Image Processing and Computer Vision (Module 1)

Last update: 15 March 2024

Academic Year 2023 – 2024
Alma Mater Studiorum · University of Bologna

Contents

1 Image acquisition and formation

1.1 Pinhole camera

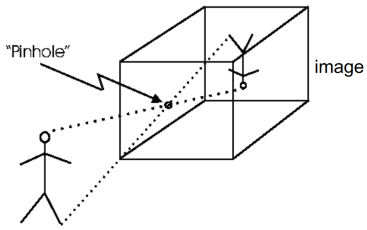
Imaging device Gathers the light reflected by 3D objects in a scene and creates a 2D representation of them. Imaging device

Computer vision Infer knowledge of the 3D scene from 2D digital images. Computer vision

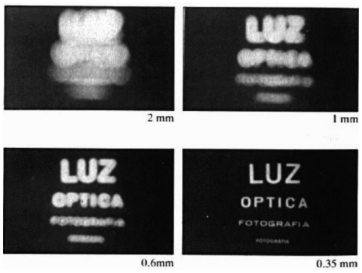
Pinhole camera Imaging device where the light passes through a small pinhole and hits the image plane. Geometrically, the image is obtained by drawing straight rays from the scene to the image plane passing through the pinhole. Pinhole camera

Remark. Larger aperture size of the pinhole results in blurry images (circle of confusion), while smaller aperture results in sharper images but requires longer exposure time (as less light passes through).

Remark. The pinhole camera is a good approximation of the geometry of the image formation mechanism of modern imaging devices.



(a) Pinhole camera model



(b) Images with varying pinhole aperture size

1.2 Perspective projection

Geometric model of a pinhole camera. Perspective projection

Scene point M (the object in the real world).

Image point m (the object in the image).

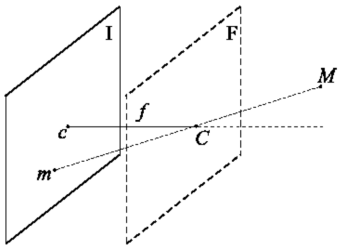
Image plane I.

Optical center C (the pinhole).

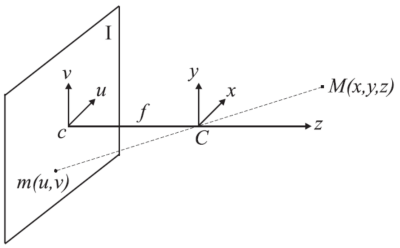
Image center/piercing point c (intersection between the optical axis – the line orthogonal to I passing through C – and I).

Focal length f.

Focal plane F.



- u and v are the horizontal and vertical axis of the image plane, respectively.
- x and y are the horizontal and vertical axis of the 3D reference system, respectively, and form the **camera reference system**.



Camera reference system

Remark. For the perspective model, the coordinate systems (U, V) and (X, Y) must be parallel.

Scene–image mapping The equations to map scene points into image points are the following:

Scene–image mapping

$$u = x\frac{f}{z} \qquad v = y\frac{f}{z}$$

Proof. This is the consequence of the triangle similarity theorems.

$$\frac{u}{x} = -\frac{f}{z} \iff u = -x\frac{f}{z}$$
$$\frac{v}{y} = -\frac{f}{z} \iff v = -y\frac{f}{z}$$

The minus is needed as the axes are inverted

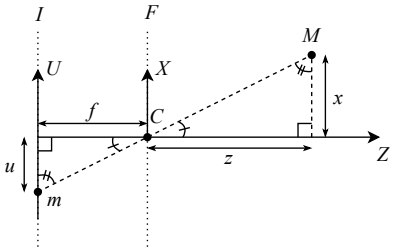


Figure 1.2: Visualization of the horizontal axis. The same holds on the vertical axis.

By inverting the axis horizontally and vertically (i.e. inverting the sign), the image plane can be adjusted to have the same orientation of the scene:

$$u = x\frac{f}{z} \qquad v = y\frac{f}{z}$$

□

Remark. The image coordinates are a scaled version of the scene coordinates. The scaling is inversely proportioned with respect to the depth.

- The farther the point, the smaller the coordinates.
- The larger the focal length, the bigger the object is in the image.

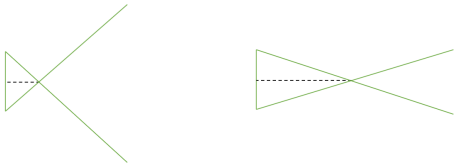


Figure 1.3: Coordinate space by varying focal length

Remark. The perspective projection mapping is not a bijection:

- A scene point is mapped into a unique image point.
- An image point is mapped onto a 3D line.

Therefore, reconstructing the 3D structure of a single image is an ill-posed problem (i.e. it has multiple solutions).

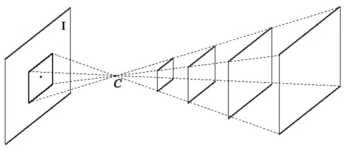


Figure 1.4: Projection from scene and image points

1.2.1 Stereo geometry

Stereo vision

Stereo vision Use multiple images to triangulate the 3D position of an object.

Stereo correspondence

Stereo correspondence Given a point L in an image, find the corresponding point R in another image.

Without any assumptions, an oracle is needed to determine the correspondences.

Standard stereo geometry

Standard stereo geometry Given two reference images, the following assumptions must hold:

- The X, Y, Z axes are parallel.
- The cameras that took the two images have the same focal length f (coplanar image planes) and the images have been taken at the same time.
- There is a horizontal translation b between the two cameras (baseline).
- The disparity d is the difference of the U coordinates of the object in the left and right image.

Fundamental relationship in stereo vision

Theorem 1.2.1 (Fundamental relationship in stereo vision)**.** If the assumptions above hold, the following equation holds:

$$z = b\frac{f}{d}$$

Proof. Let $P_L = \begin{pmatrix} x_L & y & z \end{pmatrix}$ and $P_R = \begin{pmatrix} x_R & y & z \end{pmatrix}$ be the coordinates of the object P with respect to the left and right camera reference system, respectively. Let $p_L = \begin{pmatrix} u_L & v \end{pmatrix}$ and $p_R = \begin{pmatrix} u_R & v \end{pmatrix}$ be the coordinates of the object P in the left and right image plane, respectively.

By assumption, we have that $P_L - P_R = \begin{pmatrix} b & 0 & 0 \end{pmatrix}$, where b is the baseline.

By the perspective projection equation, we have that:

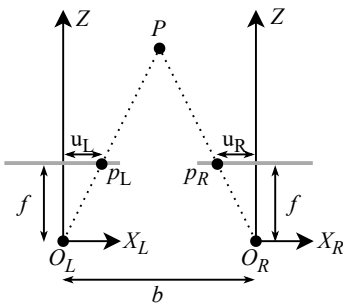
$$u_L = x_L\frac{f}{z} \qquad u_R = x_R\frac{f}{z}$$

Disparity is computed as follows:

$$d = u_L - u_R = x_L\frac{f}{z} - x_R\frac{f}{z} = b\frac{f}{z}$$

We can therefore obtain the Z coordinate of P as:

$$z = b\frac{f}{d}$$



Note: the Y/V axes are not in figure.

□

Remark. Disparity and depth are inversely proportional: the disparity of two points decreases if the points are farther in depth.

