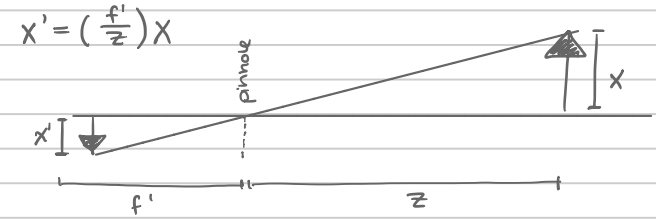
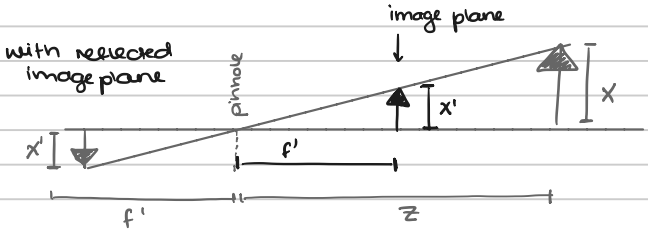
LEC1 - Image formation & basics

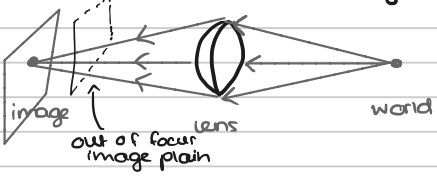
Camera obscura : pinhole mirrors upside-down but not left to right like mirror !



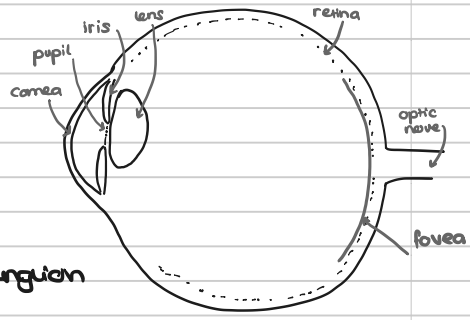


- distant objects are smaller
- Fluchtpunkt = vanishing point
  lies on the horizon
  parallel lines meet
  - each set of parallel lines meet at different vanishing point
- perspective projections:
  point → point
  line → line / point (degenerate cases)
  planes → images / line
  angles mostly not preserved

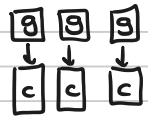
- pinhole camera limitations: too large → blurring, too small → diffraction → blur
  → no focus possible. + too dark images
  → invention of lenses: get more light & focus it



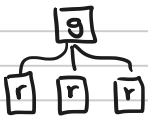
- lens : blurring if image plain out of focus
  today: lens systems ~ 15 elements
- human eye: ow vision affects a lot how we tackle ip problems:



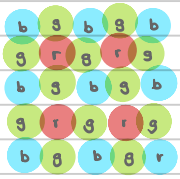
- iris like camera ; focusing by changing lens shape
- Retina: cones (Zapfen) : high resolution, precision
  rod (Stäbchen) : for low light, moves, MUCH smaller than cones
  Fovea: high resolution, mostly cones



- 1 ganglion pro cone, several rodes per ganglion
  c: every photon separately registered
  r: highly sensitive to motion



- 3 coin types respond to blue, green, red
- CCD cameras (→ Bayer filter) inspired by cones we perceive more green



Bayer filter
25% red / blue
50% green

- $I_c(x,y)$; $c = \{0 \dots 2\}$ $\quad I_c = \{I_0, I_1, I_2\} \hat{=}$ red, green, blue

$$I(x,y) = \begin{matrix} I(0,0) & \cdots & I(0,M) \\ \cdots & \ddots & \cdots \\ I(N,0) & \cdots & I(N,M) \end{matrix}$$

illumination

- Luminosity : amount of light, brightness

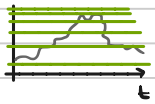
$$l : X \times Y \to \mathbb{R}^+ \qquad 0 < l(x,y) < +\infty$$

