Convolution and Filtering

Correlation: $I' = K \circ I$, $I'(x,y) = \sum_{(i,j)\in N(x,y)} K(i,j) \circ I(x+i, y+j)$

- used for template matching (bias to bright areas → norm. mean)
- $I'(x,y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} K(i,j)\, I(x+i, y+j)$

Convolution: $I' = K * I$, $I'(x,y) = \sum_{(i,j)\in N(x,y)} K(i,j)\, I(x-i, y-j)$
associative

- If K symmetrisch, conv. = corr.
- $I'(x,y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} K(-i,-j)\, I(x+i, y+j)$

Near the edge:

- Filter window falls off image
- Extrapolation (clip filter, wrap around, copy edge, reflect across edge)

Gaussian Kernel:

- Weight contribution of neighboring pixels by nearness
- $G_\sigma = (2\pi\sigma^2)^{-1} \exp\left(\frac{-(x^2+y^2)}{2\sigma^2}\right)$
- separable: $g(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-x^2}{2\sigma^2}\right) \cdot \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-y^2}{2\sigma^2}\right) = g(x)\cdot g(y)$
- advantages:
  - rotationally symmetric
  - Single lobe (neighbors influence decreases monotonically)
  - Still one lobe in freq. domain (no corruption from high freq.)
  - simple relationship to σ
  - Easy to implement efficiently

Morphological operation:

- Fit: $\{y : y = x+s, s \in S\} \subset I$ ($\forall s \in S$: match)
- Hit: $\{y : y = x-s, s \in S\} \cap I \neq \emptyset$ ($\exists s \in S$: match)
- Miss: $\{y : y = x-s, s \in S\} \cap I \neq \emptyset$ ($\neg\exists s \in S$: match)
- Erosion: $E = I \ominus S = \{x : x+s \in I \ \forall s \in S\}$ $E(x) = \begin{cases} 1 & \text{if } S \text{ fits } I \text{ at } x \\ 0 & \text{otherwise} \end{cases}$
- Dilation: $D = I \oplus S = \{x : x-s, y \in I \text{ and } s \in S\}$ $D(x) = \begin{cases} 1 & \text{if } S \text{ hits } I \text{ at } x \\ 0 & \text{otherwise} \end{cases}$
- Opening: $I \circ S = (I \ominus S) \oplus S$ $\quad (I \circ S)^c = I^c \bullet S$
- Closing: $I \bullet S = (I \oplus S) \ominus S$

- thinning: $I \otimes S = I \setminus (I \circledast S)$ • thickening: $I \odot S = I \cup (I \circledast S)$

Image Filtering (linear): $g = f \otimes h$

$$g(i,j) = \sum f(i+k, j+l)\, h(k,l)$$

Linear: $L(\alpha I_1 + \beta I_2) = \alpha L(I_1) + \beta L(I_2)$

Example Filters:

blur (box)

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

sharpen

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Gaussian (low-pass) (blurring / smoothing)

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

differential (edge detection)

Prewitt

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

Sobel

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

high-pass (sharpen)

laplacian

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

high-pass

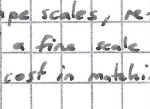
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

- gradient vs. laplacian:
  - Laplacian is isotropic
  - Laplacian gives more noisy results (2nd order derivative)
- filter is separable if $K(m,n) = f(m)\, g(n)$

Image Features

Edge detection:

- detect local gradient $|\text{grad}(f(x,y))| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$
- differential filters in x and y direction
- detect discontinuities by considering 2nd derivative
- sensitive to fine details and noise → blur first



Canny edge detector:

1. smooth img with gaussian filter
2. Find magnitude and orientation of gradient (Prewitt, Sobel)
   $M(x,y) = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$ $\quad \alpha(x,y) = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$
3. Use non-maximal suppression on gradient magnitude img (thin edges)
4. Double thresholding to detect strong and weak edge pixels. Strong: $M(x,y) \geq \theta_{high}$, weak: $\theta_{high} > M(x,y) \geq \theta_{low}$
5. Reject weak edge pixels not connected to strong ones.

