Computer Vision

Homogeneous Geometry

2D Transformations

- **Projective** (project)

$H = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}$ 8DOF 

invariants: concurrency, colinearity, tangency
cross ratio → $\frac{\overline{AC}\cdot\overline{BD}}{\overline{BC}\cdot\overline{AD}}$ = const. (if A,B,C,D points on line)

- **Affine** (move+rotate+squish+skew)

$H = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$ 6DOF 

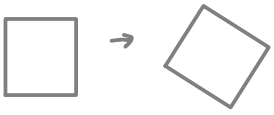
invariants: parallelism, ratio of areas $\frac{A_1}{A_2} = \frac{A_1'}{A_2'}$, ratios of lengths on parallel lines $\frac{\overline{AB}}{\overline{CD}}$ = const., linear combinations $\lambda_1 \underline{x}_1 + \lambda_2 \underline{x}_2 = \underline{x}_3 \Leftrightarrow \underline{x}_3' = \lambda_1 \underline{x}_1' + \lambda_2 \underline{x}_2'$, line at inf. $\underline{l}_\infty$

- **Similarity** (move+rotate+scale)

$H = \begin{pmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$ 4 DOF 

invariants: ratio of lengths, angles, circular points I; J, $\underline{\underline{C}}_\infty^*$

- **Euclidean** (move+rotate) (keeps lengths)

$H = \begin{pmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$ 3DOF 

invariants: lengths, areas

3D Transformations

- **Projective** (project) (keeps straight lines)

$H = \begin{pmatrix} \underline{\underline{A}} & \underline{t} \\ \underline{v}^T & v \end{pmatrix}$ 15 DOF 

invariants: intersections, tangency

- **Affine** (move+rotate+squish +sheas) (keeps parallel)

$H = \begin{pmatrix} \underline{\underline{A}} & \underline{t} \\ \underline{0} & 1 \end{pmatrix}$ 12 DOF 

invariants: parallelism of planes, volume ratios, centroids, plane at inf. $\underline{\pi}_\infty$

- **Similarity** (move+rotate+scale) (keeps angles)

$H = \begin{pmatrix} s\underline{\underline{R}} & \underline{t} \\ \underline{0} & 1 \end{pmatrix}$ 7 DOF 

invariants: angles, ratios of lengths
absolute conics $\underline{\underline{\Omega}}_\infty$; $\underline{\underline{\Omega}}_\infty^*$

- **Euclidean** (move+rotate) (keeps lengths)

$H = \begin{pmatrix} \underline{\underline{R}} & \underline{t} \\ \underline{0} & 1 \end{pmatrix}$ 6 DOF 

invariants: volume, ...

2D points ($\underline{x}$)

$\tilde{\underline{x}} = (\tilde{x}, \tilde{y}) \rightarrow \underline{x} = (x, y, w) \in \mathcal{P}^2 = \mathbb{R}^3 - (0,0,0)$

$\underline{x} = (x, y, w) = w(\tilde{x}, \tilde{y}, 1) = w\,\bar{\underline{x}} \quad (\tilde{x} = x/w, \tilde{y} = y/w)$

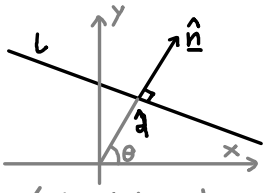
$\tilde{\underline{x}}$: inhomogenous vector $\underline{x}$: homogenous vector
$\bar{\underline{x}}$: augmented vector

2 coord. $\underline{x}_1, \underline{x}_2$ describe the same point if:
$x_1/w_1 = x_2/w_2$ and $y_1/w_1 = y_2/w_2$

point at inf.: $\underline{x} = (x, y, 0) \rightarrow$ cant be inhomog.

- transform a point: $\underline{x}' = \underline{\underline{H}}\,\underline{x}$

2D lines ($\underline{l}$)



$ax + by + cw = 0 \rightarrow \underline{l} = (a, b, c)$

normalized line: $\underline{l} = (\hat{n}_x, \hat{n}_y, \hat{d}) \quad (\|(\hat{n}_x, \hat{n}_y)\| = 1)$

↳ $(\hat{n}_x, \hat{n}_y)$: normal ved., $\hat{d}$: dist from orig.

line at inf.: $\underline{l}_\infty = (0, 0, 1)$

↳ includes all points at inf $\underline{x} = (x, y, 0)$

polar coord. line: $\underline{l} = (\cos\theta, \sin\theta, d)$

- point is on line: $\underline{x}^T \cdot \underline{l} = 0$
- intersection of 2 lines: $\underline{x} = \underline{l}_1 \times \underline{l}_2$
- line from 2 points: $\underline{l} = \underline{x}_1 \times \underline{x}_2$
- transform a line: $\underline{l}' = (\underline{\underline{H}}^{-1})^T \cdot \underline{l}$
- angle between lines: $\cos\theta = \frac{a_1 a_2 + b_1 b_2}{\sqrt{(a_1^2 + b_1^2)(a_2^2 + b_2^2)}}$