Reflectance $r : X \times Y \to (0,1) \qquad 0 \leq r(x,y) \leq 1$

$$I(x,y) = l(x,y) \cdot r(x,y)$$

↳ $I(x,y) \in \{0, \dots, +\infty\} \mathbb{R}^+$: enormous range! (and strong simplification)

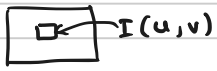
↳
- Quantization: to represent values of I e.g. 2 values per pixel $\in (0,1) \to 2^2$ possibilities 00 01 10 11
- Sampling: to map real-value signals to discrete space
  same dist between (0,1) and (1,2) ?

- Sampling:  high vs. low resolution, how many samples?
- Quantization  steps don't have to have same size! Sometimes makes sense to give all nighty dark pixel the same value

"how many

- Image values $\in \{0, 255\}$ 0 = black, 255 = white → 256 values
  color image: 3 channels $I = \{I_1, I_2, I_3\}$

- Tesselation: picture elements = pixels, how to separate space
  ⊞ vs ⊠ vs semiregular tesselation 
  → winkelsumme := 360°
  covers entire space!

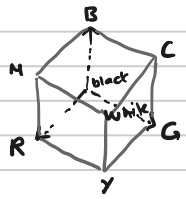


- Image model $I = l \cdot r$ exact value of $I(u,v)$?
  - 1. Integrate brightness over specified region.
    whole world: $p(x,y)$ is "everywhere" $p(x,0) = \begin{cases} 1 & \text{if } x,y \text{ are in pixel } u,v \\ 0 & \text{else} \end{cases}$
    ↑ Dirac function (δ) 'indicator function'

$$I(u,v) = \int_x \int_y l(x,y)\, p(x-u, y-v)\, dx dy$$

- visible range: 400 - 700 nm
  additive (projector) vs. subtractive (printer) color mixing
  ↑ RGB, black at beginning ↑ at beginning it's white, add all colors → black
  CMY

- RGB color model vs CMY color model $\in \{0, 255\}$



RGB → CMY with rotation
linear transformation

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1-R \\ 1-G \\ 1-B \end{bmatrix}$$

homogenous cube

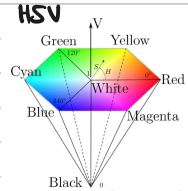


Fig 2.25: HSV hexagonal cone

- HSV: Hue, Saturation, Value: hexagon shaped, edges are pure colors, move inwards = mix colors
  more close to human perception
  = HSB; no linear transformation to RGB ~ H captures variations

- HSI: Hue, Saturation, Intensity = HSL; colon hexone / sphere
  nonlinear transformation to RGB

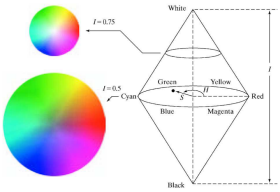
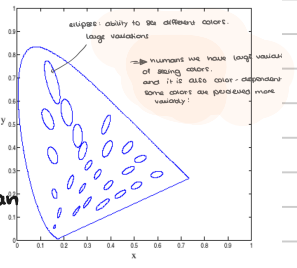


Fig 2.26: HSI Colour model

- CIE: white in middle.
  RGB is small part of it.

- Mc Adam ellipses: ability to differentiate colors is limited. / large variation among human and color-dependent



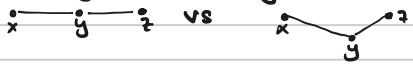
- Adaptation phenomena: color perception depends on spatial contrast & previously seen stuff (e.g. walk in on bright day)

↳ be cautious with colors in algos! Stop sign is red → but how to see it in shadow?

