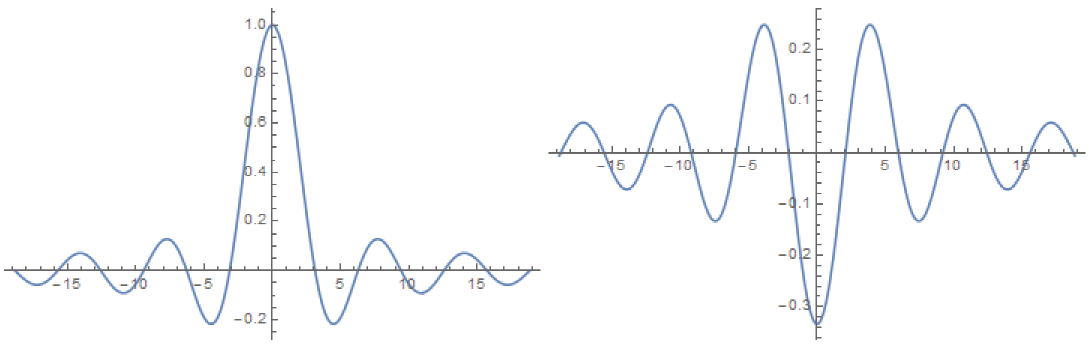


Figure 3: Left: plot of the Sinc function. Right : plot of the Sinc" function.

- The Sinc function takes the value 1 at zero and zero at multiples of π.
- The Sinc" function takes the value $-\frac{1}{3}$ at zero and $-1^{n-1}\frac{2}{\pi^2}\frac{1}{n^2}$ at $t = n\pi$. This evaluates to $\left\{-\frac{1}{3}, \frac{2}{\pi^2}, -\frac{2}{4\pi^2}, \frac{2}{9\pi^2}, -\frac{2}{16\pi^2}, \frac{2}{25\pi^2}, \frac{2}{36\pi^2}, \cdots\right\}$

Convolution of a set of samples with the sampled values of the Sinc" function give the values of the 'ideal' sampled 2nd derivative of the orginal function. As before, this gives an infinite summation which is computationally heavy.

4.1.2 Finite Difference Differentiation

A different approach to finding numerical derivatives is from Taylor Series which states how to find the value of a function $h(t)$ at a distance ϵ from a known point t_0 :

$$h(t_0+\epsilon) = h(t_0) + \epsilon \left.\frac{\partial h}{\partial t}\right|_{t_0} + \frac{\epsilon^2}{2}\left.\frac{\partial^2 h}{\partial t^2}\right|_{t_0} + \frac{\epsilon^3}{6}\left.\frac{\partial^3 h}{\partial t^3}\right|_{t_0} + \ldots + \frac{\epsilon^n}{n}\left.\frac{\partial^n h}{\partial t^n}\right|_{t_0} \tag{28}$$

Using this we can get different approximations to the second derivative

$$\frac{h(t_0+\epsilon) - 2h(t_0) + h(t_0-\epsilon)}{\epsilon^2} = \left.\frac{\partial^2 h}{\partial t^2}\right|_{t_0} + \mathcal{O}(\epsilon^2) \tag{29}$$

which has an error depending on the 4th and higher derivatives that scales with ϵ^2. In terms of the original samples, we get this approximation by convolving it with

$$\{1 \ -2 \ 1\}$$

To get a higher order approximation, we consider the expression

$$Ah(t_0) + B[h(t_0+\epsilon) + h(t_0-\epsilon)] + C[h(t_0+2\epsilon) + h(t_0-2\epsilon)]$$

and arrange for the $4th$ derivative to vanish. We evaluate the above to

$$Ah(t_0) + B\left[2h(t_0) + \epsilon^2\left.\frac{\partial^2 h}{\partial t^2}\right|_{t_0} + \frac{\epsilon^4}{12}\left.\frac{\partial^4 h}{\partial t^4}\right|_{t_0} + \ldots\right] + C\left[2h(t_0) + 4\epsilon^2\left.\frac{\partial^2 h}{\partial t^2}\right|_{t_0} + \frac{16\epsilon^4}{12}\left.\frac{\partial^4 h}{\partial t^4}\right|_{t_0} + \ldots\right]$$

