Computer Vision

Xin Wang

January 26, 2022

Contents

Chapter 1

Image Formation

A digital image is formed from three components:

1. Lighting
2. Reflectance
3. Optics and Sensors

1.1 Lighting

Lighting has several properties of concern:

- Location
- Intensity of light
- Spectrum of light

There are many different sources of light. Four commonly used models to describe light sources are:

- **Point Light Source**: The light is inside the scene at a specific location only and it shines light equally in all directions. An example is be a table lamp.
- **Area Light Source**: The light source comes from a rectangular area and projects light from one side of the rectangle. An example is a florescent light fixture in a ceiling panel.
- **Sun Light Source**: The light is outside the scene and far enough away that all rays of light are basically from the same direction. An example is the sun in an outdoor scene.
- **Spotlight Light Source**: The light is focused and forms a cone-shaped envelop as it projects out from the light source. An example is a spotlight in a theatre.

1.2 Reflectance

A general model for modelling reflectance is the **Bidirectional Reflectance Distribution Function (BRDF)**. The model describes how much light arriving at incident direction is emitted in reflected direction.

$$f_r\left(\theta_i, \sigma_i, \theta_r, \sigma_r, \lambda\right) = \frac{dL_r}{dE_i}$$

where:

- θ_i and σ_i: Incident direction
- θ_r and σ_r: Reflected direction
- λ: Wavelength
- dL_r: Output power
- dE_i: Input power

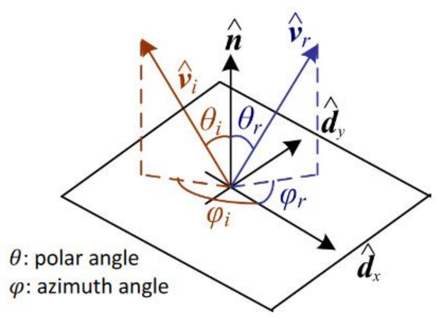
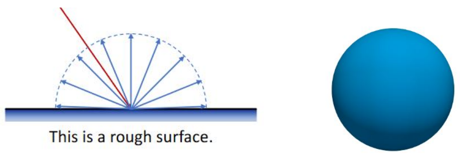


Figure 1.1: BRDF model

Considering the two ideal cases of reflection: **Diffuse reflection** and **Specular reflection**

- **Diffuse reflection**:

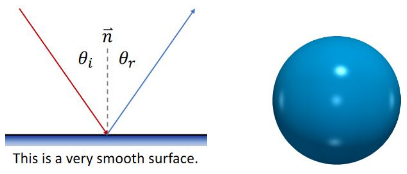


The light is scattered uniformly in all directions so the BRDF is constant:

$$f_r\left(\theta_i, \sigma_i, \theta_r, \sigma_r, \lambda\right) = f_r(\lambda)$$

This effect on the viewed image is similar to the appearance of a rough surface.

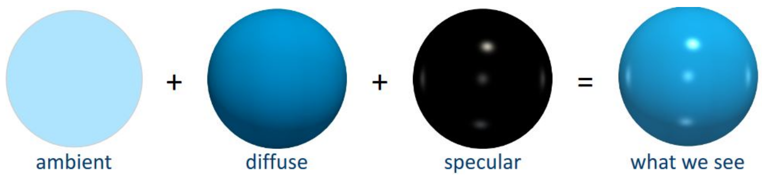
- **Specular reflection**:



The light is reflected in a mirror-like fashion. The reflection and incident directions are symmetric with respect to the surface normal $\vec{n}$: $\theta_r = \theta_i$

For the majority of cases in the real world, there is a combination of **diffuse reflection**, **specular reflection** and **ambient illumination**. Ambient illumination accounts for general illumination which may be complicated to model such as the inter-reflection between walls in a room and distant light sources as seen in sunny outdoor environments.

This observation is formally state in the **Phong Reflection Model** which is an empirical model in computer graphics that describes how a surface reflects light as a combination of ambient, diffuse and specular components.



1.2.1 Computer graphics

There is an unique relationship between computer graphics and computer vision similar to the relationship of writing and reading. Computer graphics is concerned with generating images while computer vision is concerned with intepreting images.

This relationship is particularly valuable as computer graphics in photorealistic games can be used to generate data i.e. images + labels for training computer vision algorithms.

1.3 Optics

Camera sensors imitates the human eye which are human sensors. In the human eye, there are two types of neural cells in the retina:

- **Cone cells**: Colour vision and functions in the bright light.
- **Rod cells**: More sensitive to the light but monochromatic and functions in the dim light like night time.

Camera sensors are very much the same: **Charge-coupled Device (CCD)** and **Complementary Metal-oxide Semiconductor (CMOS)**.

