1 Image formation and acquisition

1.1 pinhole camera

simplest imaging device
geometrically, image is achieved by drawing straight lines from points in the scene through the pinhole to the image plane

1.2 Perspective projection

vars of intrest:

- M(x,y,z): scene point
- m(u,v): corresponding image point
- I: image plane
- C: optical centre
- Line through C and orth to I: optical axis
- c: intersec between optical axis and image plane (piercing point/image center)
- f: focal length
- F: focal plane

equations that relate 3d obj coords (x,y,z) to image coords:

$$\frac{u}{x} = \frac{v}{y} = -\frac{f}{z} \tag{1}$$

to get rid of up/down and left/right inversions we can write

$$u == x\frac{f}{z}; \quad v = y\frac{f}{z} \tag{2}$$

going from 3D to 2D implies a loss of information

1.3 stereo vision

Standard stereo geometry:
-Cameras arranged so that there is only horiz translation between the two CRF. Naming b the distance between the two CRF the transformation between the CRF is

$$p_L - p_R = \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix} \tag{3}$$

with p_L, p_R coords in left and right CRF
-same focal length for for both cameras (coplanar image planes)

$$x_L - x_R = b$$
$$y_L = y_R = y$$
$$z_L = z_R = z$$
$$v_L = v_R = y\frac{f}{z}(*)$$
$$u_L = x_L\frac{f}{z}$$
$$u_R = x_R\frac{f}{z}$$
$$\Rightarrow u_L - u_R = b\frac{f}{z}$$
$$u_L - u_R = d(disparity)$$
$$\Rightarrow d = b\frac{f}{z}$$
$$\Rightarrow z = b\frac{fe}{d}$$

eq. for y(*) helps solve correspondence as corresponding points have the same y coord in both images ⇒ better efficiency and lower risk of mistakes due to smaller search set

1.4 Epipolar geometry

The search space of the stereo correspondence problem is always 1D because all possible 3D points that can be correspondent to a pixel lie along a line (optical ray)
epipolar line: projection of an optical ray of one camera onto the image of the other (useful if not in standard stereo geometry)
Standard stereo geometry is more computationally efficient
Dense depth map: image in which depth is known for every pixel
images from epipolar cameras can be rectified for computational efficiency

1.5 properties of perspective projection

- the transformation is non-linear
- perspective projection maps 3D lines into image lines (straight)
- ratios of length are not preserved unless the scene is planar and parallel to the image plane
- parallelism between 3D lines is not preserved: they converge in a vanishing point unless the lines are in a plane parallel to the image plane
  - the vanishing point of a 3D line is the image of the point at infinity of the line
  - that is the intersection of between the parallel to the line passing through the optical center and the image plane

A scene point is in focus when all its light rays converge into a single point: always true for pinhole camera.
Drawback: long exposure time is needed due to infinitesimal pinhole and scene has to be static.
Solution: lenses that gather light and focus it in a single point

1.6 thin lens model

$$\frac{1}{u} + \frac{1}{v} = \frac{1}{f} \tag{4}$$

- rays parallel to the optical axis are deflected to pass through F
- rays through C are undeflected

If the image ins in focus, the image formation process obeys the perspective projection model, with the center of the lens being the optical center and the distance v acting as the effective focal length of the projection
given an image distance, 1 scene plane is in focus
scene points outside the focusing plane are mapped into circles of confusion/blur circles
this problem is alleviated in some ways:

- method of acquisition (if blur circle is small enough it falls into one pixel)
- lens size (smaller lenses → smaller blur circles)
- diaphragm (affects effective lens size used)

F number = ratio of focal length to effective aperture (f/d) focusing mechanism: lens is moved wrt the image plane (sensor)

1.7 Image digitization

sensor converts irradiance into an electric quantity (e.g. voltage), which is sampled (2D array of pixels) and quantized (typically 8 bit in grayscale → 256 levels)
In practice image is sampled when sensed due to arrays being arranged in an array
In digital cameras sensor array=pixels
In analog cameras image from sensor array s conveerted to analog signal and then reconverted to digital but pixels ≠ camera sensor array
sensors are typically CCD or CMOS
CCD/CMOS is not sensitive to light wavelength → colour filtering arrays (CFA)

1.8 Camera parameters

- Signal to Noise Ratio (SNR). Main noise sources:
  - Photon Shot Noise: number of photons collected during exposure time is random (Poisson statistics)
  - Electronic Circuitry Noise
  - Quantization Noise: related to ADC conversion (in digital cameras)
  - Dark Current Noise: a random amount of charge is generated in photocapacitors in the dark due to thermal excitement. A lower bound E_{min} is set on the detected irradiation
- Dynamic Range (DR): E_{max}= irradiation that would fill up the capacity of a photodetector
  - $DR = \frac{E_{max}}{E_{min}}$
- Sensitivity: amount of signal the sensor can deliver per unit of input optical energy
- Uniformity: responses to light and dark noise vary from pixel to pixel

both SNR and DR are usually specified in decibels or bits
CCD typically provides higher SNR, DR, and better uniformity than CMOS
with CMOS electronics can be integrated on the same chip → compactness, power efficiency, lower cost.
With CMOS an arbitrary window can be read out without haveing to receive the full image
to create colour an array of CFAs is placed in front of the photodetectors so as to render each pixel sensitive to a specific range of wavelengths