Stereo matching If the assumptions for standard stereo geometry hold, to find the object corresponding to p_L in another image, it is sufficient to search along the horizontal axis of p_L looking for the same colors or patterns. Stereo matching

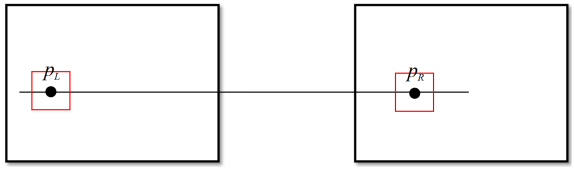


Figure 1.5: Example of stereo matching

Epipolar geometry Approach applied when the two cameras are no longer aligned according to the standard stereo geometry assumption. Still, the focal lengths and the roto-translation between the two cameras must be known. Epipolar geometry

Given two images, we can project the epipolar line related to the point p_L in the left plane onto the right plane to reduce the problem of correspondence search to a single dimension.

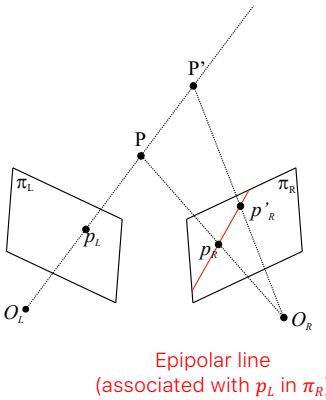
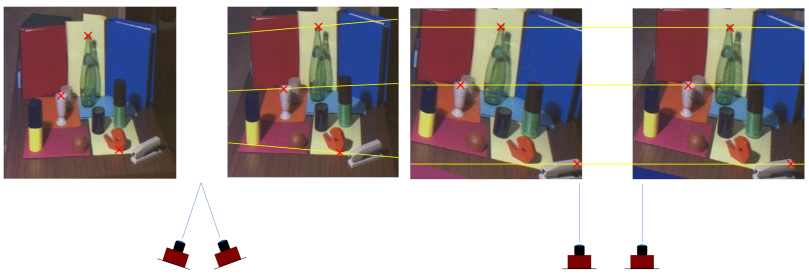


Figure 1.6: Example of epipolar geometry

Remark. It is nearly impossible to project horizontal epipolar lines and searching through oblique lines is awkward and computationally less efficient than straight lines.

Rectification Transformation applied to convert epipolar geometry to a standard stereo geometry. Rectification



(a) Images before rectification (b) Images after rectification

1.2.2 Ratios and parallelism

Given a 3D line of length L lying in a plane parallel to the image plane at distance z, then its length l in the image plane is:

$$l = L\frac{f}{z}$$

In all the other cases (i.e. when the line is not parallel to the image plane), the ratios of lengths and the parallelism of lines are not preserved.

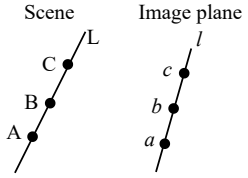


Figure 1.8: Example of not preserved ratios. It holds that $\frac{\overline{AB}}{\overline{BC}} \neq \frac{\overline{ab}}{\overline{bc}}$.

Vanishing point Intersection point of lines that are parallel in the scene but not in the image plane. Vanishing point

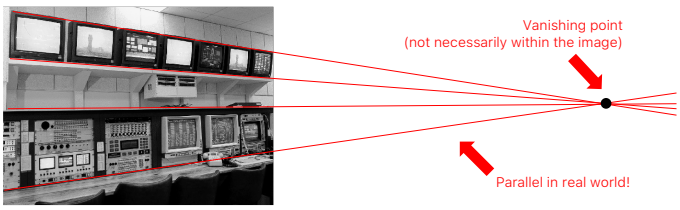


Figure 1.9: Example of vanishing point

1.3 Lens

Depth of field (DOF) Distance at which a scene point is in focus (i.e. when all its light rays gathered by the imaging device hit the image plane at the same point). Depth of field (DOF)

Remark. Because of the small size of the aperture, a pinhole camera has infinite depth of field but requires a long exposure time making it only suitable for static scenes.

Lens A lens gathers more light from the scene point and focuses it on a single image point. Lens

This allows for a smaller exposure time but limits the depth of field (i.e. only a limited range of distances in the image can be in focus at the same time).

Thin lens Approximate model for lenses. Thin lens

Scene point P (the object in the real world).

Image point p (the object in the image).

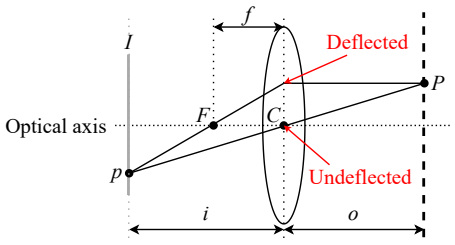
Object–lens distance o.

Image–lens distance i (i.e. focal length of the camera).

Center of the lens C.

Focal length of the lens f.

Focal plane/focus of the lens F.



A thin lens has the following properties:

- Rays hitting the lens parallel to the optical axis are deflected to pass through the focal plane of the lens F.
- Rays passing through the center of the lens C are undeflected.
- The following equation holds: Thin lens equation

$$\frac{1}{o} + \frac{1}{i} = \frac{1}{f}$$

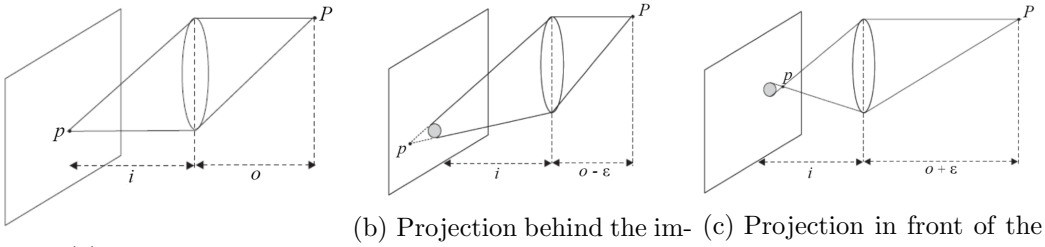
Image formation When the image is in focus, the image formation process follows the normal rules of the perspective projection model where:

- C is the optical center.
- i is the focal length of the camera.

By fixing the focal length of the lens (f), we can determine the distance of the scene point (o) or the image point (i) required to have the object in focus.

$$\frac{1}{o} + \frac{1}{i} = \frac{1}{f} \iff o = \frac{if}{i - f} \qquad \frac{1}{o} + \frac{1}{i} = \frac{1}{f} \iff i = \frac{of}{o - f}$$

Remark. Points projected in front or behind the image plane will create a circle of confusion (blur).



(a) Image in focus

(b) Projection behind the image plane

(c) Projection in front of the image plane

Adjustable diaphragm Device to control the light gathered by the effective aperture of the lens. Adjustable diaphragm

Reducing the aperture will result in less light and an increased depth of field.

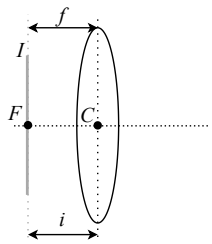
Remark. On a theoretical level, images that are not in focus appear blurred (circles of confusion). Despite that, if the circle is smaller than the photo-sensing elements (i.e. pixels), it will appear in focus.