1.3.1 Colour Filter Arrays (CFA)

A color filter array (CFA) or color filter mosaic (CFM) is a mosaic of tiny color filters placed over the pixel sensors of an image sensor to capture color information.

The most common CFA is the most single-chip digital image sensors used in digital cameras to create a colour image is the *Bayer Filter Mosaic*. The Bayer Filter Mosaic is arraged to mimic the human eyes i.e. most sensitive to green light with 50% green, 25% red and 25% blue.

The RGB of different cameras may be different, i.e. with different sensitivities to wavelengths. This different colour sensitivity is why the same picture with different cameras would look different.

1.3.2 Bayer Colour Filter

With its arrangement, only one colour is available at each pixel. The other two colours can be interpolated from neighbouring pixels. Through this interpolation at each pixel, the RGB values can be obtained.

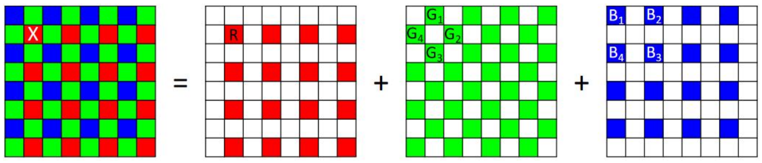
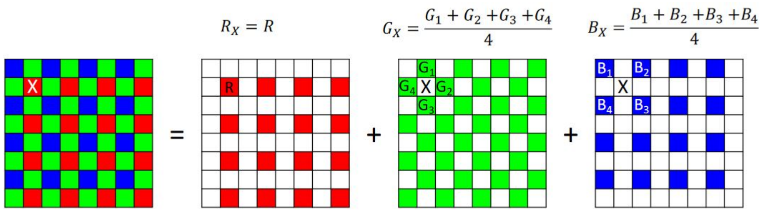


Figure 1.2: Bayer colour filter and interpolation

1.3.3 Demosaicing

A demosaicing algorithm is a digital image process used to reconstruct a full colour image from incomplete colour samples output from an image sensor overlaid with a CFA.

A simple method is bilinear interpolation where the red value of a non-red pixel is computed as the average of the two or four adjacent red pixels, and similarly for blue and green.


$R_X = R$
$G_X = \frac{G_1 + G_2 + G_3 + G_4}{4}$
$B_X = \frac{B_1 + B_2 + B_3 + B_4}{4}$
X
=
R
+
G_1
G_4
X
G_2
G_3
+
B_1
B_2
X
B_4
B_3

1.3.4 Colour spaces

Colour spaces are mathematical models describing the way colours can be represented. Colour spaces can be seen as a box containing all possible colours that can be produced by mixing primary colours of RGB.

The CIE 1931 XYZ was created by the International Commission on Illumination and represents the colour space using the primary colours X, Y and Z.

$$x = \frac{X}{X+Y+Z}$$
$$y = \frac{Y}{X+Y+Z}$$
$$z = \frac{Z}{X+Y+Z}$$

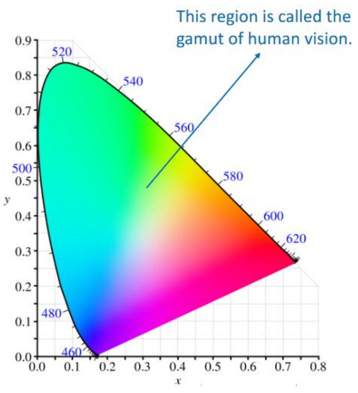


Figure 1.3: CIE 1931

There are various different forms colour spaces such as sRGB, HSV and CMYK. All these colour spaces can represent the same colour, but using different primary colours or coordinate systems.

1.3.5 Quantisation

Quantisation is the process that maps continuous signal to discrete signal. For colours, color quantization reduces the number of colors used in an image. This is important for displaying images on devices that support a limited number of colors and for efficiently compressing certain kinds of images.

It is important to note that numerical errors occur during quantisation, which depends on the number of bits used. The more bits, the less quantisation error.

Chapter 2

Image Filtering

The goal of using filters is to modify or enhance image properties and/or to extract valuable information from the pictures such as edges, corners, and blobs.