1.9 Perspective projection

By appending a 1 to the Euclidean coordinates of the object we obtain the coordinates in the projective space $\mathbb{P}^3$. We assume

$$(x \quad y \quad z \quad 1) = (kx \quad ky \quad kz \quad k) \forall k \neq 0 \tag{5}$$

these are called homogeneous coordinates (aka projective coordinates) representation of the 3D point having Euclidian coordinates (x,y,z)

1.9.1 Point at infinity of a 3D line

Considering the parametric equation of a 3D line:

$$M = M_0 + \lambda D = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + \lambda \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} x_0 + \lambda a \\ y_0 + \lambda b \\ z_0 + \lambda c \end{bmatrix} \tag{6}$$

in projective coordinates

$$\tilde{M} = \begin{bmatrix} M \\ 1 \end{bmatrix} = \begin{bmatrix} x_0 + \lambda a \\ y_0 + \lambda b \\ z_0 + \lambda c \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x_0}{\lambda} + a \\ \frac{y_0}{\lambda} + b \\ \frac{z_0}{\lambda} + c \\ \frac{1}{\lambda} \end{bmatrix} \tag{7}$$

by taking the limit with $\lambda \to \infty$ we obtain the projective coordinates of the point at infinity of the given line:

$$\tilde{M}_\infty = \lim_{\lambda \to \infty} \tilde{M} = \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} \tag{8}$$

- to map points at infinity from the projective space to the Euclidean space we would divide by the fourth coordinate (0)
- point (0,0,0,0) is undefined
- it can be shown that all points at infinity in $\mathbb{P}^3$ lie on a plane called plane at infinity

1.9.2 Perspective projection in projective coordinates

$$u = \frac{f}{z}x \qquad v = \frac{f}{z}y \tag{9}$$

$$M = [x, y, z]^T \qquad m = [u, v]^T \tag{10}$$

$$\tilde{M} = [x, y, z, 1]^T \qquad \tilde{m} = [u, v, 1]^T \tag{11}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f\frac{x}{z} \\ f\frac{y}{z} \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{12}$$

or in matrix notation

$$k\tilde{m} = \tilde{P}\tilde{M} \tag{13}$$

often expressed as

$$\tilde{m} \approx \tilde{P}\tilde{M} \tag{14}$$

where $\approx$ means "equal up to an arbitrary scale factor"
if we assume distances to be measured in focal lengths ($f = 1$), the PPM becomes

$$\tilde{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = [I_3|0] \tag{15}$$

called the canonical PPM

1.10 Building a useful camera model

For a useful camera model we need to take into account:

- Image digitization
- rototranslation between the CRF (camera reference frame) and the WRF (world reference frame)

1.10.1 Image digitization

Digitization can be accounted for by including into the projection equations the scaling factors along the two axes due to the quantization associated with the horizontal and vertical pixel size (Δu and Δv). We also need to model the translation of the piercing point wrt the origin of the image coordinate system

$$\begin{aligned} u = \frac{f}{z}x \quad &\rightarrow u = \frac{1}{\Delta u}\frac{f}{z}x = k_u\frac{f}{z}x + u_0 \\ v = \frac{f}{z}y \quad &\rightarrow v = \frac{1}{\Delta v}\frac{f}{z}x = k_u\frac{f}{z}x + u_0 \end{aligned}$$

The PPM can therefore be rewritten as

$$\tilde{P} = \begin{bmatrix} fk_u & 0 & u_0 & 0 \\ 0 & fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = A[I|0] \tag{16}$$

Matrix A, which models the characteristics of the camera, is called *Intrinsic Parameter Matrix*
$fk_u = \alpha_u$ and $fk_v = \alpha_v$ are called horizontal and vertical scale factors

1.10.2 Rigid motion between CRF and WRF

The WRF can be related to the CRF by:

- a rotation around the optical centre (3x3 rotation matrix R)
- a translation (3x1 translation vector T)

Therefore, the relation between coords of a point in the 2 RF is:

$$W = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, M = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow M = RW + T \tag{17}$$

in homogenous coordinates:

$$\tilde{W} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tilde{M} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \Rightarrow \tilde{M} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tilde{W} = G\tilde{W} \tag{18}$$

A 3D point is therefore mapped into the CRF as

$$k\tilde{m} = A[I|0]\tilde{M} \tag{19}$$

by considering the rigid body motion between WRF and CRF:

$$\tilde{M} = G\tilde{W} \rightarrow k\tilde{m} = A[I|0]G\tilde{W} \tag{20}$$

$$k\tilde{m} = A[I|0] \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tilde{W} \tag{21}$$

Accordingly, the general form of the PPM can be expressed as:

$$\tilde{P} = A[I|0]G \quad \text{or also} \quad \tilde{P} = A[R|T] \tag{22}$$

Matrix G, which encodes the position and orientation of the camera wrt the WRF, is called Extrinsic Parameter Matrix
To be noted: the actual DoFs of G are 6, 3 due to rotation and 3 due to translation

1.11 Homographies

1.11.1 P as a Homography

If the camera is imaging a planar scene we can assume the z axis of the WRF to be perpendicular to the plane so that all points have z coordinate 0. The PPM simplifies as follows:

$$k\tilde{m} = \tilde{P}\tilde{W} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{24} \\ p_{31} & p_{32} & p_{34} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = H\tilde{M} \tag{23}$$

where H is denoted as a *homography*, i.e. a linear transformation between projective planes.
Akin to P, H is known up to an *arbitrary scale factor* and thus independent elements in H(3x3) are just 8.
H is always invertible
Further examples of cases where homographies are applicable:

- two images of a planar scene
- two images taken by a camera rotating about the optical centre
- two images taken by different cameras (i.e. different A) in a fixed pose (i.e. same CRF)
- two images taken by different cameras rotated about the optical centre

1.12 Lens distortion

Lens distortion is modelled through additional parameters that don't alter the form of the PPM

- Radial distortion: points at the same distance from the centre of distortion (radius) are affected in the same way by the distortion, and distortion grows with the radius. Can be split into:
  - barrel distortion
  - pincushion distortion

- tangential distortion: due to misalignment of optical components and/or defects

Distortion is modelled through a non-linear mapping between ideal image coordinates and observed (distorted) image coordinates, which is a mapping between *CONTINUOUS* coordinates

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = L(r) \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} + \begin{bmatrix} d\tilde{x} \\ d\tilde{y} \end{bmatrix} \tag{24}$$

(x', y') are the distorted coordinates and $(\tilde{x}, \tilde{y})$ are the undistorted coordinates, r is the distance from the distortion centre (x_c, y_c) $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$
The first term regards radial distortion, while the second deals with tangential distortion
$L(r)$ is usually approximated by its Taylor series around the centre of distortion ($r = 0$)

$$L(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots \tag{25}$$

$L(0) = 1$, therefore there is no distortion at the centre of distortion. The odd terms are not present beacause $L(r)$ is undefined for $r < 0$, so it can be considered to be an even function.

2 Camera calibration [INCOMPLETE]

for the sake of camera calibration, patterned surfaces such as checkboards are used to find corresponding points.

2.1 Zhang's Method

1. Acquire n images of a planar pattern with m internal corners
2. comput a homography H_i for each image
3. Refine each H_i by minimizing the reprojetion error
4. given all the H_i get an initial guess for the intrinsics A
5. Given A and H_i get an initial guess for R_i and T_i (extrinsics)
6. compute an initial guess for lens distortion k
7. refine A, R_i, T_i, k by minimizing the reprojection error

2.2 Calibration of a stereo camera

The two cameras can be separately calibrated using Zhang's algorithm.
Rototranslation between the two cameras needs to be calibrated

3 Image filtering

Image filter: image processing algorithm that computes the new intensity (colour) of a pixel p based on intensities (colours) of pixels in its neighbourhood

3.1 Linear Translation-Equivariant filters

Given an input 2D signal $i(x, y)$, a 2D operator $T\{\dot{}\} : o(x, y) = T\{i(x, y)\}$ is said to be linear iff:

$$T\{ai_1(x, y) + bi_2(x, y)\} = ao_1(x, y)bo_2(x, y)$$

with $o_1(\dot{}) = T\{i_1(\dot{})\}, o_2(\dot{}) = T\{i_2(\dot{})\}$ and a,b constants.
$T\{\dot{}\}$ is said to be translation-equivalent iff:

$$T\{i(x - x_0, y - y_0)\} = o(x - x_0, y - y_0)$$

If the operator is LTE, the output signal is given by the convolution between the impulse response (point spread function), $h(x, y) = T\{\delta(x, y)\}$ of the operator and the input signal.

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

LTE filters can be expressed through convolutions

3.2 Convolution

3.2.1 Properties of convolution

- Associative property
- Commutative property
- Distributivity wrt sum
- Commutation with differentiation $(f * g)' = f' * g = f * g'$

3.2.2 Correlation

The correlation of signal $i(x, y)$ wrt signal $h(x, y)$ is:

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

Correlation is not commutative.
If h is an even function ($h(x, y) = h(-x, -y)$):

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

hence if h is even, the convolution between i and h is the same as the correlation of h wrt i.

3.2.3 Discrete convolution

For discrete signals, convolution is defined as foolows:

$$O(i, j) = T\{I(i, j)\} = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} I(m, n) H(i - m, j - n) \tag{26}$$

where $I(i, j)$ and $O(i, j)$ are the discrete 2D input and output signals and $H(i, j) = T\{\delta(i, j)\}$ is the kernel of the LTE operator, i.e. the response to the 2D discrete unit impulse (Kronecker delta function) $\delta(i, j)$. Properties of convolution hold in the discrete case.

3.2.4 Practical implementation of discrete convolution

Due to the image being much larger than the kernel, commutativity is exploited in order to obtain a more efficient implementation of convolution:

$$O(i, j) = T\{I(i, j)\} = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} K(m, n) I(i - m, j - n) \tag{27}$$

This equates to sliding the kernel across the whole input image and computing the convolution at each pixel. Pixels close to the border will lack some pixels for the computation of the convolution that would be outside the image. There are two main solutions:

- CROP: common in image processing, bad with chains of convolution
- PAD: zero padding, replicate, reflect, reflect_101

Computational cost: $(2k + 1)^2$ MADs, $2(2k + 1)^2$ operations per pixel.

4 Denoising

4.1 Taking a mean across time

The output at point p is:

$$O(p) = \frac{1}{N}\sum_{k=1}^{N} I_k(p) = \frac{1}{N}\sum_{k=1}^{N}(\tilde{I}(p) + n_k(p)) = \frac{1}{N}\sum_{k=1}^{N}\tilde{I}(p) + \frac{1}{N}\sum_{k=1}^{N} n_k(p) \cong \tilde{I}(p) \tag{28}$$

where $\tilde{I}(p)$ is the ideal undisturbed signal and $n_k(p)$ is the noise by which the signal is affected. Tipycally though, only a single image is available, therefore this approach is not feasible.