Focusing mechanism Allows the lens to translate along the optical axis to increase its distance to the image plane. Focusing mechanism

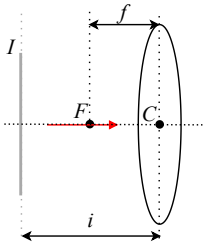
At the minimum extension (Figure 1.11a), we have that:

$$i = f \text{ and } o = \infty \text{ as the thin lens equation states that } \frac{1}{o} + \frac{1}{i} = \frac{1}{f}$$

By increasing the extension (i.e. increase i), we have that the distance to the scene point o decreases. The maximum extension determines the minimum focusing distance.



(a) Minimum extension



(b) Maximum extension

Figure 1.11: Extension of a focusing mechanism

1.4 Image digitalization

1.4.1 Sampling and quantization

The image plane of a camera converts the received irradiance into electrical signals.

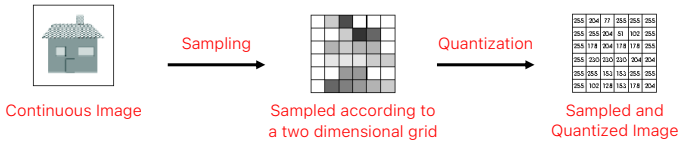


Figure 1.12: Image digitalization steps

Sampling The continuous electrical signal is sampled to produce a $N \times M$ matrix of pixels: Sampling

$$I(x,y) = \begin{pmatrix} I(0,0) & \dots & I(0,M-1) \\ \vdots & \ddots & \vdots \\ I(N-1,0) & \dots & I(N-1,M-1) \end{pmatrix}$$

Quantization Let m be the number of bits used to encode a pixel. The value of each pixel is quantized into 2^m discrete gray levels. Quantization

Remark. A grayscale image usually uses 8 bits

An RGB image usually uses $3 \cdot 8$ bits.

Remark. The more bits are used for the representation, the higher the quality of the image will be.

- Sampling with fewer bits will result in a lower resolution (aliasing).
- Quantization with fewer bits will result in less representable colors.

Figure 1.13: Sampling and quantization using fewer bits

1.4.2 Camera sensors

Photodetector Sensor that, during the exposure time, converts the light into a proportional electrical charge that will be processed by a circuit and converted into a digital or analog signal. Photodetector

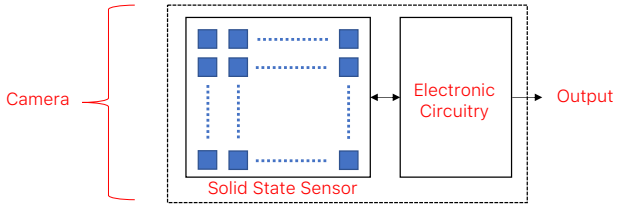


Figure 1.14: Components of a camera

The two main sensor technologies are:

Charge Coupled Device (CCD) Typically produces higher quality images but are more expensive. Charge Coupled Device (CCD)

Complementary Metal Oxide Semiconductor (CMOS) Generally produces lower quality images but is more compact and less expensive. Each sensor has integrated its own circuitry that allows to read an arbitrary window of the sensors. Complementary Metal Oxide Semiconductor (CMOS)

Color sensors CCD and CMOS sensors are sensitive to a wide spectrum of light frequencies (both visible and invisible) but are unable to sense colors as they produce a single value per pixel. Color sensors

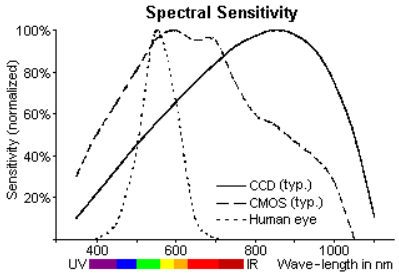


Figure 1.15: CCD and CMOS spectral sensitivity

Color Filter Array (CFA) Filter placed in front of a photodetector to allow it to detect colors. Color Filter Array (CFA)

Possible approaches are:

Bayer CFA A grid of green, blue, and red filters with the greens being twice as much as the others (the human eye is more sensible to the green range). To determine the RGB value of each pixel, missing color channels are sampled from neighboring pixels (demosaicking).

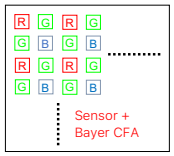


Figure 1.16: Example of Bayer filter

Optical prism A prism splits the incoming light into 3 RGB beams, each directed to a different sensor. It is more expensive than Bayer CFA.

1.4.3 Metrics

Signal to Noise Ratio (SNR) Quantifies the strength of the actual signal with respect to unwanted noise.

Signal to Noise Ratio (SNR)

Sources of noise are:

Photon shot noise Number of photons captured during exposure time.

Elecronic circuitry noise Generated by the electronics that read the sensors.

Quantization noise Caused by the digitalization of the image (ADC conversion).

Dark current noise Random charge caused by thermal excitement.

SNR is usually expressed in decibels or bits:

$$\texttt{SNR}_{\text{db}} = 20 \cdot \log_{10}(\texttt{SNR}) \qquad \texttt{SNR}_{\text{bit}} = \log_2(\texttt{SNR})$$

Dynamic Range (DR) Measures the ability of a sensor to capture both the dark and bright structure of the scene.

Dynamic Range (DR)

Let:

- E_{min} be the minimum detectable irradiation. This value depends on the noise.
- E_{max} be the saturation irradiation (i.e. the maximum amount of light that fills the capacity of the photodetector).

DR is defined as:

$$\texttt{DR} = \frac{E_{\text{max}}}{E_{\text{mix}}}$$

As with SNR, DR can be expressed in decibels or bits.

2 Spatial filtering

2.1 Noise

The noise added to a pixel p is defined by $n_k(p)$, where k indicates the time step (i.e. noise changes depending on the moment the image is taken). It is assumed that $n_k(p)$ is i.i.d and $n_k(p) \sim \mathcal{N}(0, \sigma)$.
The information of a pixel p is therefore defined as:

$$I_k(p) = \tilde{I}(p) + n_k(p)$$

where $\tilde{I}(p)$ is the real information.

Temporal mean denoising

Temporal mean denoising Averaging N images taken at different time steps.

$$\begin{aligned} O(p) &= \frac{1}{N} \sum_{k=1}^{N} I_k(p) \\ &= \frac{1}{N} \sum_{k=1}^{N} \Big(\tilde{I}(p) + n_k(p) \Big) \\ &= \frac{1}{N} \sum_{k=1}^{N} \tilde{I}(p) + \overbrace{\frac{1}{N} \sum_{k=1}^{N} n_k(p)}^{\mu = 0} \\ &\approx \tilde{I}(p) \end{aligned}$$

Remark. As multiple images of the same object are required, this method is only suited for static images.

Spatial mean denoising

Spatial mean denoising Given an image, average across neighboring pixels.

Let K_p be the pixels in a window around p (included):

$$\begin{aligned} O(p) &= \frac{1}{|K_p|} \sum_{q \in K_p} I(p) \\ &= \frac{1}{|K_p|} \sum_{q \in K_p} \Big(\tilde{I}(q) + n(q) \Big) \\ &= \frac{1}{|K_p|} \sum_{q \in K_p} \tilde{I}(q) + \frac{1}{|K_p|} \sum_{q \in K_p} n(q) \\ &\approx \frac{1}{|K_p|} \sum_{q \in K_p} \tilde{I}(q) \end{aligned}$$

Remark. As the average of neighboring pixels is considered, this method is only suited for uniform regions.