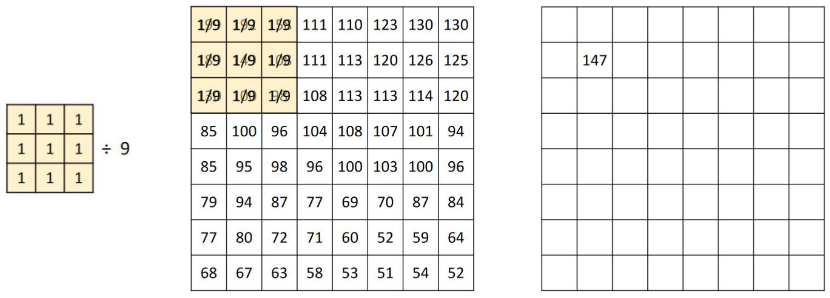
There are various types of image filters such as:

- Identity filter
- Low-pass/Smoothing filters: Moving average filter or Gaussian filter
- High-pass/Sharpening filters
- Denoising filter: Median filter

2.1 Smoothing Filters: Moving Average Filter

A moving average filter moves a window across the signal and calculates the average value within the window. A **filter kernal** is specified to indicate what values are averaged in the moving average filter. A filter kernal [1] is a small matrix used for blurring, sharpening, edge detection etc.

In images, the moving average filter removes high frequency signal e.g. noise or sharpness. This operation results in a smooth but blurry image. For example, consider a **box blur** kernal and its effect:



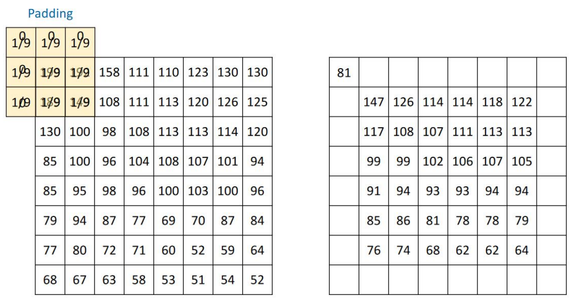
1	1	1
1	1	1
1	1	1

÷ 9

1/9	1/9	1/9	111	110	123	130	130
1/9	1/9	1/9	111	113	120	126	125
1/9	1/9	1/9	108	113	113	114	120
85	100	96	104	108	107	101	94
85	95	98	96	100	103	100	96
79	94	87	77	69	70	87	84
77	80	72	71	60	52	59	64
68	67	63	58	53	51	54	52

	147						

Due to the nature of this method of calculations, the output image will be smaller than the input image. The boundary pixels are generally dealt with using padding of various methods such as constant value, mirroring values etc. The following example uses 0 padding for its boundary pixels.



Padding

1/9	1/9	1/9						
1/9	1/9	1/9	158	111	110	123	130	130
1/9	1/9	1/9	108	111	113	120	126	125
	130	100	98	108	113	113	114	120
	85	100	96	104	108	107	101	94
	85	95	98	96	100	103	100	96
	79	94	87	77	69	70	87	84
	77	80	72	71	60	52	59	64
	68	67	63	58	53	51	54	52

81							
	147	126	114	114	118	122	
	117	108	107	111	113	113	
	99	99	102	106	107	105	
	91	94	93	93	94	94	
	85	86	81	78	78	79	
	76	74	68	62	62	64	

By increasing the size of the kernal e.g. into a 7×7 matrix, the image will become blurrier.

[1]https://en.wikipedia.org/wiki/Kernel_(image_processing)

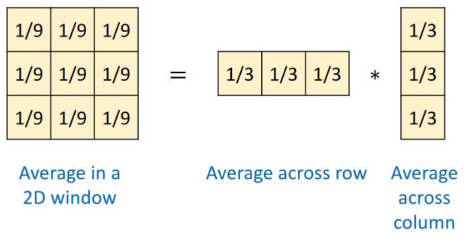
2.1.1 Brute Computational Complexity

Note that: *Image size*: $N \times N$ where N is the number of pixels and *Kernal size*: $K \times K$ where K is size of the filter kernal matrix

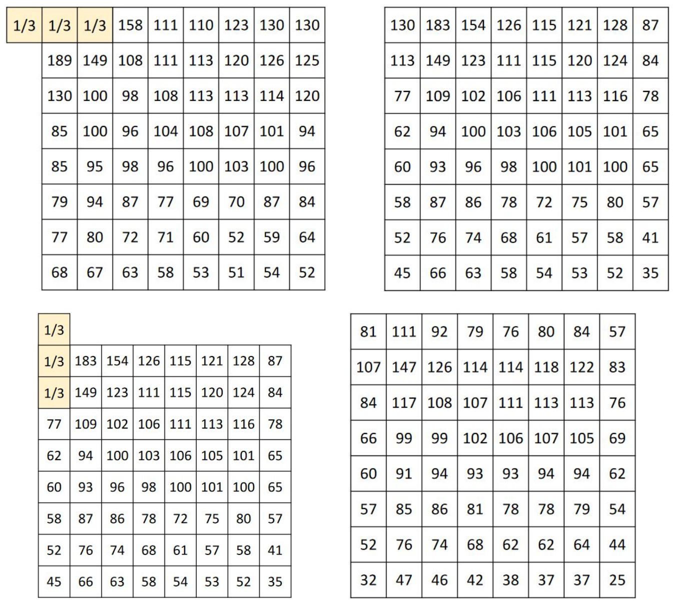
- There are N^2 pixels
- At each pixel, there are K^2 multiplications and $K^2 - 1$ summations.
- In total, there are: N^2K^2 multiplications and $N^2(K^2 - 1)$ summations
- Complexity is: $O(N^2K^2)$