4.2 Mean Filter

If we are given a single image, we may compute a mean across neighbouring pixels, i.e. a spatial mean. Mean filtering is the simplest and fastest way to denoise an image. It consists in replacing each pixel intensity with the average intensity over a chosen (square) neighbourhood. The Mean Filter is an LTE operator as it can be expressed as a convolution with a kernel, e.g. for a 3x3 mean filter:

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

According to signal theory, the Mean Filter carries out a *low pass filtering* operation, which is referred to in image processing as *image smoothing*. Smoothing is ofthen aimed at denoising, though sometimes its purpose is to cancel out small-size unwanted details that might hinder image analysis. Mean filtering is inherently fast as multiplications are not needed. Moreover, it can be implemented very efficiently by incremental calculation schemes (Box-filtering)

4.2.1 Box-Filtering

Given boxes of size $2k+1$

$$\mu(i,j) = \frac{\sum_{m=-k}^{n=k}\sum_{n=-k}^{n=k} I(i+m, j+n)}{(2k+1)^2} = \frac{s(i,j)}{2k+1)^2}$$

Where

$$s(i, j+1) = s(i,j) + V^+(i, j+1) - V^-(i, j+1)$$

and

$$V^+(i, j+1) = V^+(i-1, j+1) + a - b$$
$$V^-(i, j+1) = V^-(i-1, j+1) + c - d$$
$$V^+(i, j+1) - V^-(i, j+1) = \Delta(i, j+1)$$

Resulting in

$$s(i, j+1) = s(i,j) + \Delta(i-1, j+1) + a - b - c + d$$

Insert image from slides
We therefore have only 5 sums per pixel, independently of kernel size.

4.3 Gaussian Filter

LTE operator whose impulse response is a 2D Gaussian function (with zero mean and constant diagonal covariance matrix). The paramenter σ sets the amount of smoothing. As σ gets larger, the amount of smoothing grows.

The discrete Gaussian kernel can be obtained by sampling the corresponding continuous function, which is however of infinite extent. A finite size must be properly chosen. As the interval $[-3\sigma, +3\sigma]$ captures 99% of the area of the Gaussian function, a typical rule of thumb dictates taking a $(2k+1) \times (2k+1)$ kernel with $k = 3\lceil\sigma\rceil$. It can be noted that, as the size of the kernel grows:

- the more accurate the discrete approximation of the ideal filter turns out
- th more the computational cost grows
- the Gaussian gets smaller and smaller as we move away from the origin

4.3.1 deploying separability

To further speed up computation, one may observe that the 2D Gaussian is the product of two 1D Gaussians, therefore the original 2D convolution may be split into the chain of two 1D convolutions along x and along y

$$I(x,y) * G(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(\alpha,\beta)G(x-\alpha,y-\beta)d\alpha d\beta \tag{29}$$

$$G(x,y) = G(x)G(y) \tag{30}$$

$$I(x,y) * G(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} I(\alpha,\beta)G(x-\alpha)G(y-\beta)d\alpha d\beta \tag{31}$$

$$I(x,y) * G(x,y) = \int_{-\infty}^{+\infty} G(y-\beta) \left(\int_{-\infty}^{+\infty} I(\alpha,\beta)G(x-\alpha)d\alpha \right) d\beta \tag{32}$$

$$I(x,y) * G(x,y) = (I(x,y) * G(x)) * G(y) = (I(x,y) * G(y)) * G(x) \tag{33}$$

The speed-up brought in by the separability property can be expressed as:

$$\text{2D Filter: } N_{OPS} = 2 \times (2k+1)^2 \tag{34}$$

$$\text{1D Filter: } N_{OPS} = 2 \times 2 \times (2k+1) \tag{35}$$

$$S = \frac{2 \times (2k+1)^2}{2 \times 2 \times (2k+1)} = k + \frac{1}{2} \tag{36}$$

4.4 Median Filter

A non-linear filter whereby each pixel intensity is replaced by the mediano over a given neighbourhood. Median filtering counteracts impulse noise effectively, as outliers tend to fall at either the top or bottom of the sorted intensities. Median filtering also tends to keep sharper edges than linear filters such as the mean or gaussian, it cannot however correct Gaussian-like noise. A standard preprocessing pipeline includes first a median filter followed by a gaussian filter.

4.5 Bilateral Filter

An advanced non-linear filter to accomplish the denoising of a Gaussian-like noise without blurring the image (aka *edge preserving smoothing*). The kernel is computed based on the distance from the considered pixel and the difference in intensity.