This leads to $16C + B = 0$ and $A + 2B + 2C = 0$ which gives $A = \frac{30}{12}, B = -\frac{16}{12}, C = \frac{1}{12}$. So we have convolution with

$$\frac{1}{12}\{-1 \ 16 \ -30 \ 16 \ -1\}$$

This approximation has an error depending on the 6th and higher derivatives that scales with ϵ^4. Clearly we can continue this approach, adding samples at greater distance from the point of evaluation, and eliminating higher and higher order derivatives.

4.1.3 Fourier Domain Differentiation

As we have seen that convolution is multiplication in the Fourier Domain, we can exploit the differentiation properties of the Fourier Transform

$$\mathcal{F}[h"(t)] = -\omega^2 \mathcal{F}[h(t)]$$

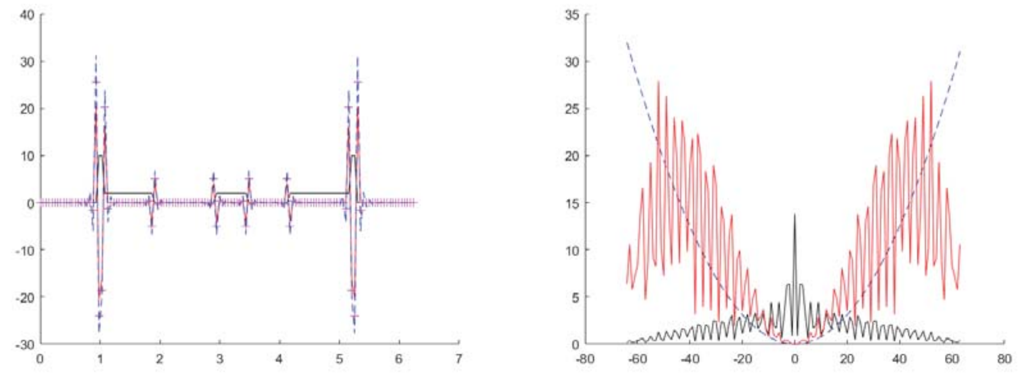
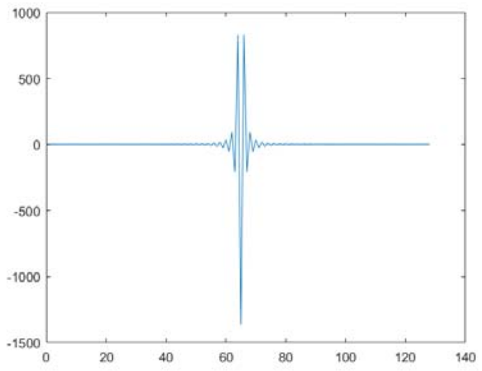


Figure 4: Left: An example function (black solid line) and its derivative (blue dashed line), approximation using nearest neighbours (red) and 2nd nearest neighbours (magenta). Right : The Fourier transform of the function (black), the derivative filter $-\omega^2$ and the result of multiplication (red).

We simply need to multiply the discrete Fourier Tranform of the samples with the square of the freqency sample values :

$$-\left\{(N/2\omega_0)^2, ((1-N/2)\omega_0)^2, \ldots, \omega_0^2, 0, \omega_0^2, \ldots, ((1-N/2)\omega_0)^2\right\} \tag{30}$$

This allows us to see the 'ideal' spatial convolution simply by inverse Fourier transform of the list Eq. 30. This is displayed here



compare this to figure 3.