2.2 Convolutions

2.2.1 Preliminaries

Convolution Given two functions f and g, their 1D convolution is defined as [3]: Continuous convolution

$$(f * g)(t) = \int_{-\infty}^{+\infty} f(\tau)g(t-\tau)\,\mathrm{d}\tau$$

In other words, at each t, a convolution can be interpreted as the area under $f(\tau)$ weighted by $g(t-\tau)$ (i.e. $g(\tau)$ flipped w.r.t. the y-axis and with the argument shifted by t).

Alternatively, it can be seen as the amount of overlap between $f(\tau)$ and $g(t-\tau)$.

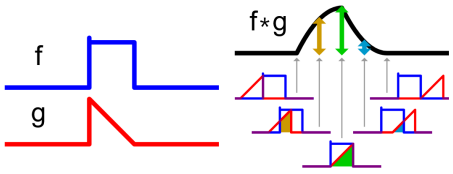


Figure 2.1: Example of convolution

Extended to the 2-dimensional case, the definition becomes:

$$(f * g)(x, y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(\alpha, \beta)g(x-\alpha, y-\beta)\,\mathrm{d}\alpha\,\mathrm{d}\beta$$

A convolution enjoys the following properties: Convolution properties

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

Commutative $f * g = g * f$.

Distributive w.r.t. sum $f * (g + h) = f * g + f * h$.

Commutative with differentiation $(f * g)' = f' * g = f * g'$

Dirac delta The Dirac delta "function" δ is defined as follows [5, 2]: Dirac delta

$$\forall x \neq 0 : \delta(x) = 0, \text{ constrained to } \int_{-\infty}^{+\infty} \delta(x)\,\mathrm{d}x = 1$$

Extended to the 2-dimensional case, the definition is the following:

$$\forall (x, y) \neq (0, 0) : \delta(x, y) = 0, \text{ constrained to } \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \delta(x, y)\,\mathrm{d}x\,\mathrm{d}y = 1$$

Sifting property The following property holds: Sifting property

$$\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x, y)\delta(\alpha - x, \beta - y)\,\mathrm{d}x\,\mathrm{d}y = f(\alpha, \beta)$$

Remark. Exploiting the sifting property, the signal of an image can be expressed through an integral of Dirac deltas (i.e. a linear combination) [1, 2]:

$$i(x, y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} i(\alpha, \beta)\delta(x - \alpha, y - \beta)\,\mathrm{d}\alpha\,\mathrm{d}\beta$$

Kronecker delta Discrete version of the Dirac delta [6]: Kronecker delta

$$\delta(x) = \begin{cases} 0 & \text{if } x \neq 0 \\ 1 & \text{if } x = 0 \end{cases}$$

Extended to the 2-dimensional case, the definition is the following:

$$\delta(x, y) = \begin{cases} 0 & \text{if } (x, y) \neq (0, 0) \\ 1 & \text{if } (x, y) = (0, 0) \end{cases}$$

Sifting property The following property holds: Sifting property

$$i(x, y) = \sum_{\alpha=-\infty}^{+\infty} \sum_{\beta=-\infty}^{+\infty} i(\alpha, \beta)\delta(x - \alpha, y - \beta)$$

2.2.2 Continuous convolutions

Image filter Operator that computes the new intensity of a pixel p based on the intensities of a neighborhood of p. Image filter

Remark. Image filters are useful for denoising and sharpening operations.

Linear translation-equivariant (LTE) operator A 2D operator $T\{\cdot\}$ is denoted as: LTE operator

$$T\{i(x, y)\} = o(x, y)$$

$T\{i(x, y)\}$ is LTE iff it is:

Linear Given two input 2D signals $i(x, y)$, $j(x, y)$ and two constants α, β, it holds that:

$$T\{\alpha \cdot i(x, y) + \beta \cdot j(x, y)\} = \alpha T\{i(x, y)\} + \beta T\{j(x, y)\}$$

Translation-equivariant Given an input 2D signal $i(x, y)$ and two offsets x_o, y_o, it holds that:

$$\text{if } T\{i(x, y)\} = o(x, y) \text{ then } T\{i(x - x_o, y - y_o)\} = o(x - x_o, y - y_o)$$

Impulse response/Point spread function/Kernel Given a 2D operator $T\{\cdot\}$, its impulse response, denoted with h, is the output of the operator when the input signal is a Dirac delta [1]:

$$h(x, y) \triangleq T\{\delta(x, y)\}$$

Theorem 2.2.1 (LTE operators as convolutions)**.** Applying an LTE operator on an image is equivalent to computing the convolution between the image and the impulse response h of the operator. LTE operators as convolutions

$$\begin{aligned} T\{i(x, y)\} &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} i(\alpha, \beta)h(x - \alpha, y - \beta)\,\mathrm{d}\alpha\,\mathrm{d}\beta \\ &= i(x, y) * h(x, y) \end{aligned}$$

In other words, the impulse response allows to compute the output of any input signal through a convolution.

Proof. Let $i(x, y)$ be an input signal and $T\{\cdot\}$ be a 2D operator. We have that:

$$
\begin{aligned}
T\{i(x,y)\} &= T\left\{\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} i(\alpha,\beta)\delta(x-\alpha,y-\beta)\,\mathrm{d}\alpha\,\mathrm{d}\beta\right\} && \text{sifting property}\\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} T\left\{i(\alpha,\beta)\delta(x-\alpha,y-\beta)\right\}\,\mathrm{d}\alpha\,\mathrm{d}\beta && \text{linearity of } T\{\cdot\}\\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} i(\alpha,\beta)T\left\{\delta(x-\alpha,y-\beta)\right\}\,\mathrm{d}\alpha\,\mathrm{d}\beta && \text{linearity of } T\{\cdot\}\\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} i(\alpha,\beta)h(x-\alpha,y-\beta)\,\mathrm{d}\alpha\,\mathrm{d}\beta && \text{translation-equivariance of } T\{\cdot\}\\
&= i(x,y) * h(x,y) && \text{definition of convolution}
\end{aligned}
$$

□

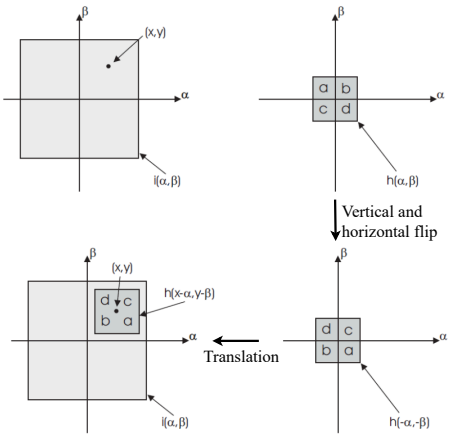


Figure 2.2: Visualization of a convolution

Cross-correlation Given two signals $i(x, y)$ and $h(x, y)$, their cross-correlation computes their similarity and is defined as follows [4]: Cross-correlation

$$
i(x,y)\circ h(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} i(\alpha,\beta)h(x+\alpha,y+\beta)\,\mathrm{d}\alpha\,\mathrm{d}\beta = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} h(\alpha,\beta)i(\alpha-x,\beta-y)\,\mathrm{d}\alpha\,\mathrm{d}\beta
$$

$$
h(x,y)\circ i(x,y) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} h(\alpha,\beta)i(x+\alpha,y+\beta)\,\mathrm{d}\alpha\,\mathrm{d}\beta = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} i(\alpha,\beta)h(\alpha-x,\beta-y)\,\mathrm{d}\alpha\,\mathrm{d}\beta
$$

Remark. Cross-correlation is not commutative.