$$O(p) = \sum_{q \in S} H(p,q) I_q$$

where $H(p,q)$ is a weighing function of p(central pixel) and q(neighbour pixel)

$$H(p,q) = \frac{1}{W(p)} G_{\sigma_s}(d_s(p,q)) G_{\sigma_r}(d_r(I_p, I_q))$$

where G_{σ_s} is the spatial Gaussian, G_{σ_r} is the intensity Gaussian, d_s is the spatial distance, d_r is the range (intensity distance), and $W(p)$ is the Normalization Factor

$$d_s(p,q) = \|p-q\|_2 = \sqrt{(u_p - u_q)^2 + (v_p - v_q)^2}$$

$$d_r(I_p, I_q) = |I_p - I_q|$$

$$W(p) = \sum_{q \in S} G_{\sigma_s}(d_s(p,q)) G_{\sigma_r}(d_r(I_p, I_q))$$

4.6 Non-local Means Filter

A non-linear edge preserving smoothing filter. It considers similarities among neighbourhoods spread over the image

$$O(p) = \sum_{q \in I} w(p,q) I(q)$$

where

$$w(p,q) = \frac{1}{Z(p)} e^{-\frac{\|N_p - N_q\|_2^2}{h^2}}$$

$$Z(p) = \sum_{q \in I} e^{-\frac{\|N_p - N_q\|_2^2}{h^2}}$$

In practice, to reduce the computational burden, only a region of the image is considered rather than the whole image.

5 Image segmentation and blob analysis

sigh some notes have been lost to the doom that are amd drivers for ubuntu, and will not for the time being be rewritten. We carry on, pondering the option of archlinux with growing intensity.

5.1 Coulour-based segmentation

In several applications, the sought for objects exhibit a known colour quite different from that of background structures. Hence, it is reasonable to try segmenting out foreground from background based on colour information. Let us denote a pixel's colour as:

$$I(p) = \begin{bmatrix} I_r(p) \\ I_g(p) \\ I_b(p) \end{bmatrix}$$

Then, segmentation can be achieved by computing and thresholding the distance between each pixel's colour and the reference background coloour μ:

$$\forall p \in I : \begin{cases} d(I(p), \mu) \leq T \rightarrow O(p) = B \ (F) \\ d(I(p), \mu) > T \rightarrow O(p) = F \ (B) \end{cases}$$

$$d(I(p), \mu) = ((I_r(p) - \mu_r)^2 + (I_g(p) - \mu_g)^2 + (I_b(p) - \mu_b)^2)^{\frac{1}{2}}$$

It is thus necessary to know the reference colour, μ, which can be conveniently estimated from one or more training images, by taking an average of pixels in the image. The decision surface would therefore be a sphere in the RGB colour spaces centered at μ with a radius as large as T. Eulidean distance can however lead to mistakes due to training samples exhibiting different variation along different channels.

5.1.1 Mahalanobis Distance

given the covariance matrix of the pixel values:

$$\Sigma = \begin{pmatrix} \sigma_{rr}^2 & \sigma_r g^2 & \sigma_{rb}^2 \\ \sigma_{gr}^2 & \sigma_{gg}^2 & \sigma_{gb}^2 \\ \sigma_{br}^2 & \sigma_{bg}^2 & \sigma_{bb}^2 \end{pmatrix} \Leftarrow \begin{cases} \sigma_{ij}^2 = \frac{1}{N} \sum_{k=1}^{N} (I_i(p_k) - \mu_i)(I_j(p_k) - \mu_j) \\ i, j \in \{r, g, b\} \end{cases}$$

and recalling that the Euclidean distance can be rewritten as:

$$d(I(p), \mu) = ((I(p) - \mu)^T (I(p) - \mu))^{\frac{1}{2}}$$

The *Mahalanobis Distance* is given by:

$$d_M(I(p), \mu) = ((I(p) - \mu)^T \Sigma^{-1} (I(p) - \mu))^{\frac{1}{2}}$$

wlog let's consider a diagonal Σ (i.e. independent components in $I(p)$)

$$\Sigma = \begin{pmatrix} \sigma_{rr}^2 & 0 & 0 \\ 0 & \sigma_{gg}^2 & 0 \\ 0 & 0 & \sigma_{bb}^2 \end{pmatrix} \Rightarrow \begin{pmatrix} 1/\sigma_{rr}^2 & 0 & 0 \\ 0 & 1/\sigma_{gg}^2 0 & \\ 0 & 0 & 1/\sigma_{bb}^2 \end{pmatrix}$$

$$\Sigma^{-1}(I(p) - \mu) = \left[(I_r(p) - \mu_r)/\sigma_{rr}^2 (I_g(p) - \mu_g)/\sigma_{gg}^2 (I_b(p) - \mu_b)/\sigma_{bb}^2 \right]$$

$$d_M(I(p), \mu) = \left(\frac{(I_r(p) - \mu_r)^2}{\sigma_{rr}^2} + \frac{(I_g(p) - \mu_g)^2}{\sigma_{gg}^2} + \frac{(I_b(p) - \mu_b)^2}{\sigma_{bb}^2} \right)^{\frac{1}{2}}$$

If we consider the equation

$$d_M^2(I(p), \mu) \leq T^2$$

we get that the decision surface is an ellipsoid centered at μ with axes aligned to the coordinate axes and semi-axes lengths given by

$$L = \begin{bmatrix} L_r \\ L_g \\ L_b \end{bmatrix} = T \begin{bmatrix} \sigma_r r \\ \sigma_g g \\ \sigma_b b \end{bmatrix}$$

Note that the Mahalanobis distance weighs the differences along components of the vector differently. In particular, they are weighed according to inverse proportionality to the learned variances.

Due to Σ being real ad symmetric, it can always be diagonalized through eigenvalue decomposition:

$$\Sigma = RDR^T \qquad R = (e_1, e_2, e_3), \quad D = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_1 & 0 \\ 0 & 0 & \lambda_1 \end{pmatrix}$$

where e_i are (orthonormal) eingenvectors of Σ, λ_i the corresponding eigenvalues and R^T the rotation matrix to transform the data into a new coordinate system having axes aligned to the eigenvectors.