3D points ($\underline{X}$)

$\tilde{\underline{X}} = (\tilde{x}, \tilde{y}, \tilde{z}) \rightarrow \underline{X} = (x, y, z, w) \in \mathcal{P}^3 = \mathbb{R}^4 - (0,0,0,0)$

$\underline{X} = (x, y, z, w) = w(\tilde{x}, \tilde{y}, \tilde{z}, 1) = w\,\bar{\underline{X}} \quad (\tilde{x} = x/w, \tilde{y} = y/w, \tilde{z} = z/w)$

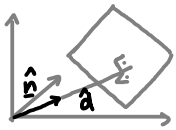
$\tilde{\underline{X}}$: inhomogenous vector $\underline{X}$: homogenous vector
$\bar{\underline{X}}$: augmented vector

2 coord. $\underline{X}_1, \underline{X}_2$ describe the same point if:
$\tilde{x}_1 = \tilde{x}_2, \tilde{y}_1 = \tilde{y}_2, \tilde{z}_1 = \tilde{z}_2$ (w_1, w_2: scale)

point at inf.: $\underline{X} = (x, y, z, 0) \rightarrow$ cant be inhomog.

- transform a point: $\underline{X}' = \underline{\underline{H}}\,\underline{X}$

3D planes (π)



$ax + by + cz + dw = 0 \rightarrow \underline{\pi} = (a, b, c, d)$

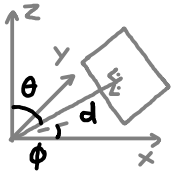
normalized plane: $\underline{\pi} = (\hat{n}_x, \hat{n}_y, \hat{n}_z, \hat{d}) \quad (\|\hat{\underline{n}}\| = 1)$

↳ $\hat{\underline{n}}$: normal ved., $\hat{d}$: dist from orig.

plane at inf.: $\underline{\pi}_\infty = (0, 0, 0, 1)$

↳ includes all points at inf $\underline{X} = (x, y, z, 0)$

spherical coord. plane:
$\underline{\pi} = (\cos\theta\cos\phi, \sin\theta\cos\phi, \sin\theta, \hat{d})$



- is point on plane: $\underline{X}^T \cdot \underline{\pi} = 0$
- plane from 3 points: solve $\begin{pmatrix} \underline{X}_1^T \\ \underline{X}_2^T \\ \underline{X}_3^T \end{pmatrix} \cdot \underline{\pi} = \underline{0}$ for $\underline{\pi}$
- point is on 3 planes: solve $\begin{pmatrix} \underline{\pi}_1^T \\ \underline{\pi}_2^T \\ \underline{\pi}_3^T \end{pmatrix} \cdot \underline{X} = \underline{0}$ for $\underline{X}$
- transform a plane: $\underline{\pi}' = (\underline{\underline{H}}^{-1})^T \cdot \underline{\pi}$

3D lines

↓ point $\underline{X}$ on line ↓

- line as 2 points $\underline{X}_1, \underline{X}_2$: $\underline{X} = \lambda_1 \cdot \underline{X}_1 + \lambda_2 \cdot \underline{X}_2$
- line as point + dir. $\underline{X}_1, \underline{d}$: $\underline{X} = \underline{X}_1 + \lambda(\underline{d}, 0)$
- line as 2 planes $\underline{\pi}_1, \underline{\pi}_2$: $\begin{pmatrix} \leftarrow \underline{\pi}_1^T \rightarrow \\ \leftarrow \underline{\pi}_2^T \rightarrow \end{pmatrix} \cdot \underline{X} = \underline{0}$
- Plücker line: $\underline{\underline{L}} = \underline{X}_1 \underline{X}_2^T - \underline{X}_2 \underline{X}_1^T \rightarrow$?

↓ plane $\underline{\pi}$ on line ↓

2D conics ($\underline{\underline{C}}$)

(5 DOF)

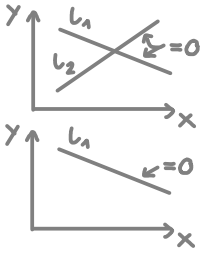
$$ax^2+bxy+cy^2+dxw+eyw+fw^2=0$$

$$\underline{\underline{C}}=\begin{pmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{pmatrix}$$

conic: $\underline{\underline{C}}$ / dual-conic: $\underline{\underline{C}}^*=\underline{\underline{C}}^{-1}$

degenerate-conic:

- rank($\underline{\underline{C}}$)=2 : $\underline{\underline{C}}=\underline{l}_1\underline{l}_2^T+\underline{l}_2\underline{l}_1^T$
- rank($\underline{\underline{C}}$)=1 : $\underline{\underline{C}}=\underline{l}_1\underline{l}_1^T$



- point is on conic : $\underline{x}^T\underline{\underline{C}}\,\underline{x}=0$
- line is tang. on conic : $\underline{l}^T\underline{\underline{C}}^*\underline{l}=0$
- tang @ point on conic : $\underline{l}=\underline{\underline{C}}\,\underline{x}$
- intersection conic-line : $(\underline{x}_1+\lambda\underline{x}_2)^T\underline{\underline{C}}(\underline{x}_1+\lambda\underline{x}_2)=0$

$$\underline{x}_1^T\underline{\underline{C}}\,\underline{x}_1+2\lambda\,\underline{x}_1^T\underline{\underline{C}}\,\underline{x}_2+\lambda^2\underline{x}_2\underline{\underline{C}}\,\underline{x}_2=0$$

- transform a conic : $\underline{\underline{C}}'=(\underline{\underline{H}}^{-1})^T\underline{\underline{C}}\,\underline{\underline{H}}^{-1}$

$$\underline{\underline{C}}^{*\prime}=\underline{\underline{H}}\,\underline{\underline{C}}^*\underline{\underline{H}}^T$$

circular points

intersection of : { all circular conics (b=0) ; all points at inf. ($\rightarrow \underline{l}_\infty$) (w=0) }

$\downarrow$

$x^2+y^2=0$ and $w=0$ $\rightarrow$ $\boxed{\underline{I}=(1,i,0)\ ,\ \underline{J}=(1,-i,0)}$

resulting conic : $\underline{\underline{C}}^*_\infty=\underline{I}\,\underline{J}^T+\underline{J}\,\underline{I}^T=\begin{pmatrix}1&0&0\\0&1&0\\0&0&0\end{pmatrix}$

- $\rightarrow$ contains all lines going through $\underline{I}$ or $\underline{J}$
- $\rightarrow$ $\underline{l}_\infty$ is its nullvector = line common to $\underline{I}$ and $\underline{J}$

- invariant for similarity transform and up:

$$\underline{I}=\underline{\underline{H}}_s\underline{I}\ /\ \underline{J}=\underline{\underline{H}}_s\underline{J}\ /\ \underline{\underline{C}}^*_\infty=\underline{\underline{H}}_s\underline{\underline{C}}^*_\infty\underline{\underline{H}}_s^T$$

- angle between 2 lines trough transform H :

$$\cos\theta=\frac{\underline{l}_1^T\underline{\underline{C}}^*_\infty\underline{l}_2}{\sqrt{(\underline{l}_1^T\underline{\underline{C}}^*_\infty\underline{l}_1)(\underline{l}_2^T\underline{\underline{C}}^*_\infty\underline{l}_2)}}$$

(even if $\underline{\underline{C}}^*_\infty,\underline{l}_1,\underline{l}_2$ transf. by some $\underline{\underline{H}}$!)

3D quadrics ($\underline{\underline{Q}}$)

$\underline{\underline{Q}}\in\mathbb{R}^{4\times4}$ with 9 DOF

quadric : $\underline{\underline{Q}}$ / dual-quadric : $\underline{\underline{Q}}^*=\underline{\underline{Q}}^{-1}$

degenerate-quadric : rank($\underline{\underline{Q}}$)$\neq4$

- point is on quadric : $\underline{X}^T\underline{\underline{Q}}\,\underline{X}=0$
- plane is tang. on quadric : $\underline{\pi}^T\underline{\underline{Q}}^*\underline{\pi}=0$
- tang @ point on quadric : $\underline{\pi}=\underline{\underline{Q}}\,\underline{X}$
- transform a quadric : $\underline{\underline{Q}}'=(\underline{\underline{H}}^{-1})^T\underline{\underline{Q}}\,\underline{\underline{H}}^{-1}$

$$\underline{\underline{Q}}^{*\prime}=\underline{\underline{H}}\,\underline{\underline{Q}}^*\underline{\underline{H}}^T$$

absolute conic

intersection of : { all spherical quadrics ; all points at inf. ($\rightarrow \underline{\pi}_\infty$) (w=0) }

$\downarrow$

conic on $\underline{\pi}_\infty$ with $x^2+y^2+z^2=0$

- $\rightarrow$ $\underline{\Omega}_\infty$: $(x,y,z)\,\mathbb{I}\,(x,y,z)^T=0$ and $w=0$
- $\rightarrow$ $\underline{\underline{\Omega}}^*_\infty=\begin{pmatrix}\mathbb{I}&0\\0^T&0\end{pmatrix}$

- invariant for similarity transform and up:

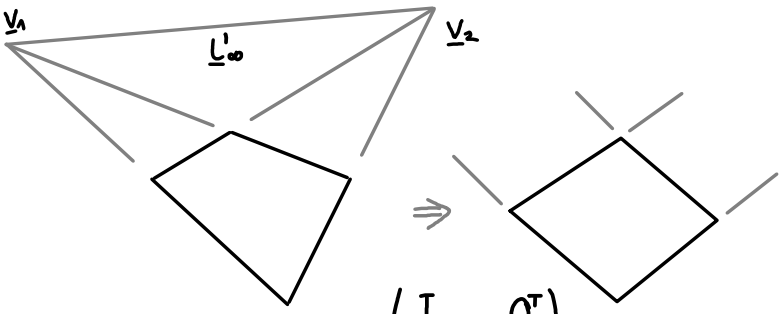
$$\underline{\underline{\Omega}}^*_\infty=\underline{\underline{H}}_s^T\underline{\underline{\Omega}}^*_\infty\underline{\underline{H}}_s^T$$

- angle between 2 lines trough transform H :

$$\cos\theta=\frac{\underline{\pi}_1^T\underline{\underline{\Omega}}^*_\infty\underline{\pi}_2}{\sqrt{(\underline{\pi}_1^T\underline{\underline{\Omega}}^*_\infty\underline{\pi}_1)(\underline{\pi}_2^T\underline{\underline{\Omega}}^*_\infty\underline{\pi}_2)}}$$

(even if $\underline{\underline{\Omega}}^*_\infty,\underline{\pi}_1,\underline{\pi}_2$ transf. by some $\underline{\underline{H}}$!)

Transformation: Projective → Affine



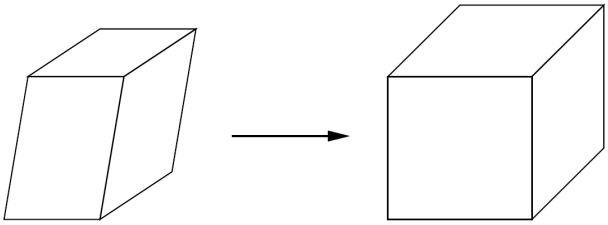
$$T_{PA} = \begin{pmatrix} I & 0^T \\ l_\infty, \pi_\infty & 1 \end{pmatrix}$$

1) find vanishing points v_1, v_2, v_3 by intersecting known parallel lines
2) find line/plane l'_∞ / π'_∞ going through v_1, v_2, v_3
3) find transform that maps l'_∞, π'_∞ to $l_\infty = (0,0,1)$, $\pi_\infty = (0,0,0,1)$
   ↳ $(T_{PA}^{-1})^T \cdot l_\infty, \pi_\infty = (0,0,0,1)$

Transformation: Affine → Similarity

how?

get $C^{*\prime}_\infty, \Omega^{*\prime}_\infty$ by finding known angles/length-ratios constraints (3 for 2D, 6 for 3D) and solving.



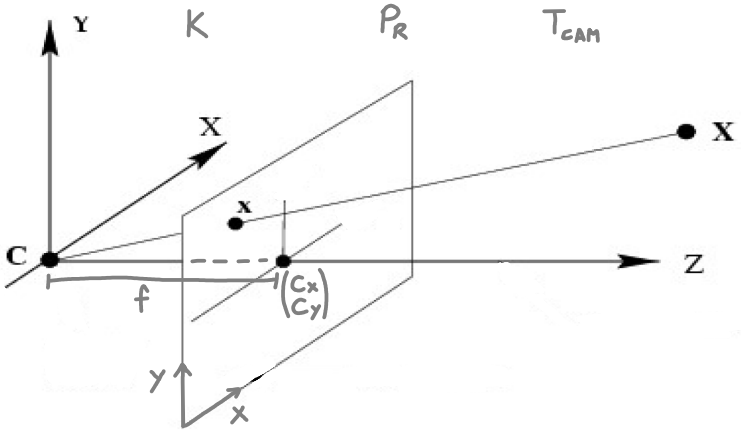
$$C^*_\infty, \Omega^*_\infty = \begin{pmatrix} A & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} I & 0 \\ 0^T & 0 \end{pmatrix} \begin{pmatrix} A & 0 \\ 0^T & 1 \end{pmatrix}^T = \begin{pmatrix} AA^T & 0 \\ 0^T & 0 \end{pmatrix} \longrightarrow T_{AM} = \begin{pmatrix} A^{-1} & 0 \\ 0^T & 0 \end{pmatrix}$$

2!2!