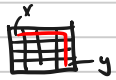
Distances

- can't be negative
- identity $d(x,x) = 0$
- symmetry $d(x,y) = d(y,x)$
- triangle inequality $d(x,z) \leq d(x,y) + d(y,z)$

} most important for sip



- 1-norm: "city block" $\sum_{i=1}^{n} |x_i - y_i|$



L^2-norm: euclidean norm $\sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$

L^p-norm: $(p \to \infty)$ $\lim_{p \to \infty} \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}} = \max_{1 \dots n} \{ |x_1 - y_1|, |x_2 - y_2|, \dots, |x_n - y_n| \}$

good where eucl. dist explodes

- Chamfer dist: (e.g. with L_1-norm) e.g.

0 1 0 0 0	I
1 0 1 2 3	D

  - build image such that $D(x,y) \in \{0, +\infty\}$ if value 0, it's 0 and else $+\infty$ (or a high number) for all $\forall x,y$

AL	AL	BR
AL	AL or BR	BR
AL	BR	BR

  - for $(x = 0 \dots N-1)$ left → right
    for $(y = 0 \dots M-1)$ top → bottom

$$D(x,y) = \min_{x,y \in AL} \{ D(x,y), D(x',y') + L^1 d((x,y),(x',y')) \}$$

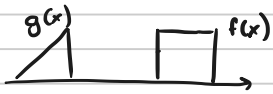
e.g. 0 was schon dort — dist. zu (x',y') + dessen wert

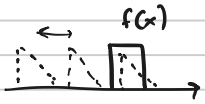
  - do the same for $x \in BR$

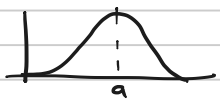
In words ~ go to every pixel, check above left of it, $O(\text{chamfer}) \sim N \cdot N \cdot 8$

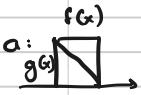
LEC 2 - Image formation & basics (2)

- f, d : 1D continuous functions
- convolution : $(f * g)(x) = \int_{-\infty}^{\infty} f(\tau)\, g(x-\tau)\, d\tau = \int f(x-\tau)\, g(\tau)\, d\tau$

e.g: 

$(f*g)x$ 



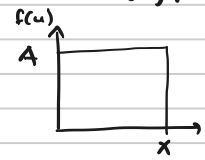
at a: 

- Fourier decomposition : $F(u) = \int f(x)\, e^{-2\pi i u x}\, dx = \frac{1}{N} \sum_{i=1}^{N-1} f(x)\, e^{-\frac{i 2\pi u x}{N}}$

$f(x) = \int F(u)\, e^{2\pi i u x}\, du$; f is continuous if it's integrable

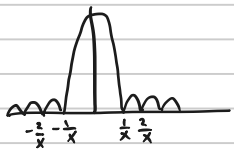
as $F(u) = \text{real}(u) + i\, \text{img}(u) = |F(u)| \cdot e^{i\phi(u)}$ $\quad \phi(u) = \tan^{-1} I(u)/R(u)$

$|F(u)| = \sqrt{R^2(u) + I^2(u)}$



$F(u) = \frac{A}{2\pi u} \sin(\pi u x)\, e^{-i\pi u x}$

$|F(u)| = Ax \left| \frac{\sin(\pi u x)}{\pi u x} \right|$



FFT algo in most processors, R, matlab, ...