5.2 Binary Morphology operators

tools used to improve or analyse binary images. The operators manipulate the image by a small "kernel" referred to as *structuring elements*.

5.2.1 Dilation (Minkowsky sum)

The output image is obtained by sliding the structuring element at each foreground pixel in the image. Useful for correcting false negatives.

5.2.2 Erosion (Minkowsky subtraction)

The output image is obtained by sliding the structuring element across the whole image keeping only those pixels where the structuring element fits the foreground pixels. Useful for correcting false positives.

5.2.3 Opening

Erosion followed by dilation:

$$A \circ B = (A \ominus B) \oplus B$$

Removes parts of the foreground that don't match the structuring element.

5.2.4 Closing

Dilation followed by erosion:

$$A \bullet B = (A \oplus B) \ominus B$$

Like opening the background by a flipped structuring element

5.3 Connected component labeling

Different blobs are given different labels while the background is left unaffected. Connected components (blobs) are foreground regions that are connected. Connectivity may be defined through 4-neighbourhoods or 8-neighbourhoods (4-way connectivity and 8-way connectivity). A path from pixel p to pixel q is a sequence of pixels $p = p_1, p_2, \ldots p_n = q$ s.t. p_i and p_{i+1} are neighbouring. A connected region is a region s.t for any two pixels p, q in that region there exists a path between p and q contained in the region.

5.4 Blob analysis

Blobs can be analysed to extract a few useful pieces of information:

5.4.1 Area

$$A = \sum_{p \in R} 1$$

Equal to the number of pixels in the blob

5.4.2 Barycentre

$$B = \begin{bmatrix} i_B \\ j_B \end{bmatrix} \quad i_B = \frac{1}{A}\sum_{p\in R} i \quad j_B = \frac{1}{A}\sum_{p\in R} j \quad p = \begin{bmatrix} i \\ j \end{bmatrix}$$

5.4.3 Orientation

Defined according to the axis (line through the baricenter) of least inertia (major axis):

$$\arg\min_I \left(\sum_{p\in\mathbb{R}} d_I^2(p) \right)$$

Orientation is typically defined as the angle between the major axis and the horizontal axis of the image.

5.4.4 Oriented Bounding Box

aka Minimum Enclosing Rectangle: given by the major and minor[1] axes by finding the points that have the highest distance from the major and minor axes on either side, and computing the lines parallel to the axes through said points and their intersections.

5.4.5 Lenght and Width

Length: extent of the object along the major axis
Width: extent of the object along the minor axis

5.4.6 Elongatedness

Ratio between length and width:

$$E = \frac{L}{W}$$

5.4.7 Rectangularity

Ratio between the area of the object and the area of the OBB:

$$R = \frac{A}{LW}$$

5.4.8 Ellipticity

Ratio between the area of the object and the area of the ellipse having major and minor axes equal to L and W respectively

$$E = \frac{A}{A_{LW}}, \quad A_{LW} = \frac{\pi}{4}LW$$

6 Local features

Edges: pixels that lie inbetween two different uniform regions.

6.0.1 1D Step-Edge

In the transitoin region between the two constant levels, the absolute value of the first derivative is high (and reaches an extremum at the inflection point) Accordingly, the simplest edge-detection operator relies on thresholding the absolute value of the first derivative of the signal.

[1] line through the baricenter perpendicular to the major axis

6.0.2 2D Step-Edge

A 2D Step-Edge is characterized by both strength and direction. Hence, we cannot use a directional derivative but instead we need an operator allowing to sense the edge whatever its direction is. For this purpose, we use the gradient:

$$\nabla I(x,y) = \frac{\partial I(x,y)}{\partial x} i + \frac{\partial I(x,y)}{\partial y} j$$

The direction of the gradient is that of the highest variation of the function, while its magnitude is the absolute value of the directional derivative along such direction.

6.1 Edge detection by Gradient Thresholding

Edge detection by gradient thresholding can be achieved by applying this scheme

6.1.1 Discrete approximation of the gradient

Backward or forward differences can be used:

$$\frac{\partial I(x,y)}{\partial x} \cong I_x(i,j) = I(i,j) - I(i,j-1), \quad \frac{\partial I(x,y)}{\partial y} \cong I_y(i,j) = I(i,j) - I(i-1,j)$$
$$\frac{\partial I(x,y)}{\partial x} \cong I_x(i,j) = I(i,j) - I(i,j+1), \quad \frac{\partial I(x,y)}{\partial y} \cong I_y(i,j) = I(i,j) - I(i+1,j)$$

which may be thought of as correlation by:

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

Or also central differences:

$$\frac{\partial I(x,y)}{\partial x} \cong I_x(i,j) = I(i,j+1) - I(i,j-1), \quad \frac{\partial I(x,y)}{\partial y} \cong I_y(i+1,j) = I(i,j) - I(i-1,j)$$

with the corresponding correlation kernels given by:

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

We can estimate the magintude by different approximations:

$$\|\nabla I\|_2 = \sqrt{I_x^2 + I_y^2} \quad \|\nabla I\|_1 = |I_x| + |I_y| \quad \|\nabla I\|_\infty = max(I_x, I_y)$$

The third approximation is faster and more invariant wrt edge direction.