Hough Transform: $\|\vec{x} - s\| - r = 0$ detect circles

Fit straight line to set of edge pixels (also circles)

$y = mx + c$ or $\rho = x\cos(\theta) + y\sin(\theta)$ $\quad 90° \Rightarrow (m \to \infty)$

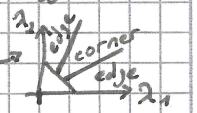
Geometric resolution: # pixels / area

Radiometric resolution: # bits / pixel

Keypoint detection:

- maximize eigenvalues of M in window w

$$M = \begin{pmatrix} \sum f_x^2(x,y) & \sum f_x(x,y) f_y(x,y) \\ \sum f_x(x,y) f_y(x,y) & \sum f_y^2(x,y) \end{pmatrix} \rightsquigarrow$$



$$C(x,y) = \det(M) - k\,(\text{trace}(M))^2 = \lambda_1\lambda_2 - k(\lambda_1 + \lambda_2)$$

- local maxima of cornerness → key point
- Gaussian weighting function to make central pixels more important

Harris corner detection:

- Invariant to brightness offset
- Invariant to shift and rotation (EW don't depend on rotation of the ellipses.)
- Not invariant to scaling (cause fixed-size window for neighborhood)

Image Pyramid

- search for correspondance: coarse scales, refine with finer
- edge-tracking: "good" edge at a fine scale has parent on coarse scale
- control of detail and comp. cost in matching (finding stripes → texture repr.)

Gaussian: $g^2 = g$, low-pass → redundant repr.

Laplacian: Δ of upsampled and normal gaussian pyramid level → bandpass, each lvl has freq., unrepresented at other lvls

Pyramid repeated smoothing and down sampling.

Skeletization:

$$S_n(X) = (X \ominus_n B) \setminus [(X \ominus_n B) \circ B]$$

$$S(X) = \bigcup_{n=1}^{\infty} S_n(X),$$

$$X = \bigcup_{n=0}^{\infty} S_n(X) \oplus_n B \quad \text{(but not from } S(X)\text{)}$$

Problems:

- Def of max disc is poorly defined on a digital grid
- Sensitive to noise on the boundary

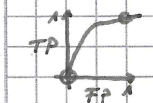
Image Segmentation

Thresholding: $B(x,y) = \begin{cases} 1 & \text{if } I(x,y) \geq T \\ 0 & \text{otherwise} \end{cases}$

Chroma keying: $I_\alpha = |I - g| > T$, $T \approx 20$ e.g., $g = (0\ 255\ 0)$

↑ variation is NOT the same in all 3 channels

ROC: characterizes performance of binary classifier



TP = True positives / # positives (P) sensitivity

FP = False positives / # negatives (N) specificity

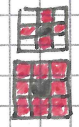
$N = FP + TN \quad P = TP + FN$

The closer to 45° → less accurate, better if closer to (0,1)

→ choose point on ROC curve with gradient

$\beta = \frac{N}{P} \frac{(V_{TN} + C_{FP})}{(V_{TP} + C_{FN})}$, often $V_{TN} = V_{TP} = 0$ v: value c: cost

Pixel connectivity:



: 4-Neighborhood

: 8-Neighborhood

Connected components labelling:

"labels each connected component of a binary img. with a different number"

① 1. Iterate through each element
2. If ≠ background-element
i.) get neighboring element
ii.) if ∄ neighbors → new unique label
↳ otherwise → assign label of smallest neighbor
iii.) store equivalence between neighboring label

② 1. Iterate through each element
2. If the element is not background, relabel with the lowest equivalent label.

Fourier Transform

Aliasing: can't shrink img by taking every 2nd pixel

$$F(g)(u,v) = \iint_{\mathbb{R}^2} g(x,y) \exp(-i2\pi(ux + vy))\, dx\, dy$$

$$F(g)(u,v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} g(x,y) \exp\left(-i2\pi\left(\frac{ux+vy}{N}\right)\right)$$

$e^{i\varphi} = \cos\varphi + i \sin\varphi$

Phase: Basis dominance

Magnitude: Basis occurence

$\cos(iy) = \frac{e^{-y} + e^{y}}{2i}$, $\cos(y) = \frac{e^{iy} + e^{-iy}}{2}$, $\sin(iy) = \frac{e^{-y} - e^{y}}{2i}$