2.1.2 Separable filter

If a big filter can be separated as the consecutive operation of two small filters, then the first filter can be applied to the the input image then the second filter. For example, consider the previous blur filter kernal divided into two smaller filters:



This calculation procedure results in the same result as the previous result:



1/3	1/3	1/3	158	111	110	123	130	130
	189	149	108	111	113	120	126	125
	130	100	98	108	113	113	114	120
	85	100	96	104	108	107	101	94
	85	95	98	96	100	103	100	96
	79	94	87	77	69	70	87	84
	77	80	72	71	60	52	59	64
	68	67	63	58	53	51	54	52

130	183	154	126	115	121	128	87
113	149	123	111	115	120	124	84
77	109	102	106	111	113	116	78
62	94	100	103	106	105	101	65
60	93	96	98	100	101	100	65
58	87	86	78	72	75	80	57
52	76	74	68	61	57	58	41
45	66	63	58	54	53	52	35

1/3							
1/3	183	154	126	115	121	128	87
1/3	149	123	111	115	120	124	84
77	109	102	106	111	113	116	78
62	94	100	103	106	105	101	65
60	93	96	98	100	101	100	65
58	87	86	78	72	75	80	57
52	76	74	68	61	57	58	41
45	66	63	58	54	53	52	35

81	111	92	79	76	80	84	57
107	147	126	114	114	118	122	83
84	117	108	107	111	113	113	76
66	99	99	102	106	107	105	69
60	91	94	93	93	94	94	62
57	85	86	81	78	78	79	54
52	76	74	68	62	62	64	44
32	47	46	42	38	37	37	25

2.1.3 Separable filter complexity

Note that: *Image size*: $N \times N$ where N is the number of pixels and there are two filter kernals: $1 \times K$ and $K \times 1$.

- There are N^2 pixels
- At each pixel, there are K multiplications and $K - 1$ summations.
- In total, there are: $2N^2K$ multiplications and $2N^2(K - 1)$ summations
- Complexity is: $O(N^2K)$ which is faster than the previous $O(N^2K^2)$

2.2 Identity Filter

The **Identity Filter Kernal** simply returns the same value of the image i.e. the input and output image is the same.

2.3 Smoothing Filters: Gaussian Filter

The Gaussian Filter uses a 2D Gaussian Distribution as its filter kernal:

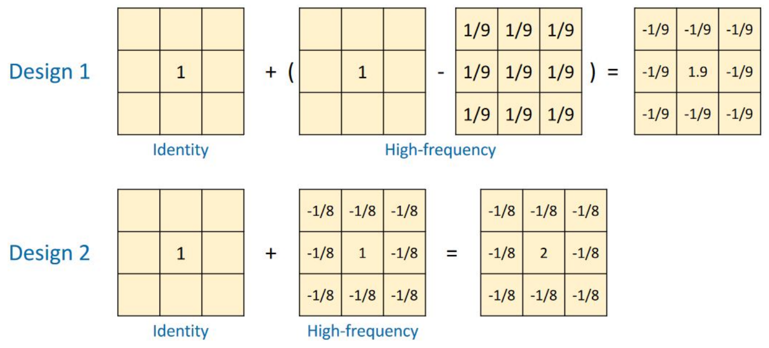
$$h(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

The 2D Gaussian filter is a separable filter, equivalent to two 1D Gaussian filters with the same σ, one along x-axis and the other along y-axis:

$$h(i,j) = h_x(i) * h_y(j)$$
$$h_x(i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{i^2}{2\sigma^2}}$$

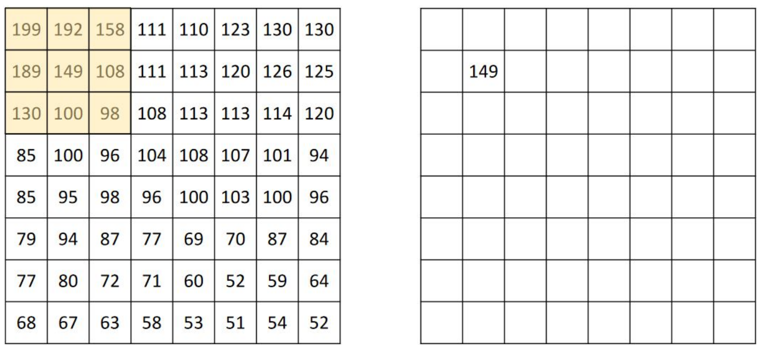
2.4 High-pass Filters

There are various methods of designing high-pass filters including using low-pass filters as seen in Design 1.