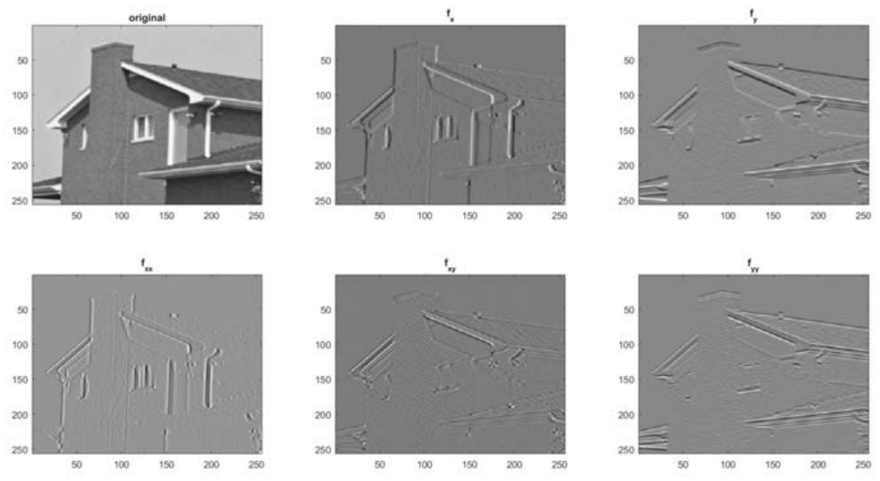


Figure 5: An image and its two first order and three second order derivaties.

4.2 Differentiation of 2D Images

When dealing with two or more dimensions we have to use multiple derivatives. There are two first order derivatives and three 2nd order derivatives in 2D

$$f_x := \frac{\partial f}{\partial x}\,, \quad f_y := \frac{\partial f}{\partial y}\,, \quad f_{xx} := \frac{\partial^2 f}{\partial x^2}\,, \quad f_{xy} := \frac{\partial^2 f}{\partial x \partial y}\,, \quad f_{yy} := \frac{\partial^2 f}{\partial y^2}$$

Note that the *gradient* operator is defined $\nabla f := \begin{pmatrix} f_x \\ f_y \end{pmatrix}$ and the *Hessian* as $\mathsf{H}(f) := \begin{pmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{pmatrix}$.

How should we define the location of an edge ? One way is to take the magnitude $g = |\nabla f| = \sqrt{f_x^2 + f_y^2}$. This has the disadvantage it has to be thresholded and searched to find the local maximum. What is the equivalent of the zero-crossing of the 2nd derivative, given that there are three of the them ? One answer is to look for the zero-crossings of the 2nd derivative *in the direction of the gradient*. First we need to define what a directional derivative is. We can get it from Taylor's series.

Define a position $\mathbf{r} := \begin{pmatrix} x \\ y \end{pmatrix}$, and consider an arbitrary length one direction vector $\hat{\mathbf{v}} := \begin{pmatrix} v_x \\ v_y \end{pmatrix} = \begin{pmatrix} \cos\theta \\ \sin\theta \end{pmatrix}$. From the fundamental theorm of calculus we can state

$$\frac{\partial f}{\partial \mathbf{v}} := \lim_{\epsilon \to 0} \left[\frac{f(\mathbf{r} + \epsilon \hat{\mathbf{v}}) - \mathbf{f}(\mathbf{r})}{\epsilon} \right] = \lim_{\epsilon \to 0} \left[\frac{f(\mathbf{r}) + \epsilon \hat{\mathbf{v}} \cdot \nabla f + \mathcal{O}(\epsilon^2) - \mathbf{f}(\mathbf{r})}{\epsilon} \right] = \hat{\mathbf{v}} \cdot \nabla f = \cos\theta f_x + \sin\theta f_y \tag{31}$$

The direction $\hat{\mathbf{n}} := \frac{\nabla f}{|\nabla f|}$ is a unit vector normal to the level-sets of f and points in the *direction* *of steepest descent* of f. From Eq. 31 we define the derivative in the direction of steepest descent as

$$\frac{\partial}{\partial \mathbf{n}} := \left(\cos\theta \frac{\partial}{\partial x} + \sin\theta \frac{\partial}{\partial y}\right) \tag{32}$$

and therefore the 2nd derivative in the direction of steepest descent as

$$\frac{\partial^2}{\partial \mathbf{n}^2} := \left(\cos\theta \frac{\partial}{\partial x} + \sin\theta \frac{\partial}{\partial y}\right)\left(\cos\theta \frac{\partial}{\partial x} + \sin\theta \frac{\partial}{\partial y}\right) = \left(\cos^2\theta \frac{\partial^2}{\partial x^2} + 2\cos\theta\sin\theta \frac{\partial^2}{\partial x \partial y} + \sin^2\theta \frac{\partial^2}{\partial y^2}\right) . \tag{33}$$