Remark. The cross-correlation $h \circ i$ is similar to a convolution without flipping the kernel.

If h is an even function (i.e. $h(x, y) = h(-x, -y)$), we have that $h \circ i$ has the same result of a convolution:

$$
\begin{aligned}
h(x,y) * i(x,y) &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} i(\alpha,\beta)h(x-\alpha,y-\beta)\,\mathrm{d}\alpha\,\mathrm{d}\beta\\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} i(\alpha,\beta)h(\alpha-x,\beta-y)\,\mathrm{d}\alpha\,\mathrm{d}\beta && \begin{array}{r}\text{signs in } h \text{ swappable}\\ \text{for Dirac delta}\end{array}\\
&= h(x,y)\circ i(x,y)
\end{aligned}
$$

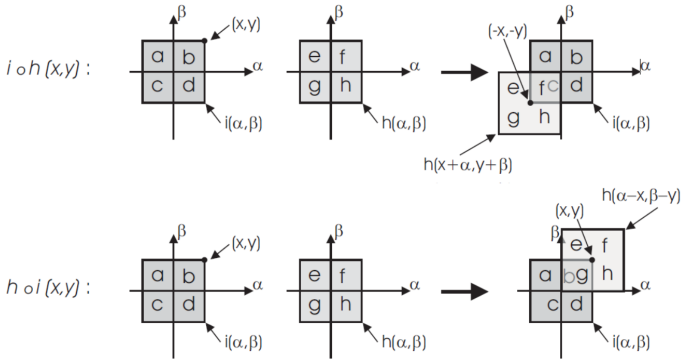


Figure 2.3: Visualization of cross-correlation

2.2.3 Discrete convolutions

Discrete convolution Given an input 2D signal $I(i,j)$ and the kernel $H(i,j) = T\{\delta(i,j)\}$ of a discrete LTE operator (where $\delta(i,j)$ is the Kronecker delta), a discrete convolution is defined as: Discrete convolution

$$T\{I(i,j)\} = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} I(m,n)H(i-m,j-n) = O(i,j)$$

In practice, the kernel is finitely defined and is applied to each pixel of the image:

$$T\{I(i,j)\} = \sum_{m=-k}^{k} \sum_{n=-k}^{k} K(m,n)I(i-m,j-n) = O(i,j)$$

Example. For simplicity, a kernel of size 3 is considered. Given an image I and a kernel K, the output $O(1,1)$ of the pixel $(1,1)$ is computed as:

$$\begin{aligned} O(1,1) &= \begin{pmatrix} I(0,0) & I(0,1) & I(0,2) \\ I(1,0) & I(1,1) & I(1,2) \\ I(2,0) & I(2,1) & I(2,2) \end{pmatrix} * \begin{pmatrix} K(0,0) & K(0,1) & K(0,2) \\ K(1,0) & K(1,1) & K(1,2) \\ K(2,0) & K(2,1) & K(2,2) \end{pmatrix} \\ &= I(0,0)K(2,2) + I(0,1)K(2,1) + I(0,2)K(2,0) + \\ &\quad + I(1,0)K(1,2) + I(1,1)K(1,1) + I(1,2)K(1,0) + \\ &\quad + I(2,0)K(0,2) + I(2,1)K(0,1) + I(2,2)K(0,0) \end{aligned}$$

Note that by definition, K has to be flipped.

Remark. In convolutional neural networks, the flip of the learned kernels can be considered implicit.

Border handling Computing the convolution of the pixels at the borders of the image might be an issue as it goes out-of-bounds, possible solutions are: Border handling

Crop Ignore border pixels on which the convolution overflows.

Pad Add a padding to the image:

Zero-padding Add zeros (e.g. `000`$|a \dots d|$`000`).

Replicate Repeat the bordering pixel (e.g. `aaa`$|a \dots d|$`ddd`).

Reflect Use the n pixels closest to the border (e.g. `cba`$|abc \dots dfg|$`gfd`).

Reflect_101 Use the n pixels closest to the border, skipping the first/last one (e.g. `dcb`$|abcd \dots efgh|$`gfe`).

2.2.4 Common linear kernels

Mean filter LTE operator that computes the intensity of a pixel as the average intensity of the pixels in its neighborhood. Mean filter

The kernel has the form (example with a 3×3 kernel):

$$\begin{pmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{pmatrix} = \frac{1}{9} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Remark. The mean filter has a low-pass effect which allows the removal of details from the signal. This allows for image smoothing and, to some extent, denoising (but adds blur).

Remark. As the intensity of a pixel is computed by averaging its neighborhood, the results for pixels located between low-intensity and high-intensity areas might not be ideal.

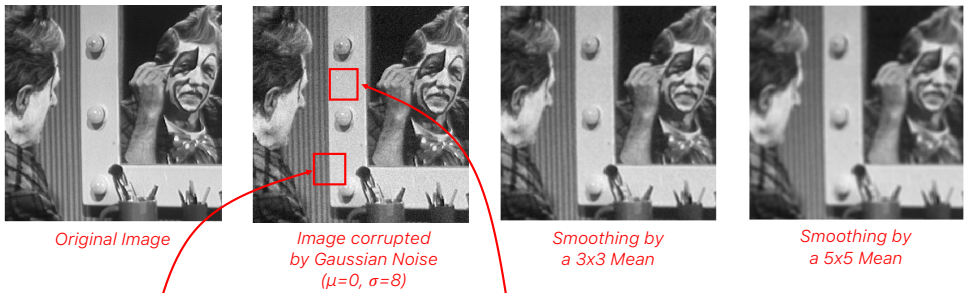


Figure 2.4: Example of mean filter application

Gaussian filter LTE operator whose kernel follows a 2D Gaussian distribution with $\mu = 0$ and given σ. Gaussian filter

Remark. The smoothing strength of the filter grows with σ.

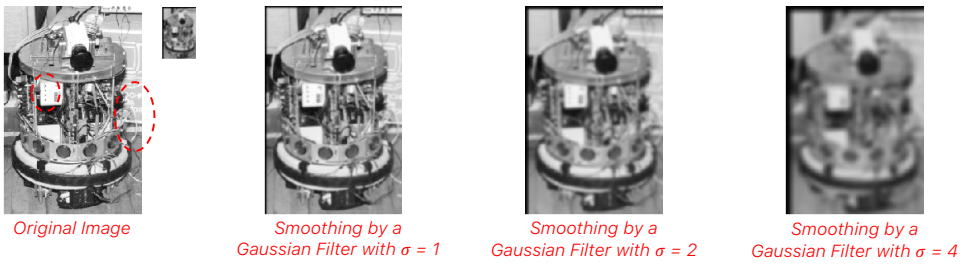


Figure 2.5: Example of Gaussian filter application

Sampling In practice, the kernel is created by sampling from the wanted Gaussian distribution. One can notice that a higher σ results in a more spread distribution and therefore a larger kernel is more suited, on the other hand, a smaller σ can be represented using a smaller kernel as it is more concentrated around the origin.