2.5 Denoising Filters: Median Filter

Median filters are non-linear filters that is often used for denoising an image. A common method of performing median filter is to move the sliding window and replacing the centre pixel using the *median value* in the window.



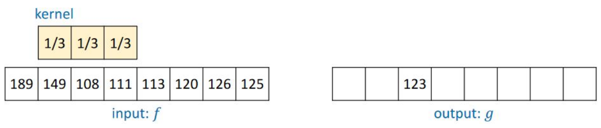
199	192	158	111	110	123	130	130
189	149	108	111	113	120	126	125
130	100	98	108	113	113	114	120
85	100	96	104	108	107	101	94
85	95	98	96	100	103	100	96
79	94	87	77	69	70	87	84
77	80	72	71	60	52	59	64
68	67	63	58	53	51	54	52

	149						

Chapter 3

Image Filtering II

The concept of filtering can be described mathematically for example: where this operations can be described



as:

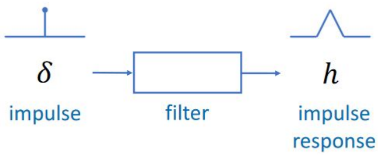
$$g[n] = \frac{1}{3}.f[n-1] + \frac{1}{3}.f[n] + \frac{1}{3}.f[n+1]$$

This concept of describing filters mathematically is extensively used in Signal Processing where it describes filters as:

A device/process 'h' that removes unwanted components/features from a input signal 'f' and generates an output signal 'g'

Impulse Response

Impulse responses are used to mathematically describe a filter. The impulse response h is the output of a filter when the input is an impulse signal δ.



- **Continuous signal**: An impulse signal is the *Dirac Delta function* $\delta(x)$:

$$\delta(x) = \begin{cases} +\infty & \text{if } x = 0 \\ 0 & \text{otherwise} \end{cases}$$

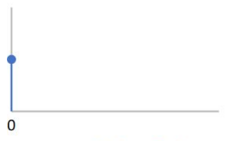
$$\int^{\infty} \delta(x)dx = 1$$

- **Discrete signal**: An impulse signal is the *Kronecker function* $\delta[i]$:

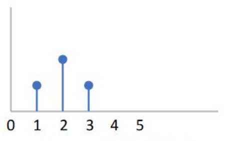
$$\delta[i] = \begin{cases} 1 & \text{if } i = 0 \\ 0 & \text{otherwise} \end{cases}$$

The impulse response h completely characterises a *linear time-invariant* filter. Because any input signal can be form using impulses, if one knows how the system may response to one impulse then you know how the system will response to many impulses.

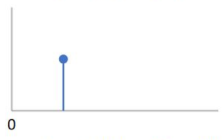
Time-invariant system: If a filter is a *time-invariant system*, when the input is shifted by time step k, the output will also shift by k. In the following example, note how the impulse response for $f[n]$ and $f[n-2]$ are the same but only shifted by 2 in a LTI system.



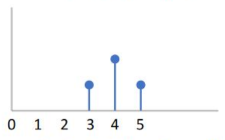
Input $f[n] = \delta[n]$



Output $g[n] = h[n]$

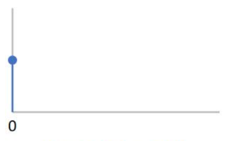


Input $f[n] = \delta[n-2]$

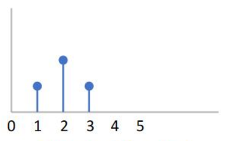


Output $g[n] = h[n-2]$

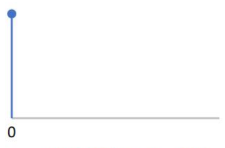
Linear system: If a filter is a *linear system*, when the input is multiplied by k, the output will also by multiplied by k.



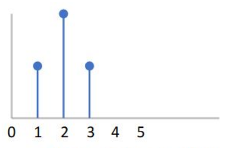
Input $f[n] = \delta[n]$



Output $g[n] = h[n]$

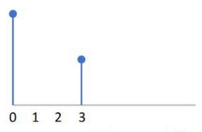


Input $f[n] = 2 \cdot \delta[n]$

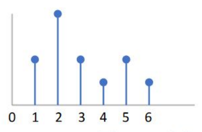


Output $g[n] = 2 \cdot h[n]$

Similarly, if a filter is a linear system, when combining two input signals linearly, their outputs will also be combined linearly.