Pinhole-camera model (3D→2D)

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \underbrace{\begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}}_{K} \underbrace{\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}}_{P_R} \underbrace{\begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix}}_{T_{CAM}} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

P



f: image plane dist
p_x, p_y: width/height of pixel
c_x, c_y: coord (2D) of center pixel
α: skew angle of pixel
↳ $f_x = f/p_x$, $f_y = f/p_y$, $s = \tan\alpha \cdot f/p_y$
K: callibration matrix of the camera
P_e: projection to plane $z=1$
T_{CAM}: move world in camera frame
↳ if R, t move camera instead → $T_{CAM} = \begin{pmatrix} R^T & -R^T t \\ 0^T & 1 \end{pmatrix}$
P: projection mat. $P = K \cdot P_e \cdot T_{CAM}$

- project point: $x = PX$
- project line: $L = PX_1 \times PX_2$ (X_1, X_2: line)
- project plane: ?
- project quadric: $C^* = PQ^*P^T$

pseudo inv.

- back-project point: $X(\lambda) = P^+ x + \lambda C$
- back-project line: $\pi = P^T L$
- back-project conic: $Q_\infty = P^T C P$ ↳ degenerate!

Direct linear transform (DLT)

(get camera proj. $\underline{\underline{P}}$ from pairs $\underline{x}_i, \underline{X}_i$ (inhomog.!))

($\underline{\underline{P}} \in \mathbb{R}^{3\times 4}$ → 11 DOF → 6+ points needed)

- **algebraic error:**

$$x_i = \frac{p_{00}X_i + p_{01}Y_i + p_{02}Z_i + p_{03}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}} \Rightarrow p_{00}X_i + p_{01}Y_i + p_{02}Z_i + p_{03} - x_i(p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}) = 0$$

$$y_i = \frac{p_{10}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}}{p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}} \Rightarrow p_{10}X_i + p_{11}Y_i + p_{12}Z_i + p_{13} - y_i(p_{20}X_i + p_{21}Y_i + p_{22}Z_i + p_{23}) = 0$$

↓ 6+ of those pairs of equations

minimize $\|\underline{\underline{A}}\underline{p}\|$: ⇐ $\underline{\underline{A}} \cdot (\underbrace{p_{00},p_{01},p_{02},p_{03}}_{P_0^T}, \underbrace{p_{10},p_{11},p_{12},p_{13}}_{P_1^T}, \underbrace{p_{20},p_{21},p_{22},p_{23}}_{P_2^T})^T = \underline{0}$

singular-value decomp. of A

$A = U \cdot S \cdot V^T$ → p = rightmost col of V (with smallest S entry)

⇓

$\underline{\underline{P}}$ found. now find $\underline{\underline{K}}, \underline{\underline{R}}, \underline{t}$ with $P = \underline{\underline{K}}(\underline{\underline{R}}|\underline{t}) = (\underline{\underline{KR}}|\underline{Kt})$

- RQ decomposition of $M = (KR)^{-1}$ → $M = R^* \cdot Q^*$ → $K = R^{*-1}$, $R = Q^{*-1}$
- get camera position $\underline{C}$: solve $\underline{\underline{P}}\underline{C} = 0$ (same way as solving $\underline{\underline{A}}\underline{p} = \underline{0}$) → $\underline{t} = -\underline{\underline{R}} \cdot \underline{C}$

- **algebraic error + normalization:**

normalize every coord $(x_i, y_i, X_i, Y_i, Z_i)$ so that $\frac{1}{N}\sum^n [\text{coord}] = 0$, $\frac{1}{N}\sum^n [\text{coord}]^2 = 1$

↳ $\underline{x}_n = \underline{\underline{T}} \cdot \underline{x}$, $\underline{X}_n = \underline{\underline{U}} \cdot \underline{X}$ → get P_n with above steps → $P = T^{-1} P_n U$ → continue above steps.

- **geometric error (gold standard):**

same as algebraic, but minimize $\sum_i \text{dist}(\underline{x}_i, \underline{\underline{P}}\underline{X}_i)^2$

$\underline{X}(\lambda) = \underline{\underline{P}}^+ \underline{x} + \lambda \underline{C}$

Two-view geometry

(find point ↔ line mapping of 2 cameras given matching pairs $\underline{x}_i, \underline{x}_i$)

$\underline{l}' = \underline{\underline{F}}\underline{x}$, $\underline{l} = \underline{\underline{F}}'\underline{x}'$

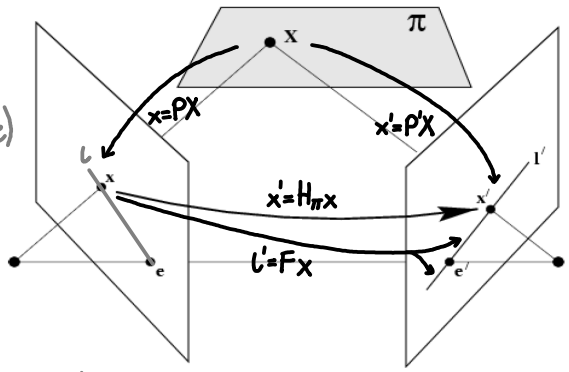
$\underline{x}'^T \underline{\underline{F}} \underline{x} = \underline{0}$, $\underline{x}^T \underline{\underline{F}}' \underline{x}' = 0$