As a rule-of-thumb, given σ, an ideal kernel is of size $(3\sigma + 1) \times (3\sigma + 1)$.

Separability As a 2D Gaussian $G(x, y)$ can be decomposed into a product of two 1D Gaussians $G(x, y) = G_1(x)G_2(y)$, it is possible to split the convolution into two 1D convolutions.

$$\begin{aligned} I(x,y) * G(x,y) &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(\alpha,\beta) G(x-\alpha, y-\beta)\, \mathrm{d}\alpha\, \mathrm{d}\beta \\ &= \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(\alpha,\beta) G_1(x-\alpha) G_2(y-\beta)\, \mathrm{d}\alpha\, \mathrm{d}\beta \\ &= \int_{-\infty}^{+\infty} G_2(y-\beta) \left(\int_{-\infty}^{+\infty} I(\alpha,\beta) G_1(x-\alpha)\, \mathrm{d}\alpha \right) \mathrm{d}\beta \\ &= (I(x,y) * G_1(x)) * G_2(y) \end{aligned}$$

Remark. The speed-up in number-of-operations is linear.

2.2.5 Common non-linear kernels

Remark. Linear filters are ineffective when dealing with impulse noise and have the side effect of blurring the image.

Figure 2.6: Example of impulse noise and denoising with mean filter

Remark. As they lose linearity, non-linear filters are technically not convolutions anymore.

Median filter The intensity of a pixel is obtained as the median intensity of its neighborhood. Median filter

Remark. Median filters are effective in removing impulse noise (as outliers are excluded) without introducing significant blur. It also tends to result in sharper edges.

Remark. Median filters are not suited for Gaussian noise. It might be useful to apply a linear filter after a median filter.

Figure 2.7: Example of median filter application

Bilateral filter Given two pixels p and q, the following can be computed: Bilateral filter

Spatial distance $d_s(p,q) = \|p-q\|_2$

Range/intensity distance $d_r(p,q) = |\texttt{intensity}(p) - \texttt{intensity}(q)|$

Given a pixel p, its neighborhood $\mathcal{N}(p)$ and the variances σ_s, σ_r of two Gaussians, the bilateral filter applied on p is computes as follows:

$$O(p) = \sum_{q \in \mathcal{N}(p)} H(p,q) \cdot \texttt{intensity}(q)$$

$$\text{where } H(p,q) = \frac{G_{\sigma_s}(d_s(p,q))G_{\sigma_r}(d_r(p,q))}{\sum_{z \in \mathcal{N}(p)} G_{\sigma_s}(d_s(p,z))G_{\sigma_r}(d_r(p,z))}$$

where the denominator of H is a normalization factor.

Remark. Bilateral filters allow to deal with Gaussian noise without the introduction of blur.

Remark. Neighboring pixels with similar intensities result in larger weights in the filter, while pixels with different intensities (i.e. near an edge) result in smaller weights. This allows to effectively ignore pixels that belong to a different object from being considered when computing the intensity of a pixel.

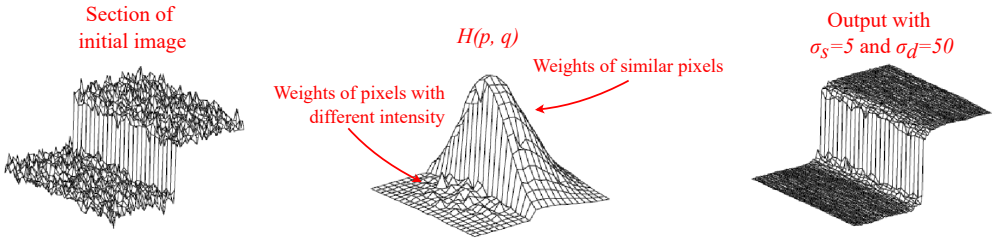
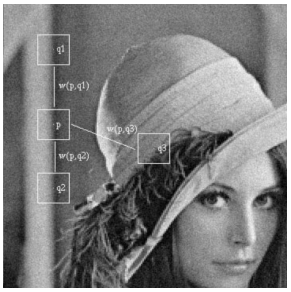


Figure 2.8: Example of bilateral filter application

Non-local means filter Exploits patches with similar pixels to denoise the image. Non-local means filter

Let:

- I be the image plane.
- $\mathcal{N}_i$ be the intensity matrix of a patch centered on the pixel i.
- S_i be a neighborhood of the pixel i.
- h be the bandwidth (a hyperparameter).



The intensity of a pixel p is computed as follows:

$$O(p) = \sum_{q \in S_p} w(p,q) \cdot \texttt{intensity}(q)$$

$$\text{where } w(p,q) = \frac{1}{Z(p)} e^{-\frac{\|\mathcal{N}_p - \mathcal{N}_q\|_2^2}{h^2}}$$

$$Z(p) = \sum_{q \in I} e^{\frac{\|\mathcal{N}_p - \mathcal{N}_q\|_2^2}{h^2}}$$

Instead of using the full image plane I, a neighborhood S_p is used for computational purposes. $Z(p)$ is a normalization factor.

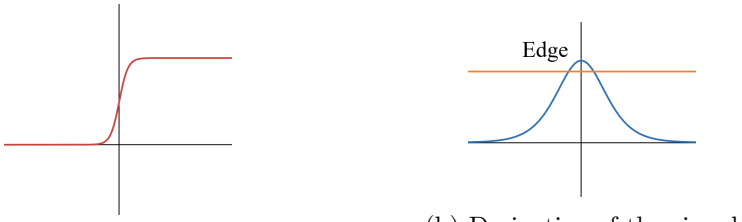
3 Edge detection

Edge Pixel lying in between regions of the image with different intensities. Edge

3.1 Gradient thresholding

3.1.1 1D step-edge

In the transition region, the absolute value of the first derivative grows (the absolute value is used as the polarity is not relevant). By fixing a threshold, an edge can be detected. 1D step-edge



(a) Input signal

(b) Derivative of the signal

3.1.2 2D step-edge

In a 2D signal (e.g. an image), the gradient allows to determine the magnitude and the direction of the edge. 2D step-edge

$$\nabla I(x,y) = \begin{pmatrix} \frac{\partial I(x,y)}{\partial x} & \frac{\partial I(x,y)}{\partial y} \end{pmatrix} = \begin{pmatrix} \partial_x I & \partial_y I \end{pmatrix}$$

$$\text{Magnitude: } \|\nabla I(x,y)\|$$

$$\text{Direction: } \arctan\left(\frac{\partial_y I}{\partial_x I}\right) \in [-\frac{\pi}{2}, \frac{\pi}{2}]$$

$$\text{Direction and sign: } \arctan 2(\partial_x I, \partial_y I) \in [0, 2\pi]$$

Discrete gradient approximation Approximation of the partial derivatives as a difference. Discrete gradient approximation

Backward difference Backward difference

$$\partial_x I(i,j) \approx I(i,j) - I(i,j-1) \qquad \partial_y I(i,j) \approx I(i,j) - I(i-1,j)$$

Forward difference Forward difference

$$\partial_x I(i,j) \approx I(i,j+1) - I(i,j) \qquad \partial_y I(i,j) \approx I(i+1,j) - I(i,j)$$

Remark. Forward and backward differences are equivalent to applying two cross-correlations with kernels $\begin{pmatrix} -1 & 1 \end{pmatrix}$ and $\begin{pmatrix} -1 \\ 1 \end{pmatrix}$.