Input $f[n] = 2 \cdot \delta[n] + \delta[n-3]$



Output $g[n] = 2 \cdot h[n] + h[n-3]$

3.1 Convolution and Linear Time-invariant System (LTI)

For a linear time-invariant system, the impulse response h completely characterises how the LTI system works. The output g can be described as the *convolution* between input f and impulse response h:

$$g[n] = f[n] * h[n] = \sum_{m=-\infty}^{\infty} f[m]h[n-m]$$

Convolutions are usually implemented by:

1. Flip the kernel
2. Multiply the signal with the flipped kernel
3. Sum over the support of the kernel

$$\sum_{m=-\infty}^{\infty} f[m]h[n-m] = \sum_{m=-\infty}^{\infty} f[n+m]h[-m]$$

3.1.1 Propoerties of convolution

- **Commutativity**: $f * h = h * f$
- **Associativity**: $f * (g * h) = (f * g) * h$
- **Distributivity**: $f * (g + h) = (f * g) + (f * h)$
- **Differentiation:** $\frac{d}{dx}(f * g) = \frac{df}{dx} * g = f * \frac{dg}{dx}$

3.1.2 2D convolution

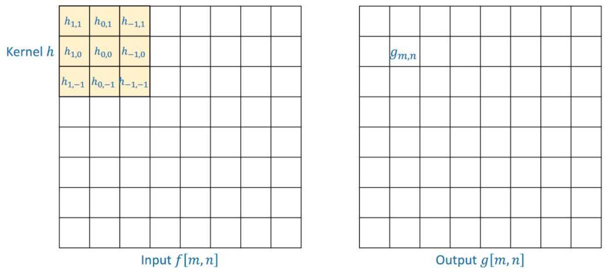
2D convolution is defined mathematically as:

$$g[m, n] = f[m, n] * h[m, n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i, j]h[m - i, n - j]$$

2D convolutions are usually implemented by:

1. Flip the kernel both horizontally and vertically
2. Multiply the image patch centred at pixel (m, n) with the flipped kernel
3. Sum over the support of the kernel

$$g[m, n] = f[m, n] * h[m, n] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[m + i, n + j]h[-i, -j]$$



$h_{1,1}$ $h_{0,1}$ $h_{-1,1}$
Kernel h
$h_{1,0}$ $h_{0,0}$ $h_{-1,0}$
$h_{1,-1}$ $h_{0,-1}$ $h_{-1,-1}$
$g_{m,n}$
Input $f[m, n]$
Output $g[m, n]$

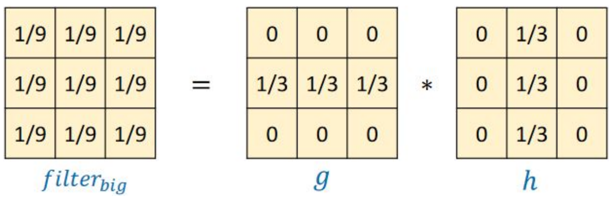
3.1.3 Asspciativity and separable filters

Note the associativity property of convolution:

$$f * (g * h) = (f * g) * h$$

If a big filter can be separated as the convolution of two small filters, such as $g * h$, then it is possible to first convolve f with g then with h.

$$f * filter_{big} = f * (g * h) = (f * g) * h$$



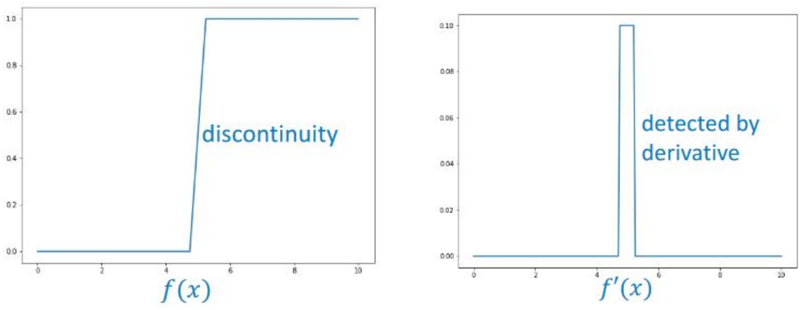
Chapter 4

Edge Detection I

In computer vision, an edge refers to lines where image brightness changes sharply and has discontinuities. Edges capture important properties of the world and are important low-level features for analysing and understanding images.