- properties : $F(f+g) = F(f) + F(g)$ linearly separable

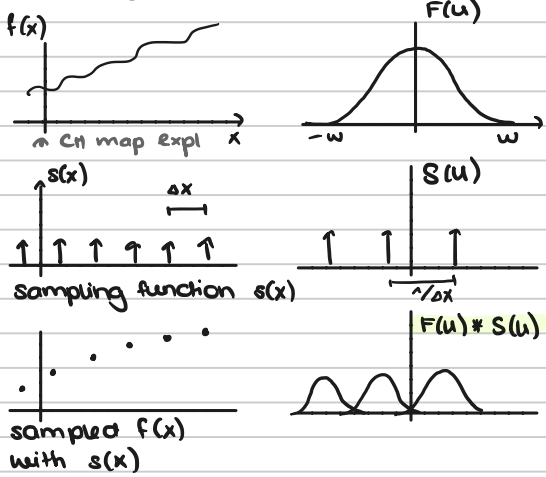
$F(f \cdot g) \neq F(f) \cdot F(g)$

$\forall a,b : a \cdot f(x) = a\, F(u)$

$f(a \cdot x) = \frac{1}{a} F(u/a)$

$f(x) * g(x) = F(u) \cdot F(v)$	important!
$f(x) \cdot g(x) = F(u) * F(v)$	

Sampling



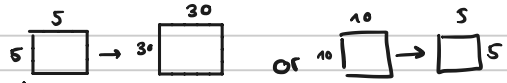
Aliasing : if $F(u) * S(u)$ overlap :

"duplication of signal"
c : center of overlap
every $\frac{1}{2\Delta x} = u$

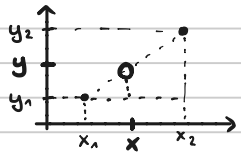
reduce overlap by getting more samples!
narrow sampling function $s(x) \rightarrow$ wide $S(u)$
$\rightarrow$ we can't shrink an image by taking every second pixel $\rightarrow$ aliasing!
characteristic errors: small phenomena look bigger, fast look slower
e.g. wheels roll wrong way in TV

Nyquist theorem : $\Delta x \leq \frac{1}{2w}$
to recover the signal

Interpolation: a form of sampling
up-/downsampling possible

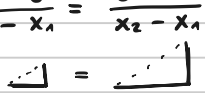


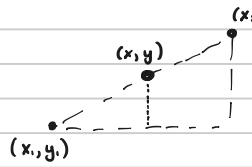
1D case:



$y = ax + b$

$$\frac{y - y_1}{x - x_1} = \frac{y_2 - y_1}{x_2 - x_1}$$

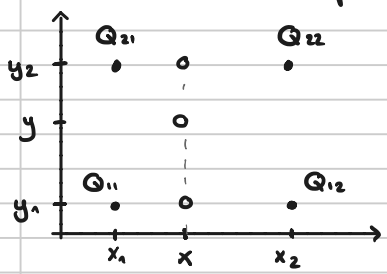




from triangle inequality

1D case is a linear operation, 2D not

2D case: Bilinear interpolation



① at height y_1: $f(x, y_1) = \frac{x - x_1}{x_2 - x_1} f(Q_{11}) + \frac{x_2 - x}{x_2 - x_1} f(Q_{12})$

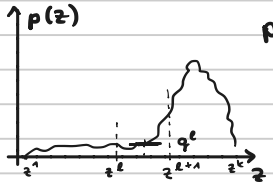
② at height y_2: $f(x, y_2) =$ " $f(Q_{21}) +$ " $f(Q_{22})$

③ interpolate in y-direction

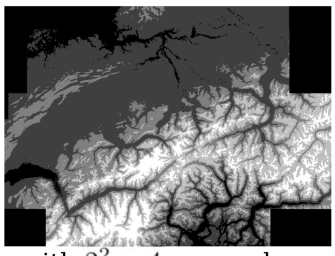
$$f(x, y) \simeq \frac{y_2 - y}{y_2 - y_1} f(x y_1) + \frac{y - y_1}{y_2 - y_1} f(x y_2)$$

for $x_2 - x_1 = 1 = y_2 - y_1$: $f(x, y) \simeq \begin{bmatrix} 1-x & x-1 \end{bmatrix} \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \begin{bmatrix} 1-y \\ y-1 \end{bmatrix}$

Quantization



prob. distribution of obs. value z. (left)
here: lots of high values → distribute bins accordingly! not fixed bins


with $2^2 = 4$ grey values

Lloyd-Max Quantization: fixed amount of bins K
what interval to take from z^ℓ to $z^{\ell+1}$, q^ℓ: average height

- reduce the average quantization error with error function (loss function):

$$E = \sum_{\ell=1}^{K} \int_{z^\ell}^{z^{\ell+1}} (z - q^\ell)^2 \, p(z) \, dz$$