Central difference Central difference

$$\partial_x I(i,j) \approx I(i,j+1) - I(i,j-1) \qquad \partial_y I(i,j) \approx I(i+1,j) - I(i-1,j)$$

Remark. Central difference is equivalent to applying two cross-correlations with kernels $\begin{pmatrix}-1 & 0 & 1\end{pmatrix}$ and $\begin{pmatrix}-1\\0\\1\end{pmatrix}$.

Discete magnitude approximation The gradient magnitude can be approximated using the approximated partial derivatives: Discete magnitude approximation

$$\|\nabla I\| = \sqrt{(\partial_x I)^2 + (\partial_y I)^2} \qquad \|\nabla I\|_+ = |\partial_x I| + |\partial_y I| \qquad \|\nabla I\|_{\max} = \max(|\partial_x I|, |\partial_y I|)$$

Among all, $\|\nabla I\|_{\max}$ is the most isotropic (i.e. gives a more consistent response).

Example. Given the following images:

$$E_v = \begin{pmatrix}0 & 0 & h & h\\0 & 0 & h & h\\0 & 0 & h & h\\0 & 0 & h & h\end{pmatrix} \quad E_h = \begin{pmatrix}0 & 0 & 0 & 0\\0 & 0 & 0 & 0\\h & h & h & h\\h & h & h & h\end{pmatrix} \quad E_d = \begin{pmatrix}0 & 0 & 0 & 0 & h\\0 & 0 & 0 & h & h\\0 & 0 & h & h & h\\0 & h & h & h & h\end{pmatrix}$$

The magnitudes are:

	$\|\nabla I\|$	$\|\nabla I\|_+$	$\|\nabla I\|_{\max}$
E_h	h	h	h
E_v	h	h	h
E_d	$\sqrt{2}h$	$2h$	h

Remark. In practice, the signal of an image is not always smooth due to noise. Derivatives amplify noise and are therefore unable to recognize edges.
Smoothing the signal before computing the derivative allows to reduce the noise but also blurs the edges making it more difficult to localize them.
A solution is to smooth and differentiate in a single operation by approximating the gradient as a difference of averages.

Smooth derivative Compute the approximation of a partial derivative as the difference of the pixels in a given window. For instance, considering a window of 3 pixels, the cross-correlation kernels are: Smooth derivative

$$\frac{1}{3}\begin{pmatrix}-1 & 1\\-1 & 1\\-1 & 1\end{pmatrix} \qquad \frac{1}{3}\begin{pmatrix}-1 & -1 & -1\\1 & 1 & 1\end{pmatrix}$$

Prewitt operator Derivative approximation using central differences. The cross-correlation kernels are: Prewitt operator

$$\frac{1}{3}\begin{pmatrix}-1 & 0 & 1\\-1 & 0 & 1\\-1 & 0 & 1\end{pmatrix} \qquad \frac{1}{3}\begin{pmatrix}-1 & -1 & -1\\0 & 0 & 0\\1 & 1 & 1\end{pmatrix}$$

Sobel operator Prewitt operator where the central pixels have a higher weight. The cross-correlation kernels are: Sobel operator

$$\frac{1}{4}\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix} \qquad \frac{1}{4}\begin{pmatrix}-1 & -2 & -1\\ 0 & 0 & 0\\ 1 & 2 & 1\end{pmatrix}$$

Remark. Thresholding is inaccurate as choosing the threshold is not straightforward. An image has strong and weak edges. Trying to detect one type might lead to poor detection of the other.
A better solution is to find a local maxima of the absolute value of the derivatives.

3.2 Non-maxima suppression (NMS)

Algorithm that looks for local maxima of the absolute value of the gradient along the gradient direction. Non-maxima suppression (NMS)
The algorithm works as follows:

1. Given a pixel at coordinates (i, j), estimate the magnitude $G = \|\nabla I(i, j)\|$ and the direction θ of the gradient.
2. Consider two points A and B along the direction θ passing through (i, j) and compute their gradient magnitudes G_A and G_B.
3. Substitute the pixel (i, j) as follows:

$$\texttt{NMS}(i,j) = \begin{cases} 1 & (G > G_A) \wedge (G > G_B) \text{ (i.e. local maximum)} \\ 0 & \text{otherwise} \end{cases}$$

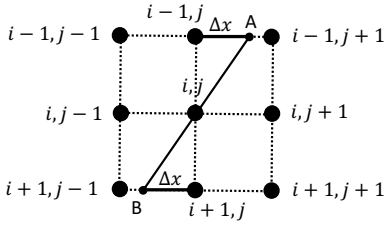
Remark. After applying NMS, the resulting signal (now composed of 0s and 1s) is converted back to the original gradient magnitudes in such a way that pixels 0ed by NMS remain 0 and pixels set at 1 by NMS return to their original value.
After the conversion, a thresholding step might be applied to filter out unwanted edges that are either due to noise or not relevant.

3.2.1 Linear interpolation

As there might not be two points A and B along the direction θ belonging the the discrete pixel grid and because one doesn't want to consider two points too far from (i, j), it is possible to use linear interpolation to estimate the gradients of A and B even if off-grid by an offset Δx:

$$G_1 = \|\nabla I(i-1, j)\| \qquad G_2 = \|\nabla I(i-1, j+1)\|$$
$$G_3 = \|\nabla I(i+1, j)\| \qquad G_4 = \|\nabla I(i+1, j-1)\|$$

$$G_A \approx G1 + (G2 - G1)\Delta x$$
$$G_B \approx G3 - (G3 - G4)\Delta x$$



3.3 Canny's edge detector

Method based on three criteria:

Good detection Correctly detect edges in noisy images.

Good localization Minimize the distance between the found edges and the true edges.

One response to one edge Detect only one pixel at each true edge.

In the 1D case, the optimal operator consists in finding the local extrema of the signal obtained by convolving the image and the Gaussian first-order derivative.
Generalized for the 2D case, Canny's edge detector does the following:

Canny's edge detector

1. Gaussian smoothing.
2. Gradient computation.
3. NMS and thresholding.

It is possible to exploit the convolutions property of being commutative w.r.t. differentiation and simplify the smoothing and gradient computation:

$$\partial_x I(x,y) = \frac{\partial}{\partial x}(I(x,y) * G(x,y)) = I(x,y) * \frac{\partial G(x,y)}{\partial x}$$
$$\partial_y I(x,y) = \frac{\partial}{\partial y}(I(x,y) * G(x,y)) = I(x,y) * \frac{\partial G(x,y)}{\partial y}$$

By leveraging the separability of a 2D Gaussian ($G(x,y) = G_1(x)G_2(y)$), the computation can be reduced to 1D convolutions:

$$\partial_x I(x,y) = I(x,y) * (G_1'(x)G_2(y)) = (I(x,y) * G_1'(x)) * G_2(y)$$
$$\partial_y I(x,y) = I(x,y) * (G_1(x)G_2'(y)) = (I(x,y) * G_1(x)) * G_2'(y)$$

Remark. When magnitude varies along the object contour, thresholding might remove true edges (edge streaking).