$\sin(y) = \frac{e^{iy} - e^{-iy}}{2i}$, $\iint_{\mathbb{R}^2} e^{-i2\pi(ax+by)} - e^{-i2\pi(ax+by)}\, dx\, dy = \delta(a,b) - \delta(a,b)$

If img corrupted with $g(x,y)$ design band-pass at (u_0, v_0) to remove

$$F \cdot G = \mathcal{F}(f * g) \text{ and } F * G = \mathcal{F}(f \cdot g)$$

Nyquist sampling theorem: $\omega_s \geq 2\omega$

Optical Flow

While = 1, Black = 0

Apparent motion of brightness patterns (tracking, motion segmentation, stabilization, compression, ...)

+ small motion + spatial coherence

Brightness constancy: for small Δt brightness doesn't change

- $I(x,y,t-1) = I(x,y,t-1) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + e$
- with small δt: $I_x \frac{dx}{dt} + I_y \frac{dy}{dt} + I_t \approx 0$

Aperture problem:

The motion direction of a contour is ambiguous, cause the motion component parallel to the line cannot be inferred based on visual input.

Horn and Schunk: Use global smoothness to solve above

$$e_s = \iint ((u_x^2 + u_y^2) + (v_x^2 + v_y^2))\, dx\, dy$$
$$e_c = \iint (I_x u + I_y v + I_t)^2\, dx\, dy$$
$\Big\}$ minimize $(e_s + \lambda e_c)$

Lucas-Kanade: Assume single velocity

$$M\vec{u} = \underbrace{\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}}_{A^TA} \begin{bmatrix} u \\ v \end{bmatrix} = -\underbrace{\begin{bmatrix} \sum I_x t \\ \sum I_y t \end{bmatrix}}_{A^Tb}$$

Solvable when:
- A^TA is invertible
- A^TA not too small (λ_1, λ_2 not too small)
- A^TA well conditioned (λ_1 / λ_2 not too large)

Corners and textured areas OK

Iterative refinement

Problems:
- large motion (> 1 px): Coarse-fine estimation, Exhaustive neighborhood search
- Point doesn't move like neighbors: motion segment
- Brightness constancy doesn't hold: Exhaustive neighborhood search with normalized correlation.

Coarse-to-fine: Build pyramid and do H/S or LK on each layer to get rid of distance constraint.

Parametric Motion Model: Global (non local) motion model

- 2D Models: Translation, Affine, Quadratic, Planar projective transform
- 3D Models: Instantaneous camera motion models, Homography + epipole, Plane + Parallax

- $E(\text{Transform}) = \sum [I(\text{Transform}) - I_0(x)]^2$ → least squares
- Translation: $x + \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$ Affine: $\begin{bmatrix} a & b \\ c & d \end{bmatrix} x + \begin{bmatrix} \delta x \\ \delta y \end{bmatrix}$, $u = a_1 + a_2 x + a_3 y$, $v = a_4 + a_5 x + a_6 y$

$$Err(\vec{a}) = \sum [I_x(u) + I_y(u) + I_t]^2$$

- Quadratic: instant approximation to planar motion,
$u = q_1 + q_2 x + q_3 y + q_4 x^2 + q_5 xy$
$v = q_6 + q_7 x + q_8 y + q_9 xy + q_{10} y^2$

- Projective: exact planar motion, $u = x' - x$, $v = y' - y$
$x' = \frac{h_1 + h_2 x + h_3 y}{h_7 + h_8 x + h_9 y}$, $y' = \frac{h_4 + h_5 x + h_6 y}{h_7 + h_8 x + h_9 y}$

- Instantaneous camera motion: $Z(x,y)$: only local parameter
$u: -xy\Omega_x + (1+x^2)\Omega_y - y\Omega_z + (T_x - T_z x)/Z$
$v: -(1+y^2)\Omega_x + xy\Omega_y - x\Omega_z + (T_y - T_z x)/Z$

- Homography + Epipole: Same as Projective with
$x' = \dots + \frac{\gamma t_1}{\gamma t_3}$ and $y' = \dots + \frac{\gamma t_1}{\gamma t_3}$