$\underline{\underline{F}} = [e']_\times H_\pi = [e']_\times P' P^+$ (fundamental matrix) ← plücker matrice

7 DoF

$\underline{\underline{F}}' = \underline{\underline{F}}^T$, $e'^T \underline{\underline{F}} = \underline{0}$, $\underline{\underline{F}}\underline{e} = \underline{0}$

e, e': epipoles, l, l': epilines



- **8 point alg:** (find $\underline{\underline{F}}, \underline{\underline{P}}'$ from 8 matches $\underline{x} \leftrightarrow \underline{x}'$, $\underline{\underline{P}} = [\underline{\underline{I}}|\underline{0}]$)

$\underline{x}'^T \underline{\underline{F}} \underline{x} = \underline{0}$ → $x'x f_{11} + x'y f_{12} + x' f_{13} + y'x f_{21} + y'y f_{22} + y' f_{23} + x f_{31} + y f_{32} + f_{33} = 0$

↳ $\begin{bmatrix} x'x, x'y, x', y'x, y'y, y', x, y, 1 \\ \vdots \end{bmatrix} \cdot [f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{13}, f_{23}, f_{33}]^T = 0$

$\underline{\underline{A}} \cdot \underline{f} = 0$

↳ minimize $\|\underline{\underline{A}}\underline{f}\|$: SVD of $A \Rightarrow U \cdot S \cdot V^T$ → f rightmost col of V

↳ $P = [\underline{\underline{I}}|0]$, $P' = [[e']_\times F + e'v^T | \lambda e']$ (v, λ: any value!)

↳ pick with e.g. known angles between walls

normalization:

$\underline{x}_n = T \cdot \underline{x}$, $\underline{x}'_n = T' \underline{x}'$ so that img bottom-left = (-1,-1), top-right = (1,1)

↳ get F_n like left.

→ $F = T'^T F_n T$

- **7 point alg:** (1 point less + use $\det(F)=0$)
  same as 8 point, but take 2 rightmost rows of $V \to F_1, F_2 \to$ find λ with $\det(F_1+\lambda F_2)$ by: $\lambda = EW(F_2^{-1}F_1)$ (only real!) $\to F = F_1+\lambda F_2$
  (alternative F from 8-point $\to USV^T = \text{svd}(F) \to$ set $S_{33}=0 \to$ recreate $F' = US'V^T$)
- **5 point alg:** (3 point less + use $\det(F)=0$, $2FF^TF - \text{trace}(FF^T)F = 0 \Leftrightarrow \lambda_1=\lambda_2$ and $\lambda_3=0$) ??
- **known up:** (constraint from knowing how camera up changed from P to P') ???

- **3D point $\underline{X}$ constraint:** (↱ stack any 4 rows to ↴)
  matching 2D point: $\begin{pmatrix} w\cdot x \\ w\cdot y \\ w \end{pmatrix} = \begin{pmatrix} \vdash \underline{P}_1 \dashv \\ \vdash \underline{P}_2 \dashv \\ \vdash \underline{P}_3 \dashv \end{pmatrix} \cdot \underline{X} \to \begin{pmatrix} \underline{P}_3 \cdot x - \underline{P}_1 \\ \underline{P}_3 \cdot y - \underline{P}_2 \end{pmatrix} \cdot \underline{X} = 0 \to \begin{pmatrix} \vdash \underline{A}_1 \dashv \\ \vdash \underline{A}_2 \dashv \\ \vdash \underline{A}_3 \dashv \\ \vdash \underline{A}_4 \dashv \end{pmatrix} \cdot \underline{X} = \underline{0} \to$ **if constraint is met: $\det(\underline{A}) = 0$**
  matching 2D line: $\underline{\pi} = \underline{P}^T \cdot \underline{l}$, $\underline{\pi}^T \underline{X} = 0 \to (\underline{P}_1 l_1 + \underline{P}_2 l_2 + \underline{P}_3 l_3) \cdot \underline{X} = 0$

Factorization (reconstruct camera projections $\underline{P}_i$ $(i=1,..,m)$ and 3D points $\underline{X}_j$ $(j=1,..,n)$, given all 2D points $\underline{x}_{ij}$)