However, it is well known that differetiation amplifies noise, making edge detection through differentiation problematic. It is therefore necessary to denoise images for edge detection, which does however lead to blurred edges that will be harder to detect.

6.2 Smooth Derivatives

Smoothing and differentiation are carried out jointly within a single step. This is achieved by computing differences of means. To avoid blurring edges, the two operations are carried out along orthogonal directions

$$\mu_x(i,j) = \frac{1}{3}[I(i,j-1) + I(i,j) + I(i,j+1)] \quad \mu_y(i,j) = \frac{1}{3}[I(i-1,j) + I(i,j) + I(i+1,j)]$$

$$\tilde{I}_x(i,j) = \mu_y(i,j+1) - \mu_y(i,j) = \frac{1}{3}[I(i-1,j+1) + I(i,j+1) + I(i+1,j+1) - I(i-1,j) - I(i,j) - I(i+1,j)]$$

$$\implies \text{ which is equivalent to the kernel } \frac{1}{3}\begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix}$$

and similarlty for $\tilde{I}_y(i,j)$ the kernel turns out to be:

$$\frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \end{bmatrix}$$

In the *Prewitt operator*, derivatives are approximated by central differences (better for diagonal edges)

$$\tilde{I}_x(i,j) = \mu_y(i,j+1) - \mu_y(i,j-1) \quad \Longrightarrow \quad \begin{bmatrix} -1 & 0 & -1 \\ -1 & 0 & -1 \\ -1 & 0 & -1 \end{bmatrix}$$

$$\tilde{I}_y(i,j) = \mu_x(i,j+1) - \mu_x(i,j-1) \quad \Longrightarrow \quad \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

In the *Sobel operator*, the central pixel is weighted more to further improve isotropy:

$$\mu_y(i,j) = \frac{1}{4}[I(i-1,j) + 2(i,j) + I(i+1,j)]$$
$$\tilde{I}_x(i,j) = \mu_y(i,j+1) - \mu_y(i,j-1)$$
$$\Longrightarrow \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Thresholding is suboptimal for edge localization, therefore we search for maxima of the gradient magnitude.

6.3 Non maxima suppression

Maxima are searched along the gradient direction. Pixels that are not maxima are suppressed

6.3.1 Discrete formulation

The gradient direction is discretized. 2π is divided into 8 direction bins

6.3.2 Continuous formulation

Is based on gradient interpolation

The overall flow-chart of an edge detector based on smooth derivatives and NMS ca be sketched as in the figure

6.4 Canny's Edge Detector

Canny proposed a set of criteria to evaluate an edge detector:

1. Good detection: robustness to noise
2. Good localization: edges should be localized with the minimum possible error (distance to the "true" edge)
3. One response to one edge: the fildter should detect one single edge point at each "true" edge

In the 1D case, the optimal edge detection consists in finding local extrema of the convolution of the signal by a 1st order Gaussian derivative (Gaussian: smoothing, derivative: variations). This theoretical result can also be regarded as proof that the gaussian filter is optimal for smoothing.

In the 2D case, the optimal filter results to be composed of the sequence of Gaussian smoothing, gradient computation and NMS along the gradient direction. 2D convolution by a gaussian can be sped up through separability:

$$\tilde{I}_x(x,y) = \frac{\partial}{\partial x}(I(x,y) * G(x,y)) = I(x,y) * \frac{\partial G(x,y)}{\partial x}$$
$$\tilde{I}_y(x,y) = \frac{\partial}{\partial y}(I(x,y) * G(x,y)) = I(x,y) * \frac{\partial G(x,y)}{\partial y}$$
$$G(x,y) = G(x)G(y)$$
$$\Longrightarrow \tilde{I}_x(x,y) = I(x,y) * (G'(x)G(y)) = (I(x,y) * G'(x)) * G(y)\tilde{I}_x(x,y) = I(x,y) * (G'(x)G(y)) = (I(x,y) * G'(x)) * G(y)$$

After NMS an extra thresholding is applied to distinguish true edges and unwanted ones. However, edge streaking may occur when gradient magintude varies along object contours. To address this issue, Canny proposed *hysteresis thresholding*: 2 thresholds are used, a higher (T_h) and a lower (T_l) one. Pixels are taken as an edge if they are above T_h or if they are higher than T_l and are neighbours to an already detected edge. The overall flow-chart of the Canny edge detector is as follows:

6.5 Edge detection through the second derivative

Edges may also be located by looking for zero-crossing of the second derivative of the signal. The second derivative along the gradient direction can be obtained as $n^T Hn$ where

$$n = \frac{\nabla I(x,y)}{\|\nabla I(x,y)\|} \quad \text{and} \quad H = \begin{bmatrix} \frac{\partial^2 I(x,y)}{\partial x^2} & \frac{\partial^2 I(x,y)}{\partial x \partial y} \\ \frac{\partial^2 I(x,y)}{\partial y \partial x} & \frac{\partial^2 I(x,y)}{\partial y^2} \end{bmatrix}$$

Computing this turns out to be very slow, so instead we can rely on the Laplacian as a second order differential operator

$$\nabla^2 I(x,y) = \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2} = I_{xx} + I_{yy}$$

6.5.1 Discrete Laplacian

One can use forward and backward differences to approximate the first and second order derivatives respectively:

$$I_x x \simeq I_x(i,j) - I_x(i,j-1) = I(i,j-1) - 2I(i,j) + I(i,j+1)$$

$I_y y$ analogous. This leads to the following convolution kernel:

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

It can be shown that the zero-crossing of the Laplacian typically lies close to those of the second order derivative along the gradient, while being much faster to compute.