- gradient descent: $\frac{\partial E}{\partial q^\ell} = -2 \int_{z^\ell}^{z^{\ell+1}} (z - q^\ell)^2 \, p(z) \, dz \overset{!}{=} 0$

$$q^\ell = \frac{\int_{z^\ell}^{z^{\ell+1}} z \, p(z) \, dz}{\int_{z^\ell}^{z^{\ell+1}} p(z) \, dz}$$

$$\frac{\partial E}{\partial z^\ell} \overset{!}{=} 0 \iff z^\ell = \frac{z^\ell - z^{\ell+1}}{2}$$

Idea: first optimize q^ℓ with random z^ℓ, then optimise z^ℓ
loop until no change

Median cut algo for color images (3D, not possible for Lloyd-Max)

LEC 3 : Linear filtering

correlation: $I'(x,y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I(x+i, y+j)$

convolution: $I'(x,y) =$ " $F(-i,-j)$ " "flipping"
$=$ " " $I(x-i, y-j)$

if $F(X,Y) = F(-X,Y) \rightarrow$ corr = conv (symmetric filter)

$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$ → no change $\quad \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$ → slide all pixels to the left $\quad \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$ → average filtering = blurring

$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ → sharpening filter "laplacian"

box filter: all weights equal, can show "lines" = artefacts
gaussian smoothing: no artefacts, looks like a fuzzy blob (what happens in reality)

1D: $\mathcal{N}(\mu, \sigma) = \frac{1}{z} \exp\left(-\frac{1}{2\sigma}(x-\mu)^2\right)$

2D: $\mathcal{N}(x, \mu, \Sigma) = 1/z \exp\left(-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)\right)$

Σ: covariance matrix $\in \mathbb{R}^{2\times 2}$ if $\Sigma = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}$ → norm to 1

μ doesn't matter (→ normalize all filters to 1!), but σ does! ⊞ or ▦ ?
$\sigma = 1$ pixel → ▦

Efficiency: ▦ * img = (▯ * img) * ▭ 9 vs. 6 multiplications!

dot product: $\langle a, b \rangle = a \cdot b = a_1 b_1 + a_2 b_2 + \dots + a_n b_n = |a||b| \cos\varphi$
φ is the similarity, measure by norm to 1 a and b
applying a filter can be seen as taking dot product → find structures

→ this filter finds circles

NCC: Normalized cross-correlation $NCC(I,T) = \frac{1}{n} \sum_{x,y} \frac{(I(x,y) - \bar{I})(T(x,y) - \bar{T})}{\sigma_I \sigma_T}$
$\bar{I}$: mean of img, $\bar{T}$: mean of template
σ_I, σ_T: stdev of intensities → scale to $\in [-1, 1]$ (remove mean: all at 0)
- no longer linear! $\sigma^2 = \frac{1}{n}(\sum x^2) - \mu^2$
- intensity invariant (find face in highly illuminated pic is no problem)

size of filter matters → change img (not filter) size
first filter coarse scales (grob) → then refine, have different blur amounts

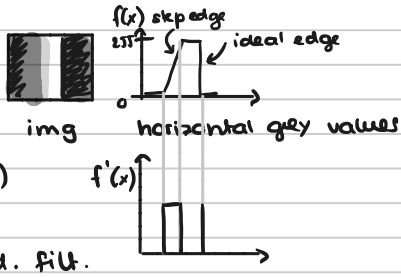
Gaussian pyramid: create each level from previous one: smooth & sample
gaussians are low pass filters

Summary: linear filtering: Form a new image whose pixels are a weighted sum of original pixel values
e.g. box filter / gaussian filter smoothing, searching for a template
properties: output is a shift-invariant function of the input (same at each img location)
pyramid representations: important for describing and searching an img at all scales.

smooth to avoid aliasing! (→ vgl. aliasing)

LEC 4 - Edges, corners & lines

Edges contain most shape information!