- **orthographic factorization:** (assume $\underline{P}_i$ is an orthographic projection)
  $\underline{x}_{ij} = \begin{pmatrix} \tilde{x}_{ij} \\ \tilde{y}_{ij} \end{pmatrix}$, $\underline{P}_i = \begin{pmatrix} \underline{P}_i^x \\ \underline{P}_i^y \end{pmatrix}$, $\underline{X}_j = \begin{pmatrix} X_j \\ Y_j \\ Z_j \end{pmatrix} \to \underline{x}_{ij} = \underline{P}_i \underline{X}_j$
  $\underline{x} = \begin{pmatrix} \underline{x}_{11} \cdots \underline{x}_{1n} \\ \vdots \\ \underline{x}_{m1} \cdots \underline{x}_{mn} \end{pmatrix}$, $\underline{P} = \begin{pmatrix} \underline{P}_1 \\ \vdots \\ \underline{P}_m \end{pmatrix}$, $\underline{X} = (\underline{X}_1, \ldots, \underline{X}_n) \to \underline{x} = \underline{P}\underline{X}$
  $\in \mathbb{R}^{2m \times n}$, $\in \mathbb{R}^{2m \times 3}$, $\in \mathbb{R}^{3 \times n}$, $\text{rank}(\underline{x}) = 3$
  svd: $\underline{x} = U\Sigma V^T \to$ set all EW in Σ to 0 except largest 3 (force rank 3 constraint.)
  reconstruct affine $\tilde{\underline{P}}, \tilde{\underline{X}}$: $\tilde{\underline{P}} = U$, $\tilde{\underline{X}} = \Sigma V'$
  solve for $\underline{C}$: $\tilde{\underline{P}}_i^x \underline{C} \tilde{\underline{P}}_i^{xT} = 1$; $\tilde{\underline{P}}_i^y \underline{C} \tilde{\underline{P}}_i^{yT} = 1$; $\tilde{\underline{P}}_i^x \underline{C} \tilde{\underline{P}}_i^{yT} = 0$
  solve for $\underline{A}$: $\underline{C} = \underline{A}^{-1} \cdot \underline{A}^{-T}$ (Cholensky factorization)
  reconstruct metric $\underline{P}, \underline{X}$: $\underline{P} = \tilde{P}A^{-1}$, $\underline{X} = A\tilde{\underline{X}}$
- **perspective factorization:**
  $\underline{x}_{ij} = \begin{pmatrix} \tilde{x}_{ij} \\ \tilde{y}_{ij} \end{pmatrix}$, $\underline{P}_i = \begin{pmatrix} \underline{P}_i^x \\ \underline{P}_i^y \end{pmatrix}$, $\underline{X}_j = \begin{pmatrix} X_j \\ Y_j \\ Z_j \\ 1 \end{pmatrix} \to \lambda_{ij} \underline{x}_{ij} = \underline{P}_i \underline{X}_j$ (λ_{ij}: factor from w-coord $w_{ij} = \underline{P}_i^w \underline{X}_j$, UNKNOWN)
  $\underline{x} = \begin{pmatrix} \lambda_{11}\underline{x}_{11} \cdots \\ \vdots \\ \cdots \lambda_{ij}\underline{x}_{ij} \end{pmatrix}$, $\underline{P} = \begin{pmatrix} \underline{P}_1 \\ \vdots \\ \underline{P}_m \end{pmatrix}$, $\underline{X} = (\underline{X}_1, \ldots, \underline{X}_n) \to \underline{x} = \underline{P}\underline{X}$
  $\in \mathbb{R}^{2m \times n}$, $\in \mathbb{R}^{2m \times 4}$, $\in \mathbb{R}^{4 \times n}$, $\text{rank}(\underline{x}) = 4$
  1) assume $\lambda_{ij} = 1 \ \forall (i,j)$
  2) calculate $\underline{P}_i, \underline{X}_j$ like orthographic (force rank 4 constraint.)
  3) if 5th EW of Σ was small → stop.
  4) estimate new λ_{ij} from $\lambda_{ij}\underline{x}_{ij} = \underline{P}_i \underline{X}_j$
  5) go to 2.

- **refine results of above methods:** minimize reprojection error $\min_{P,X} \sum_{ij} \| \underline{x}_{ij} - \underline{P}_i \underline{X}_j \|_2$ (bundle adjustment)

feature definition

local feature = properties of a smal patch of an image used to describe an object

- has to be: invariant to changes in viewpoint, illumination, background, occlusion, ...
- has to be: unique in comparison to other features.