6.5.2 Laplacian of Gaussian (LOG)

Gaussian smoothing is deployed along edge detection through the Laplacian, forming an edge detector referred to as LOG (Laplacian of Gaussian). Edge detection by the LOG can be summarized conceptually as:

1. Gaussian smoothing: $\tilde{I}(x,y) = I(x,y) * G(x,y)$
2. Second order differentiation by the Laplacian: $\nabla^2 \tilde{I}(x,y)$
3. Extraction of the zero-crossing of $\nabla^2 \tilde{I}(x,y)$

7 Finding Correspondences between images

A great variety of computer vision problems can be dealt with by finding *corresponding points* betweeen two or more images of a scene. The task of establishing correspondences is split into 3 successive steps:

1. Detection of salient points
2. Description - computation of a suitable descriptor based on a neighbourhood around a key point
3. Matching descriptors between images

and should be invariant to the transformations that may relate images.

7.0.1 Properties of good detectors/descriptors

- Detector
  - Repeatablity: finding the same salient points in different images
  - Saliency: finding useful points
- Descriptor
  - Distinctiveness vs robustness trade-off
  - Compactness: descriptions should be concise

Good keypoints should exhibit a large variation along all directions.

7.1 Moravec Corner Detector

$$C(p) = \min_{q \in n_8(p)} \|N(p) - N(q)\|^2$$

The cornerness at p is given by the minimum squared difference between the patch centered at p and those centered at its 8 neighbours. The cornerness of the whole image is computed, then thresholding and NMS are applied to identify corners.

7.2 Harris Corner Detector

The Harris corner detector is a continuous formulation of the Moravec corner detector. Denoting as $(\Delta x, \Delta y)$ a generic infinitesimal shift, the error function may be written as:

$$E(\Delta x, \Delta y) = \sum_{x,y} w(x,y)(I(x + \Delta x, y + \Delta y) - I(x,y))^2$$

Where $w(t)$ is 1 inside the chosen neighbourhood and 0 elsewhere Due to the shift being infinitesimal, we may take advantage of Taylor series expansion of the intensity function at (x, y):

$$I(x + \Delta x, y + \Delta y) - I(x,y) \simeq \frac{\partial I(x,y)}{\partial x}\Delta x + \frac{\partial I(x,y)}{\partial y}\Delta y = I_x(x,y)\Delta x + I_y(x,y)\Delta y$$

$$E(\Delta x, \Delta y) = \sum_{x,y} w(x,y)(I_x(x,y)\Delta x + I_y(x,y)\Delta y)^2 =$$

$$\sum_{x,y} w(x,y)(I_x(x,y)^2\Delta x^2 + I_y(x,y)^2\Delta y^2 + 2I_x(x,y)I_y(x,y)\Delta x\Delta y) =$$

$$= \sum_{x,y} w(x,y) \left(\begin{bmatrix} \Delta x \Delta y \end{bmatrix} \begin{pmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{pmatrix} \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \right)$$

$$E(\Delta x, \Delta y) = \begin{bmatrix} \Delta x & \Delta y \end{bmatrix} M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}$$

wlog we can assume M to be diagonal, as it is real and symmetric, thus always diagonalizable:

$$M = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \implies E(\Delta x, \Delta y) = \lambda_1 \Delta x^2 + \lambda_2 \Delta y^2$$

we have that if:

- $\lambda_1, \lambda_2 \simeq 0$: we are in a flat region
- $\lambda_1 >> \lambda_2$: we are on an edge
- $\lambda_1, \lambda_2 >> 0$: we are in a corner

The previous considerations also hold with M not diagonal by diagonalization:

$$M = R \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} R^T$$

where the columns of R are the orthonormal eigenvectors of M, and R^T is the rotation matrix that alignes the image axes to the eigenvectors of M Computing the eigenvalues of M may be slow, thus a more efficient cornereness function is proposed:

$$C = det(M) - k \cdot tr(M)^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)^2$$

With this new function we have

- $C \simeq 0$: Flat region
- $C > 0$: Corner
- $C < 0$: Edge

It is worth noting that the weighing function used is actually not a box function but rather Guassian shaped to assing more wight to closer pixels. Corner detection follows the usual steps of

- computing the cornerness C
- Thresholding the values of C
- perform NMS on the thresholded values of C

7.3 Shi-Tomasi Corner Detector

A popular variant to the Harris corner detector, providing better results when tracking corner points across video frames. The cornerness function is:

$$C = \min(\lambda_1, \lambda_2)$$

It results particularly useful in tracking corner points in videos.

7.4 Invariance properties of the Harris corner detector

7.4.1 Rotation

Yes, as the eignevalues of matrix M are invariant to rotation of the image axes

7.4.2 Intensity changes

- yes, to additive bias ($I' = I + b$) due to the use of derivatives
- no to a linear chang ($I' = aI$) as derivatives undergo the same change

so overall, no invariance to affine intensity changes.

7.4.3 Scale

No, due to fixed detection window size (impossible to detect homologous features when they appear at different scales in images). Images contain features at different scales, thsu a multiscale feature detector is necessary. A point may show up as salient at different scales. The characteristic scale of a feature is the scale at which it's more salient.