To detect edges, we note that an image can be regarded as a function of pixel position. As known from mathematics, derivatives characterises the discontinuities of a function which can be used to help detect edges. For example, consider the definition of a continuous function:

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$



(a) Continuous function

(b) Differentiated function showing detected discontinuity

For a discrete function, the finite difference can be calculated using:

- Forward difference: $f'[x] = f[x+1] - f[x]$
- Backward difference: $f'[x] = f[x] - f[x-1]$
- Central difference: $f'[x] = \frac{f[x+1]-f[x-1]}{2}$

This finite difference can be performed using convolution with kernals such as:

- $h = [1, -1, 0]$
- $h = [0, 1, -1]$
- $h = [1, 0, -1]$

4.1 Convolution

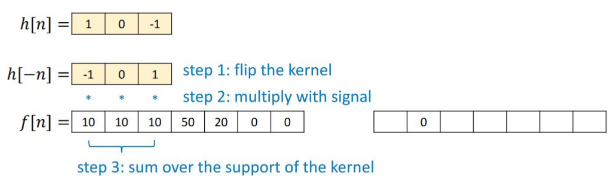
As mentioned earlier, convolution is often implemented:

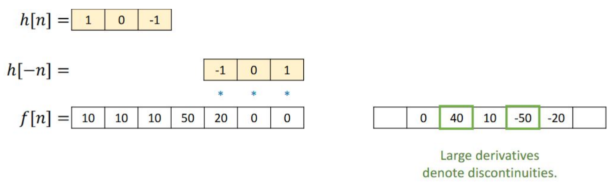
$$g[n] = f[n] * h[n] = \sum_{m=-\infty}^{\infty} f[n-m]h[m] = \sum_{m=-\infty}^{\infty} f[n+m]h[-m]$$

The *central difference*, without accounting for $\frac{1}{2}$, can be defined as:

$$\begin{aligned} f'[x] &= f[x+1] - f[x-1] \\ &= f[x+1] * 1 + f[x] * 0 + f[x-1] * (-1) \\ &= f[x+1] * h[-1] + f[x] * h[0] + f[x-1] * h[1] \end{aligned}$$

The convolution kernal is this defined as: $h = [1, 0, -1]$. The process of convolution with central difference $f'[x] = f[x] * h[x]$ can be seen as shown below:


$h[n]$ = 1 0 -1
$h[-n]$ = -1 0 1
step 1: flip the kernel
step 2: multiply with signal
$f[n]$ = 10 10 10 50 20 0 0
0
step 3: sum over the support of the kernel


$h[n]$ = 1 0 -1
$h[-n]$ = -1 0 1
$f[n]$ = 10 10 10 50 20 0 0
0 40 10 -50 -20
Large derivatives denote discontinuities.

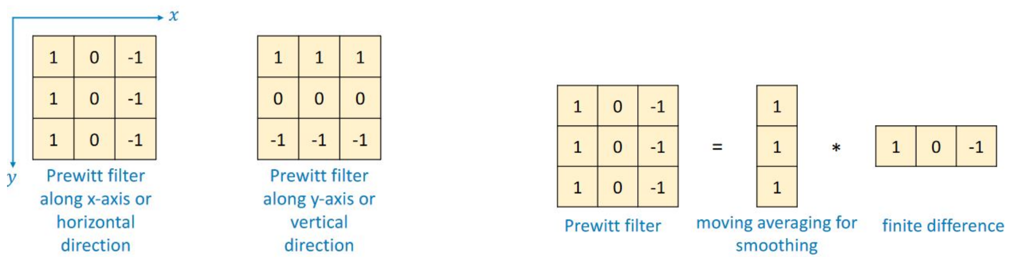
Note the following:

- The kernal $h[n]$ is flipped $h[-n]$ as mentioned earlier on how convolution implemented
- Large derivatives denote discontinuities

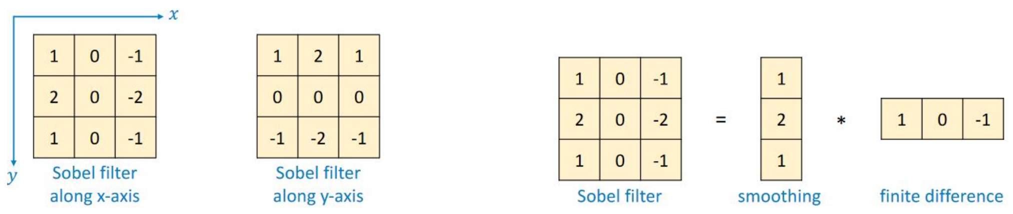
The following methods only enable discontinuities to be detected in 1D. Similar filters have to be designed to extend edge detection to a 2D image.