- derivatives & edges: detect changes (edges) with derivatives! $\frac{df}{dx} \simeq [f(x+1) - f(x-1)]/2 \simeq f(x+1) - f(x-1)$
  - gradient vector: $\left[\frac{\partial f}{\partial x} \quad \frac{\partial f}{\partial y}\right]$
  - $[1 \; 0 \; -1]$ and $\begin{bmatrix}1\\0\\-1\end{bmatrix}$ horizontal / vertical edge d. filt.
  - combined with smoothing: Prewitt filters: $P_x = \begin{bmatrix}-1&0&1\\-1&0&1\\-1&0&1\end{bmatrix}$ $P_y = \begin{bmatrix}-1&-1&-1\\0&0&0\\1&1&1\end{bmatrix}$
  - magnitude (stärke) of gradient: $\sqrt{p_x^2 + p_y^2}$ or $\max\{|p_x|, |p_y|\}$
- causes for edges: andere Tiefe (vorder- Hintergrund), 3D shape, reflectance change (e.g. surface material change), illumination discontinuity (shadow)
- computational model: $[-1 \quad 1]$ is a convolution (approximation of $\frac{\partial f}{\partial x}$)
- noise: is detected by edge filters → smooth before finding edges!
  edge is location with high gradient, 2 derivatives: x and y direction
  we can use derivatives of gaussian filters because:
  $D * (G * I) = (D * G) * I$ D: derivative, G: gauss smooth, I: img

Prewitt

1	0	-1
1	0	-1
1	0	-1

Sobel

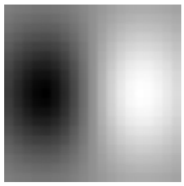
1	0	-1
2	0	-2
1	0	-1

Roberts Cross

-1	0
0	1

Laplacian

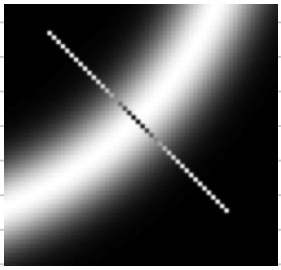
0	-1	0
-1	4	-1
0	-1	0



derrivative of Gauss in x direction

- Gradient magnitude: $I(x,y)$: img
  $I_x(x,y)$: der. in x direction $= I_x$
  $[I_x \; I_y]$: vector gradient
  $\sqrt{I_x^2 + I_y^2}$ = gradient magnitude

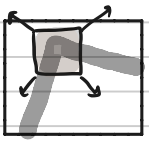
- more smoothing → eliminates noise edges, makes edges smoother & thicker, removes fine detail



- canny edge detection:
  1.) Apply derivative of gaussian $(D*G)*I \rightarrow I_x, I_y$
  2.) compute magnitude of gradients $\sqrt{I_x^2 + I_y^2}$
  3.) Non-maximum suppression: thin multi-pixel-wide 'ridge' (= kante) down to single pixel
  4.) link & threshold:
    - low and high edge thresholds → accept ove high-t also accept low th. ones that are connected to high threshold!
      ↳ Track edge by hysteresis

- hysteresis: "depending on its history", a lag or momentum factor
  idea: 2 thresholds: k_{high} and k_{low}
  use k_{high} to find strong edges to start edge chain
  use k_{low} to find weak edges which continue edge chain
  typical threshold ratio $k_{high} / k_{low} = 2$
  → closes gaps! — — ⇒ ——