- Plane + Parallax: $u = x^w - x = \frac{\gamma}{1 + \gamma t_3}(t_3 x - t_1)$
$v = y^w - y = \frac{\gamma}{1 + \gamma t_3}(t_3 y - t_2)$

Correlation and SSD: For large displacements do template matching

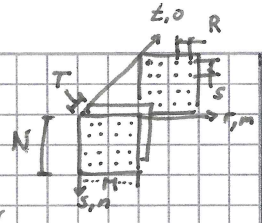
Bayesian Flow: Add uncertainty model to LK tracking (Rhombus)

Phong reflection model:

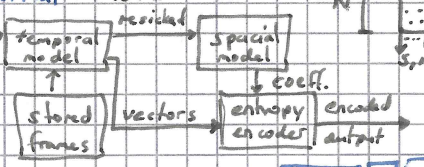
$$I = I_a k_a + I_d (k_d (N \cdot L) + k_s (R \cdot V)^\alpha)$$

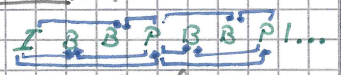
Position $x(t)$, $\dot{x}(t) = \frac{p(t)}{m}$
Momentum $p(t)$, $\dot{p}(t) = F_i(t)$
Rotation $q(t)$, $\dot{q}(t) = \frac{1}{2}(I^{-1}(t)L(t))\,q(t)$
Angular momentum $L(t)$, $\dot{L}(t) = (r_i(t) - x(t)) \times F_i(t)$

Video Compression



Video Format: 2D+t





Frames:

- I-Frame: Intracoded, independent of others
- P-Frame: Previously, dependent of prev. I or P frame
- B-Frame: Bi-directional, dep. of prev. and following I or P.
- Take advantage of temporal redundancy
- predict on previously coded frames
- Ineffective if many changes / high motion

If temporal redundancy fails:

Block-matching motion estimation:

Partition each frame into blocks, describe their motion
↳ assume same motion within block
+ no object identification required
− Produces blocking artefacts − assumes translational motion
∀ blocks search for best matching block in refer. frame with (MAE (Mean absolute error), MSE (mean squared err.))
→ Full search: check all candidate blocks
→ Partial search: Check carefully selected subset

Summary:

- RGB → YUV: small color differences not noticeable, brightness is more important
- temporal redundancy: use MC and ME, drop B
- spacial redundancy: use DCT and get rid of high-freq

Textures

Chaos mosaic:

Best for random looking textures (vegetation, hay)

1. tile input img
2. pick random blocks and place in random location
3. smooth edges

Doesn't work for structured textures (brick wall)

Efros & Heeng:

- to synthesize new pixel search input img for similar neighborhood, pick match at random
- Faster: combine with blocks and synthesize blocks, not pixels.

Transformations

Rotation about an arbitrary axis: CW

$$\begin{bmatrix} \cos\varphi + u_x^2(1-\cos\varphi) & u_xu_y(1-\cos\varphi) - u_z\sin\varphi & u_xu_z(1-\cos\varphi) + u_y\sin\varphi \\ u_yu_x(1-\cos\varphi) + u_z\sin\varphi & \cos\varphi + u_y^2(1-\cos\varphi) & u_yu_z(1-\cos\varphi) - u_x\sin\varphi \\ u_zu_x(1-\cos\varphi) - u_y\sin\varphi & u_zu_y(1-\cos\varphi) + u_x\sin\varphi & \cos\varphi + u_z^2(1-\cos\varphi) \end{bmatrix}$$

Rotation 2D: $\begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}$ CCW $\quad R^{-1}(\varphi) = R^T(\varphi) = R(-\varphi)$

Taylor theorem: All maps can be approximated as linear maps over short distance

Linear: $f(au+bv) = af(u) + bf(v)$

Reflection: $\begin{pmatrix} a & 0 \\ 0 & -b \end{pmatrix}$ Shear: $\begin{pmatrix} 1 & a \\ b & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & d_y & d_z & 0 \\ \cdot & 1 & \cdot & d_x \\ \cdot & \cdot & 1 & \cdot \\ \cdot & \cdot & \cdot & 1 \end{pmatrix}$