Applying this to f itself gives

$$f_{nn} := \frac{\partial^2 f}{\partial \mathbf{n}^2} = \frac{f_x^2 f_{xx} + 2 f_x f_y f_{xy} + f_y^2 f_{yy}}{f_x^2 + f_y^2} \tag{34}$$

The zero crossings of f_{nn} give the location of the edges and is known as the *Canny Edge Detector*. A technical difficulty with this function is that the denominator could become zero. Therefore sometimes we use the *Unnormalised* Canny Edge Detector

$$f_{nn}^{\mathrm{Un}} := |\nabla f|^2 f_{nn} = f_x^2 f_{xx} + 2 f_x f_y f_{xy} + f_y^2 f_{yy} = \begin{pmatrix} f_x & f_y \end{pmatrix} \begin{pmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{pmatrix} \begin{pmatrix} f_x \\ f_y \end{pmatrix} = \nabla f^{\mathsf{T}} \mathsf{H}(f) \nabla f . \tag{35}$$

Examples of some of these quantities are show in figure 6.

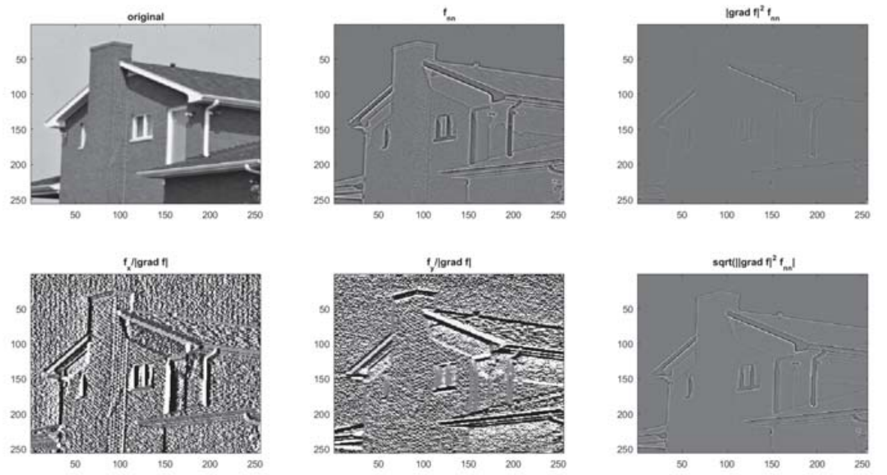

original
f_{nn}
$|\text{grad } f|^2 f_{nn}$
$f_x/|\text{grad } f|$
$f_y/|\text{grad } f|$
$\text{sqrt}(||\text{grad } f|^2 f_{nn}|)$

Figure 6: An image and the result of applying the Canny Edge Detector Eq. 34 . Also shown are the Unnormalised Canny Edge Detector Eq. 35 and its square root, and the images of $\cos\theta = \frac{f_x}{\sqrt{f_x^2+f_y^2}}$ and $\sin\theta = \frac{f_y}{\sqrt{f_x^2+f_y^2}}$.

4.3 Differentiation of 3D Images

In 3D we have a very similar analysis. There are three first order derivatives,

$$f_x := \frac{\partial f}{\partial x}\,, \quad f_y := \frac{\partial f}{\partial y} \quad f_z := \frac{\partial f}{\partial z}\,,$$

and six second order derivatives

$$f_{xx} := \frac{\partial^2 f}{\partial x^2}\,, \quad f_{yy} := \frac{\partial^2 f}{\partial y^2}\,, \quad f_{zz} := \frac{\partial^2 f}{\partial z^2}\,, \quad f_{xy} := \frac{\partial^2 f}{\partial x \partial y}\,, \quad f_{xz} := \frac{\partial^2 f}{\partial x \partial z}\,, \quad f_{yz} := \frac{\partial^2 f}{\partial y \partial z}\,,$$

and we have the gradient and Hessian defined as

$$\nabla^{\mathsf{3D}} f = \begin{pmatrix} f_x \\ f_y \\ f_z \end{pmatrix} \quad ; \quad \mathsf{H}^{\mathsf{3D}}(f) := \begin{pmatrix} f_{xx} & f_{xy} & f_{xz} \\ f_{xy} & f_{yy} & f_{yz} \\ f_{xz} & f_{yz} & f_{zz} \end{pmatrix} .$$