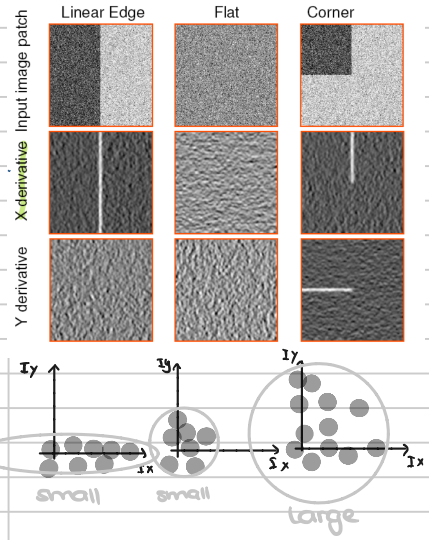
corners: edge detectors perform poorly at corners, but are important to detect
- at corners, gradient is ill defined, but around corner: 2 directions

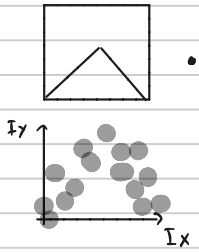


- Harris corner: shuffle ▢ a bit around and check changes
  ▢ flat region: no change ▯ edge: change in 1 direction, no change in "edge" direction. ◱ corner: change in all directions

$$SSD(u,v) = \sum_{x,y} I(x+u, y+v) - I(x,y))^2 = [u\ v]\begin{bmatrix} \Sigma I_x^2 & \Sigma I_x I_y \\ \Sigma I_x I_y & \Sigma I_y^2 \end{bmatrix}\begin{bmatrix} u \\ v \end{bmatrix}$$

covariance matrix





- corner detection must be rotation invariant! → Eigenvalues
  "principal component ellipse ⬭ which is only large at edges
  ↳ data → center data around 0 → compute covariance = inner product
- Eigenvalues tell us "changes" in data, the "directions"

- Eigenvalue problem: λ: Skalar = Eigenvalue
  $\vec{v}$: Eigenvector, C: Matrix

$$C\vec{v} = \lambda\vec{v} \iff \| C - \lambda \mathbb{1} \| = 0 \qquad \mathbb{1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \dim(C) = 3,3$$

$$\left\| \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} - \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \right\| = 0 = \left\| \begin{bmatrix} C_{11}-\lambda_1 & C_{12} \\ C_{21} & C_{22}-\lambda_2 \end{bmatrix} \right\| = 0$$

solve for λ_1, λ_2 and get $\vec{v}_1$ and $\vec{v}_2$
⇒ Eigenvalues not affected by rotation! corner: λ_1, λ_2 are both large!

- computational considerations: $R = \det M - k(\text{trace } M)^2$ M: covariance mat
  det is connected to eigenvalues
  - detect flat regions: below treshold: edge, within: flat, above: corner ?

Harris Corner Detection summary

convolution, point product is fast

1. Compute x and y derivatives of image
   compute gradients
   conv
$$I_x = G_\sigma^x * I \qquad I_y = G_\sigma^y * I$$
   gaussian smoothed edges, σ: how much smoothing?
   "compute covariance matrix"
2. Compute product of derivatives at each pixel
   use I_x, I_y from step 1 and get
   point product
$$I_{x^2} = I_x I_x \qquad I_{xy} = I_x I_y \qquad I_{y^2} = I_y I_y$$
   is an image
3. Compute sums of products of derivatives at each pixel
   conv
$$S_{x^2} = 1_{|I|} * I_{x^2} \qquad S_{xy} = 1_{|I|} * I_{xy} \qquad S_{y^2} = 1_{|I|} * I_{y^2}$$
   $I_{x^2} * \begin{bmatrix} 1 & \cdots & 1 \\ \vdots & & \vdots \\ 1 & \cdots & 1 \end{bmatrix} = S_{x^2}$ filter of values 1 of size t
4. Define at each pixel the matrix
   slow
$$H(x,y) = \begin{bmatrix} S_{x^2}(x,y) & S_{xy}(x,y) \\ S_{xy}(x,y) & S_{y^2}(x,y) \end{bmatrix}$$
5. Compute the response of the detector at each location
$$R = Det(H) - k(Trace(H))^2$$
   values on the diagonal
6. Threshold on the value of R. Compute non-maximum suppression.
   depending on whether you want to detect flats, edges or corners