Translation (affine, except homogene): $\begin{pmatrix} 1 & 0 & b_1 \\ 0 & 1 & b_2 \\ 0 & 0 & 1 \end{pmatrix}$ $\begin{pmatrix} 1 & \cdot & \cdot & b_x \\ \cdot & 1 & \cdot & b_y \\ \cdot & \cdot & 1 & b_z \\ 0 & \cdot & \cdot & 1 \end{pmatrix}$

Basiswechsel: $f(x_n) = [b_1 b_2 \dots b_n]\,x_n$

Non-uniform scale: $\begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix}$, $\begin{pmatrix} s_x & \cdots & 0 \\ \vdots & s_y & \vdots \\ 0 & \cdots & s_z \end{pmatrix}$

Homogenous: $x_c = x_h / w_h$, $y_c = y_h / w_h$ Scale invariant

$[x_h, y_h, w_h]^T$ or $[x_h, y_h, z_h, w_h]^T$

Arbitrary axis reflection:

1. Translate reflection line L to origin
2. Rotate L to allign with x
3. Reflect about x-axis
4. Rotate back
5. Translate back.

Rotation about an not arbitrary point: CCW

$$\begin{pmatrix} \cos\varphi & -\sin\varphi & x_0(1-\cos\varphi) + y_0\sin\varphi \\ \sin\varphi & \cos\varphi & y_0(1-\cos\varphi) - x_0\sin\varphi \\ 0 & 0 & 1 \end{pmatrix}$$

Rigid transformation: Trans. + Rot. (dist. preserving)

Similarity transformation: Trans. + rot. + unif. scale (angle preserv.)

Affine transformation: trans. + rot. + scale + shear (parallel pres.)

Z-Buffer algorithm:

1. Set depth $(x,y) = 1.0$, intensity(x,y) = background color
2. For each polygon, find all pixel covered
3. Calc. $z(x,y)$ of each pixel covered by the polygon
4. If $z(x,y) <$ depth (x,y), polygon is closer
   set depth $(x,y) = z(x,y)$, change color.

Radiosity: Emitted + reflected light

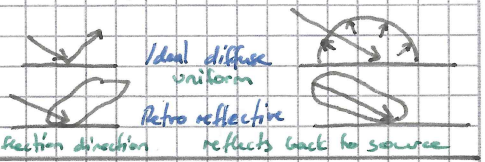
$$\underset{\text{radiosity}}{B_i} = \underset{\text{emission}}{E_i} + \underset{\text{reflectivity}}{\rho_i} \sum_1^n \underset{\text{formfactor from } i \text{ to } j}{B_j F_{ij}}$$

Color / Light:

Visible 380 to 740 nm

Freq. determines color of light: $f = \frac{v}{\lambda}$, $v = 3 \cdot 10^8$ m/s

RGB (additiv) light — CMYK (subtractive) printcolor



Auto correlation matrix

$$R_{cc} = E[\vec{c}\vec{c}^H] = E[A\vec{f}\cdot\vec{f}^H A^H] = A R_{ff} A^H$$

$$E[c_i^2] = [R_{cc}]_{i,i} = [A R_{ff} A^H]_{i,i}$$

$$R_{ff} = E[f_i \cdot f^H] = \frac{F \cdot F^H}{n}, \quad F = [f_1 \dots f_n]$$ image collection

$$\vec{c} = A\vec{f}, \quad \|\vec{c}\|^2 = \|\vec{f}\|^2, \quad f^{m \times n}, A^{m \times m}, A^{-1} = A^H$$

$R_{ff} V = V\Lambda$, V is unitary $\quad A = V$, $R_{cc} = \Lambda$

Objects

Implicit representation (Algebraic surfaces, fractals) $(3x^2+4y^2+2z^2=3xy)$:
- + compact
- + easy to determine a points whereabouts
- + for simple shapes, exact descriptions / no sampling error
- + easy to handle changes in topology
- − expensive to find all all points in the shape
- − very difficult to model complex shapes