4.2 Edge detection filters

4.2.1 Prewitt filter


x
y
1 0 -1 / 1 0 -1 / 1 0 -1
Prewitt filter along x-axis or horizontal direction
1 1 1 / 0 0 0 / -1 -1 -1
Prewitt filter along y-axis or vertical direction
1 0 -1 / 1 0 -1 / 1 0 -1 = 1 / 1 / 1 * 1 0 -1
Prewitt filter
moving averaging for smoothing
finite difference

4.2.2 Sobel filter



4.3 Image Gradient

There are two outputs from Sobel filters that are combined to describe edges:

- describing discontinuity along x-axis
- describing discontinuity along y-axis

The two outputs are combined to describe the two properties of edges: *Magnitude* and *Orientation* using the following process:

1. Compute the derivatives along x-axis and y-axis:

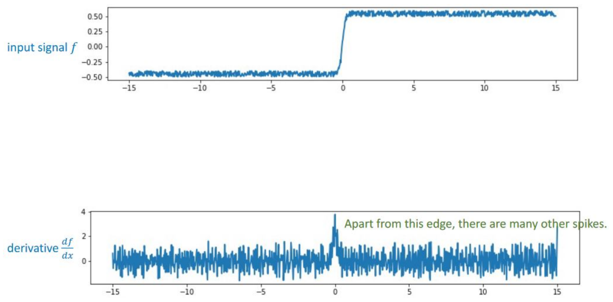
$$\begin{aligned} g_x &= f * h_x \\ g_y &= f * h_y \end{aligned}$$

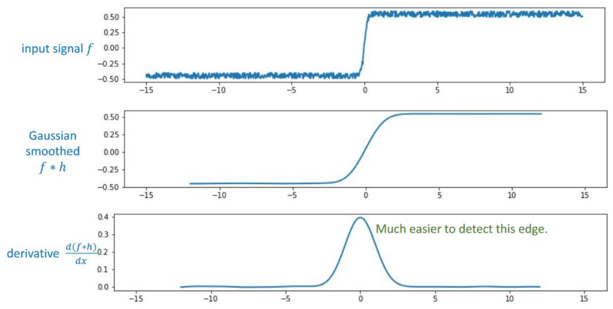
2. Compute the magnitude of the gradient:
$$g = \sqrt{g_x^2 + g_y^2}$$
3. Compute the orientation or angle of the gradient:
$$\theta = \arctan 2(g_y, g_x)$$

4.4 Smoothing

Derivatives are sensitive to noise thus using a smoothing kernel beforehand to suppress the noise would result in a better result. This can be seen in the Prewitt and Sobel filter which have a smoothing kernel built in.

An alternative is to use the Gaussian kernel for smoothing before calculating the derivatives. The following images show the ease of edge detection after smoothing compared to no smoothing performed beforehand.





4.4.1 Derivative of Gaussian Filter

Derivative of Gaussian filter is an operator that performs Gaussian smoothing before taking the derivative.

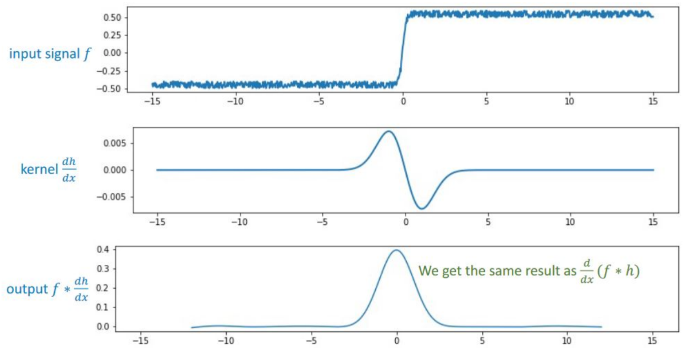
$$g[x] = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \quad \text{Gaussian kernel}$$
$$\frac{d}{dx}(f * h) \quad \text{Derivative of Gaussian filter}$$

Using the differentiation of convolution:

$$\frac{d}{dx}(f * h) = \frac{df}{dx} * h = f * \frac{dh}{dx}$$

The analytical form for the derivative of Gaussian kernel can be defined as:

$$\frac{d}{dx}(\underbrace{f}_{\text{signal}} * \underbrace{h}_{\text{Gaussian kernel}}) = f * \frac{dh}{dx} = f * \underbrace{\frac{-x}{\sqrt{2\pi}\sigma^3} e^{-\frac{x^2}{2\sigma^2}}}_{\text{derivative of Gaussian kernel}}$$



- 1D Gaussian filter: $h[x] = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$
- 2D Gaussian filter: $h[x, y] = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$

It is a separable filter and equivalent to convolution of two 1D Gaussian filters. This means that 2D Gaussian smoothing can be accelerated using separable filtering.

The process of a Gaussian Derivative Filter is defined as:

1. Smooth the input image with a 2D Gaussian filter
2. Take the derivative along x-axis and y-axis
3. Calculate the magnitude and orientation

Parameter σ in derivative of Gaussian

Large σ value suppresses noise and results smoother derivative. Different σ values finds edges at different scales.