Hysteresis thresholding Given a high threshold T_h and a low threshold T_l, depending on its magnitude, a pixel (i,j) can be considered a:

Hysteresis thresholding

Strong edge $\nabla I(i,j) > T_h$.

Weak edge $\nabla I(i,j) > T_l$ and the pixel (i,j) is a neighbor of a strong/weak edge.

In practice, the algorithm starts from strong edges and "propagates" them.

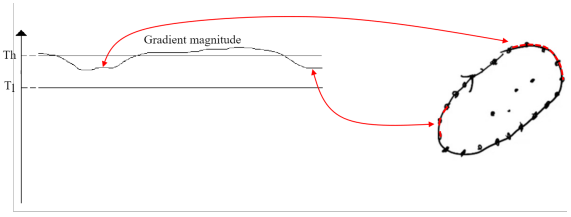


Figure 3.2: Application of hysteresis thresholding

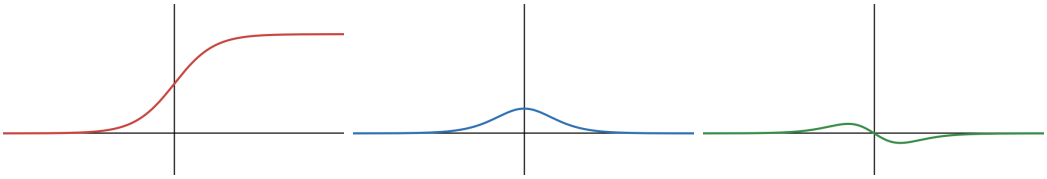
Remark. The output of Canny's edge detector is not an image with edges but a list of edges.
Note that if the edges of two objects intersect, this will be recognized as a single edge.

3.4 Zero-crossing edge detector

Zero-crossing edge detector Detect edges by finding zero-crossing of the second derivative of the signal. Zero-crossing

Remark. A zero-crossing is a point at 0 where the function changes sign.

Remark. This approach does not require a threshold anymore but is computationally more expensive.



(a) Input signal (b) First-order derivative (c) Second-order derivative

Laplacian Approximation of the second-order derivative: Laplacian

$$\nabla^2 I(x,y) \approx \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2} = \partial_{x,x} I + \partial_{y,y} I$$

Discrete Laplacian Use forward difference to compute first-order derivatives, followed by backward difference to compute second-order derivatives. Discrete Laplacian

$$\begin{aligned}
\partial_{x,x} I(i,j) &\approx \partial_x I(i,j) - \partial_x I(i,j-1) \\
&= \big(I(i,j+1) - I(i,j)\big) - \big(I(i,j) - I(i,j-1)\big) \\
&= I(i,j+1) - 2I(i,j) + I(i,j-1)
\end{aligned}$$

$$\begin{aligned}
\partial_{y,y} I(i,j) &\approx \partial_y I(i,j) - \partial_y I(i-1,j) \\
&= \big(I(i+1,j) - I(i,j)\big) - \big(I(i,j) - I(i-1,j)\big) \\
&= I(i+1,j) - 2I(i,j) + I(i-1,j)
\end{aligned}$$

This is equivalent to applying the cross-correlation kernel:

$$\nabla^2 = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Remark. It can be shown that zero-crossings of the Laplacian typically lay close to those of the second-order derivative.

3.4.1 Laplacian of Gaussian (LOG)

Laplacian of Gaussian (LOG) does the following: Laplacian of Gaussian (LOG)

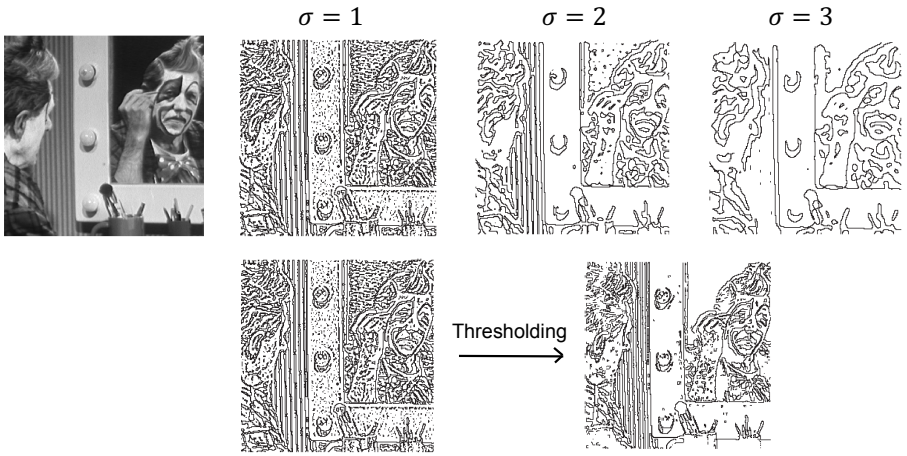
1. Gaussian smoothing.
2. Second-order differentiation using the Laplacian filter.
3. Zero-crossings extraction.

Discrete zero-crossing As the image consists of a discrete grid of pixels, finding a zero-crossing as per definition is not always possible. Instead, an edge is detected when there is a change of sign in the magnitude. The edge pixel can be identified as:

- The pixel where the magnitude is positive.
- The pixel where the magnitude is negative.
- The pixel where the absolute value of the magnitude is smaller. This is usually the best choice as it is closer to the true zero-crossing.

Remark. A final thresholding might still be useful to remove uninteresting edges.

Remark. Smaller values of σ of the smoothing operator detect more detailed edges, while higher σ captures more general edges.



4 Local features

Correspondence points Image points projected from the same 3D point from different views of the scene. Correspondence points

Example (Homography)**.** Align two images of the same scene to create a larger image. Homography requires at least 4 correspondences. To find them, it does the following:

- Independently find salient points in the two images.
- Compute a local description of the salient points.
- Compare descriptions to find matching points.

Local invariant features Find correspondences in three steps: Local invariant features

Detection Find salient points (keypoints). Detection

The detector should have the following properties:

Repeatability Find the same keypoints across different images.

Saliency Find keypoints surrounded by informative patterns.

Fast As it must scan the entire image.

Description Compute a descriptor for each salient point based on its neighborhood. Description

A descriptor should have the following properties:

Invariant Robust to as many transformations as possible (i.e. illumination, weather, scaling, viewpoint, ...).

Distinctiveness/robustness trade-off The description should only capture important information around a keypoint and ignore irrelevant features or noise.

Compactness The description should be concise.

Matching Identify the same descriptor across different images. Matching

Bibliography

[1] Ramani Duraiswami. *Filters*. http://users.umiacs.umd.edu/~ramani/cmsc828d_audio/Filters.pdf. 2006.

[2] M. Sonka, V. Hlavac, and R. Boyle. *Image Processing, Analysis, and Machine Vision*. Cengage Learning, 2015. ISBN: 978-1-133-59360-7.

[3] Wikipedia contributors. *Convolution — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Convolution&oldid=1212399231. 2024.

[4] Wikipedia contributors. *Cross-correlation — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Cross-correlation&oldid=1193503271. 2024.

[5] Wikipedia contributors. *Dirac delta function — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Dirac_delta_function&oldid=1198785224. 2024.

[6] Wikipedia contributors. *Kronecker delta — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Kronecker_delta&oldid=1192529815. 2023.