Explicit representation (Triangle meshes, point clouds)
- List of points:
  - + easy to represent any kind of geometry
  - + useful for large datasets ($\gg 1$ point / pixel)
  - − Difficult to draw in undersampled regions
  - − Hard to do processing / simulation
- List of vertices and polygons (triangles or quads):
  - + Easier to do processing / simulation, adaptive sampling
  - − more complicated datastructures

Occlusion using depth buffer
- Store one depth value per coverage sample (not per pixel)
- Constant space per sample → constant space for depth buffer
- Occlusion test per covered sample $\mathcal{O}(1)$
- Only requires that surface depth can be evaluated at a screen sample point

Transparency:
Add another channel α for opacity (not commutative)
$C = \alpha_B B + (1-\alpha_B)\alpha_A A \quad \alpha_C = \alpha_B + (1-\alpha_B)\alpha_A$
Rendering back to front (not always possible)

Rendering a mixture of opaque and transparent triangles
1. Render opaque surfaces using depth-buffered occlusion (if depth test passed, triangle overwrites value in color buffer at sample)
2. Disable depth buffer update, render semi-transparent surfaces in back-to-front order. If depth test passed, triangle is composed over contents of color buffer at sample)

Raycasting vs. Rasterization

Rasterization:
- Proceeds in triangle order
- Most processing is biased on 2D primitives (3D geometry projected into screenspace)
- Store depth buffer (random access to regular structure of fixed size)

Raycasting:
- Proceeds in screen sample order (no depth buffer, front-to-back or back-to-front)
- Must store entire scene

↳ reordering of loops and math in 3D

Shortest path to object (Lagrangian multiplier)

1. $f(\underline{x})$ = path — point → sphere, line → tube, plane → plane
   $g(\underline{x})$ = object
2. $\mathcal{L} = f - \lambda g, \quad \mathcal{L}' = 0$

Interpolation

Spline of higher degree can lead to oscillation on the edges
- Interpolation → exactly
- Continuity → at least 2 times differentiable everywhere
- Locality → moving one control point doesn't affect whole curve

	I	C	L
Natural	✓	✓	X
Hermite	✓	X	✓
B-spline	X	✓	✓

Laplace

∇ - Divergence of gradient, trace of hessian, sum of f'', "average" curvature

Divergence: $\int_{\partial\Omega} n \nabla\varphi = \int_{\Omega} \nabla\cdot\nabla\varphi = \int_{\Omega} \Delta\varphi = 0$

Hessian matrix $\begin{bmatrix} \frac{\partial^2 f}{\partial x_1 x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n x_1} & \cdots & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$

Gradient descent: $x_{k+1} = x_k - \nabla f_0^{-1}(x_k)$

Newton method:
- + Great for convex problems
- − Nonconvex optimization is a BLACK ART

Forward euler: $q_{k+1} = q_k + \nabla f(q_k)$
- not very stable
- unconditionally stable for linear ODE's

Symplectic euler:
- + easy to implement, very often used in practice

Numerical iterators: stability, accuracy, symmetry, consistency / convergence, conservation, comp. efficiency

Particle systems: easy to understand, simple equation for each particle, easy to scale up

Eulerian: fast, regular computation, easy to represent, simulated in grid "trapped", grid causes blur, need PDE's

Lagrangian: easy concept, resolution/domain not limited by grid, good particle distribution can be tough, expensive finding neighbors

Solving the denoising problem:

$$\min_{\alpha_i,\, D\in\mathcal{C}} \sum_{i=1}^{n} \underbrace{\frac{1}{2}\|x_i - D\alpha_i\|_2^2}_{\text{reconstruction}} + \underbrace{\lambda\, v(\alpha_i)}_{\text{sparsity}}, \quad n > 100'000$$

Mahalanobis distance: $(x-\mu)^T \Sigma^{-1} (x-\mu) > t$

Covariance matrix: $\Sigma_{ij} = E[(X_i-\mu_i)(X_j-\mu_j)]$

Digital Image

Digital cameras: Transmission interference, compression artefacts, spilling, scratches, sensor noise, bad contrast, motion blur

CCD: Mature technology, specific technology, high production cost, high power consumption, higher fill rate, blooming, sequential readout

CMOS: Recent technology, standard IC technology, cheap, low power, less sensitive per pixel amplification, random pixel access, smart pixels, on chip integr. w. others