Using spherical polar notation we have

$$\frac{\partial}{\partial \mathbf{n}^{\mathsf{3D}}} := \left(\cos\theta \sin\phi \frac{\partial}{\partial x} + \sin\theta \sin\phi \frac{\partial}{\partial y} + \cos\phi \frac{\partial}{\partial z} \right) \tag{36}$$

and the Canny operator is

$$f^{\mathsf{3D}}_{nn} := \frac{\partial^2 f}{\partial \mathbf{n}^{\mathsf{3D},2}} = \frac{f_x^2 f_{xx} + f_y^2 f_{yy} + f_z^2 f_{zz} + 2 f_x f_y f_{xy} + 2 f_x f_z f_{xz} + 2 f_y f_z f_{yz}}{f_x^2 + f_y^2 + f_z^2} = \frac{\nabla^{\mathsf{3D}} f^{\mathsf{T}} \mathsf{H}^{\mathsf{3D}}(f) \nabla^{\mathsf{3D}} f}{|\nabla^{\mathsf{3D}} f|^2} \tag{37}$$

See the code `Canny3D` on the course webpage for examples.

4.4 Finding a Continuous Connected Level Set in an Image

The Canny edge operator (2D or 3D) gives a response at every sample point (i.e. at each pixel/voxel). These values are very unlikely to be identically zero. Instead we have to interpolate to find *zero-crossings.* There is a well-established algorithm for this in 2D "Marching Squares", or "Marching Cubes" in 3D. We'll briefly sketch the idea in 2D. For 3D see the paper W.E.Lorenson and H.E.Cline, "Marching cubes: A high resolution 3D surface construction algorithm", *In* : Proceedings of the 14th annual conference on Computer graphics and interactive techniques, Pages 163-169 (1987).

"Marching Squares" works by taking the output of the 2D Canny operator and considering groups of 2×2 pixels in turn. Each of these is either positve or negative, so can be labelled as one of 16 binary patterns. These patterns give rise to different outputs from Marching Squares

Case	Number	Action
$\begin{array}{cc}\hline 0 & 0 \\ \hline 0 & 0 \\ \hline\end{array}$ or $\begin{array}{cc}\hline 1 & 1 \\ \hline 1 & 1 \\ \hline\end{array}$	(two cases)	No zero crossings, no output
$\begin{array}{cc}\hline 1 & 0 \\ \hline 0 & 0 \\ \hline\end{array}$ or $\begin{array}{cc}\hline 0 & 1 \\ \hline 1 & 1 \\ \hline\end{array}$, etc.	(eight cases)	Two zero crossings, output two interpolated points, and a connecting line segment
$\begin{array}{cc}\hline 1 & 1 \\ \hline 0 & 0 \\ \hline\end{array}$ or $\begin{array}{cc}\hline 0 & 1 \\ \hline 0 & 1 \\ \hline\end{array}$, etc.	(four cases)	Two zero crossings, output two interpolated points, and a connecting line segment
$\begin{array}{cc}\hline 0 & 1 \\ \hline 1 & 0 \\ \hline\end{array}$ or $\begin{array}{cc}\hline 1 & 0 \\ \hline 0 & 1 \\ \hline\end{array}$	(two cases)	Four zero crossings, output four interpolated points choose between two possible pairs of line segments

This template is scanned across the image in raster fashion (it is a *sequential algorithm*) with the patterns overlapping. Thus the sets of output line segments are connected and the result is a finite number of *closed connected polygons* representing a piecewise linear representation of the zero levels sets of the input image. Note that the algorithm can be used to find any level set for a value T in an image simply by subtracting T from an image and using that as input.