Quantization is lossy: original signal cannot be reconstructed

Lossless: Every bit remains in the compressed file (GIF)
Lossy: Some information is permanently eliminated (JPEG)
↳ advantage of redundancy

Additive Gaussian Noise:

$$I(x,y) = f(x,y) + c, \quad c \sim \mathcal{N}(0,\sigma^2)$$

$$p(c) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{(c-\mu)^2}{2\sigma^2}\right)$$

Poisson noise: $p(k) = \frac{\lambda^k e^{-\lambda}}{k!}$

SNR: $S = \frac{F}{\sigma}, \quad F = \frac{1}{xy}\sum_{x=1}^{x}\sum_{y=1}^{y} f(x,y)$

PSNR: $S_{peak} = \frac{F_{max}}{\sigma}$

Math:

$\sin(x) = \frac{1}{2i}(e^{ix} - e^{-ix})$ $\quad \cos(x) = \frac{1}{2}(e^{ix} + e^{-ix})$

$\cos(-x) = \cos(x)$ $\quad \sin(-x) = -\sin(x)$

$\sin(x \pm y) = \sin(x)\cos(y) \pm \cos(x)\sin(y)$

$\cos(x \pm y) = \cos(x)\cos(y) \mp \sin(x)\sin(y)$

$\sin(x)\cos(y) = \frac{1}{2}(\sin(x-y) + \sin(x+y))$

$\sin(x)\sin(y) = \frac{1}{2}(\cos(x-y) - \cos(x+y))$

$\cos(x)\cos(y) = \frac{1}{2}(\cos(x-y) + \cos(x+y))$

$$\delta(x-\alpha) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{ip(x-\alpha)}\,dp$$

$$\delta(u-u_0, v-v_0) = \iint_{\mathbb{R}^2} \exp(-j2\pi[(u-u_0)x + (v-v_0)y])\,dx\,dy$$

Phong shader: Small α → more reflection
Diffuse, ambient, specular

1. Interpolating the lighting parameters across the surface of a polygon
2. Computing the color per fragment (not vertex)

Interpolating the normal vectors results in smoother looking specular highlights.

Shader adaption: More code from the fragment shader to the vertex shader (calculate color using vertex shader)

Graphics pipeline: Vertex shader, rasterization, fragment shader
(OpenGL)

Fade out in vertex shader:
After projection in the vertex shader, the z-coord describes the depth in the space.
Either pass it to the fragment shader or pass the whole position, or compute the alpha there and pass it.

Convolution complexity: $\mathcal{O}(MNk^2)$
separable: $\mathcal{O}(MNk)$
Multiplications: $A \cdot N^2$, A - # distant values in kernel

Kernel separability: Rank = 1

Border behaviour:
- Zero-padding
- Border-replication

Filter dependency:
$W_{ap} = W_{hp} + W_{lp}, \quad W_{bp} = W_{lp} \otimes W_{hp}$

Spaces: Object, world & screen space

Filter size: If sum is higher than 1 then the intensity of the filtered image will be large → brighter than usual

Filter operations: $D_x \Rightarrow D_y = D_x^T$

$|\nabla I| = \sqrt{(D_x \otimes I)^2 + (D_y \otimes I)^2}$ $\quad \varphi(\nabla I) = \arctan\left(\frac{D_y \otimes I}{D_x \otimes I}\right)$

Magnitude $\quad$ Gradient orientation

PCA

1.1 Center & normalize data (subtract mean & divide by var)
1.2 Or compute covariance matrix of D
2. Perform EVD of cov. matrix (SVD: UIV^T)
3. EV give directions that describe variation in the data, independent of each other
4. The corresponding EW describes how large the variation along this direction is.

If orthogonal + real then EW & EV orthogonal

Covariance matrix: symmetric + real (for real data)

Compression: Image → row vector (center + norm matrix)
Subtract the mean image and project centered image onto the k EV with largest EW.

Decompression:
Multiplying k coefficients with corresponding EV
Then add mean image.

Eigenfaces: Components of the PCA on the face dataset (EV)

Pattern recognition: Compress an image and measure the error between it and the original.
If error below threshold, assume it is a part of the data set.

Lossless: All coefficients need to be kept
= more storage use (+ mean img & Eigen image)

Optical flow *

Aperture problem:
Refers to the fact, that when flow is computed for a point that lies along a linear feature, it is not possible to determine the exact location of the corresponding point in the second image. Thus, it is only possible to determine the flow that is normal to the linear feature.

Lucas-Kanade:
1. Brightness constancy: The intensity of the objects in the scene do not change in time
2. Small motion: Objects move very slowly from frame to frame, which means that corresponding points from 2 consecutive images are not far apart.
3. Spacial coherence: All points in a neighborhood have the same motion

Optical flow computation (iterative algorithm):
1. Estimate velocity at each pixel using one iteration of Lucas-Kanade estimation
2. Warp one image towards the other using the estimated flow field
3. Refine estimate by repeating the process.

Image stabilization:
Estimate the flow between the frames and wrap the image using the same flow over all the pixels so that the flow is close to 0.
(others: video compression, slow motion)

Image features *

Robust feature detector:
Invariant to brightness change, shift + rotation, scale

Edge detection:
Maximum of first derivative, zero crossing of second derivative. Only take edges with strong first derivative

Make Harris corners scale invariant:
Look for strong responses of DoG filter over scale space. Only consider local maxima in both position and scale space.

Rigid body collisions:
Adjust state of bodies after collisions by updating forces that act on the bodies or by updating momentum of the involved bodies.

Clipping: Save computation, memory consistency, stability (division by zero close to camera)

Normal map vs. bump map:
Normal maps have more information in them (because of rgb input)
Bump maps use only a black to white map

Mesh-rendering:
Lightning parameters are interpolated across the surface of a polygon, then the color is computed per fragment

Oriented pyramids:
Build a laplacian pyramid of the image of the texture and then apply some oriented filters to each level. This gives us image information at a particular scale and orientation.
+ Efficient (smaller filters on smaller images are more efficient than big filters on big images)

Aliasing: Sampling frequency is smaller than rotational speed mod features of wheel
→ low-pass filter or higher sampling rate

Histogram:
Encode frequency information (global brightness) without personal information (spatial relationships)
- Many textures can have same histogram (same brightness)
→ Covariance matrix (encodes position as well)
→ Co-occurence matrix (probability distribution for intensity pairs)

Integral image: The sum of all pixels in a rectangle, spanning from the origin to a point P. It allows to efficiently compute the convolution with a constant rectangle.

Homogeneous coordinates:
+ all transformations can be expressed as matrix operations
+ an arbitrary number of affine and projective mappings, applied one after the other, can be combined in one single matrix.

JPEG:
- General loss of sharpness and oscillations around high contrast edges.
- blocking structure
- loss of color detail

Z-Fighting: Two objects have the same value in the depth buffer (floatin point unprecision)

Mip mapping:
Stores many levels of the same texture. At each level, a different amount of low-pass filters are applied. low-pass filters reduce aliasing!

Nyquist proof:

Continuous signal : $f(t)$
Sampling function : $S_{\Delta t}(t)$
Sampling frequency : ω_s
Sampled function : $f(t)\, S_{\Delta t}(t)$
Fourier transformation: $F(\omega)$ of $f(t)$

$$\tilde{F}(\omega) = \mathcal{F}\left(f(t)\, S_{\Delta t}(t)\right)$$
$$= F(\omega) * S_{\Delta t}(\omega)$$
$$= \int_{-\infty}^{\infty} F(\tau)\, S_{\Delta t}(\omega - \tau)\, d\tau$$
$$= \int_{-\infty}^{\infty} F(\tau) \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} \delta\left(\omega - \tau - \frac{n}{\Delta T}\right) d\tau$$
$$= \frac{1}{\Delta T} \sum_{n=-\infty}^{\infty} F(\omega - n\omega_s)$$

If we want reconstruction of $f(t)$ to be possible, neighbors must not overlap:
$F(\omega - \omega_s)$ and $F(\omega + \omega_s)$
thus $\omega_s > 2\omega_h$, ω_h – highest frequency of $f(t)$

When nyquist fails → aliasing (sol: low-pass f.)