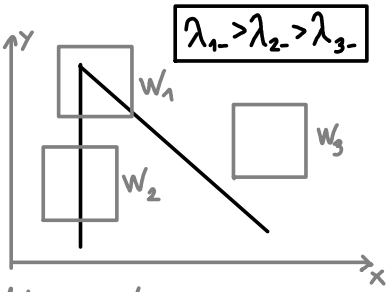
detection (finds feature positions in an image)

- standard corner detector:

$$E(u,v) = \sum^{W} (I(x+u,y+v) - I(x,y))^2$$

$$\approx \sum^{W} [u\ v] \underbrace{\begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}}_{\underline{\underline{H}}} \begin{bmatrix} u \\ v \end{bmatrix}$$

$I(x,y)$: intensity of image
$(x,y) \in W$: all pixels in a small patch W (7x7 px)
(u,v) : direction that patch W is shifted
$(I_x(x,y) = \frac{\partial I(x,y)}{\partial x})$



- $EV(\underline{\underline{H}})_+ = \underline{x}_+$: (u,v) with largest increase in E
- $EV(\underline{\underline{H}})_- = \underline{x}_-$: (u,v) with smallest increase in E
- $EW(\underline{\underline{H}})_+ = \lambda_+$: ammount of increase in $\underline{x}_+$
- $\boxed{EW(\underline{\underline{H}})_- = \lambda_-}$ ammount of increase in $\underline{x}_-$

($I_x(x,y) = I(x+1,y) - I(x-1,y)/2$; $I_y(x,y) = I(x,y+1) - I(x,y-1)/2$; points from $\lambda_-(x,y)$: local maxima → treshold)

- Harris corner detector:

instead of $\lambda_- = EW(\underline{\underline{H}})_-$ do: $\lambda_- \approx \det(\underline{\underline{H}}) - k \cdot \overbrace{(h_{11}+h_{22})}^{\text{trace}(H)}{}^2 = \lambda_+\lambda_- - k(\lambda_+ + \lambda_-)$ ($k = 0{,}04 \dots 0{,}06$)

- difference of gaussian (DoG) corner detector: ???
- MSER blob detector: (region : connected group of pixels all above or below treshold)
  slowly raise treshold intensity → when a region grows the slowest its a blob (bounded by strong edges)
- LoG blob detector: smooth image with gaussian filter → do laplacian → max on center of blobs
- ANMS adaptive non-maxima suppression: ???

description (gets values describing the patch arround a detection)

- transformation invariants:
  - similarity (3 Points): $\frac{\|A-B\|}{\|A-C\|} = \text{const.}$
  - affinity (4 Point): $\frac{|A-B \times A-C|}{|A-B \times A-D|} = \text{const.}$
  - projectivity (5 Point): $\frac{|A-B \times A-E|}{|A-C \times A-E|} \Big/ \frac{|B-D \times B-E|}{|C-D \times C-E|} = \text{const.}$

(if patch is assumed small, similarity holds for affine and even projective transforms!)

- photometric invariants:

approx of color transf.: $\begin{pmatrix} r' \\ g' \\ b' \end{pmatrix} \approx \begin{pmatrix} s_R & 0 & 0 \\ 0 & s_g & 0 \\ 0 & 0 & s_b \end{pmatrix} \cdot \begin{pmatrix} r \\ g \\ b \end{pmatrix} + \begin{pmatrix} o_R \\ o_g \\ o_b \end{pmatrix}$

if patch transformation matches, then color normalized patches (mean=0, var=1) are invariant.

- **normalized cross correlation (NCC)** (directly compare pixels of 2 patches!) (invariant to: • translation • illumination)

$$s(u,v) = \frac{\sum_i (u_i-\bar{u})(v_i-\bar{v})}{\sqrt{\sum_i (u_i-\bar{u}) \sum_i (v_i-\bar{v})}}$$

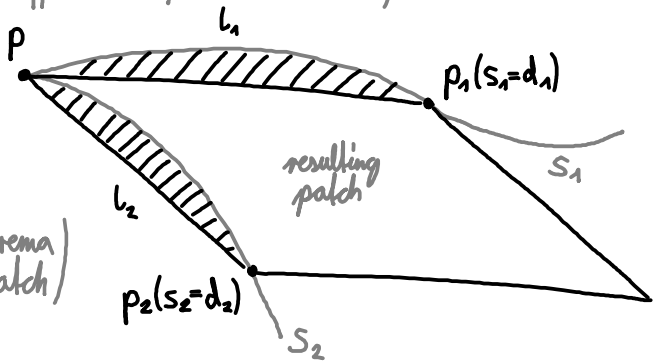
u_i, v_i: intensities of 2 patches being compared.
i: usually 1...49 = 7x7px around detected edge.
$\bar{u}, \bar{v}$: avg. intensities of 2 patches
(s high → good match)

- **parallelogram next to edge corner** (invariant to: affine transf. + illumination)

1) Harris corner detection: for feature position p
2) Canny edge detection: find 2 edges of corner s_1, s_2
3) find 2 points on 2 edges with same area $L_1 = L_2$

$$L = \int_0^d \left| \frac{dp(s)}{ds} \times (p(0)-p(s)) \right| ds$$

(pick d_1, d_2 to get local extrema of avg color intensity in patch)



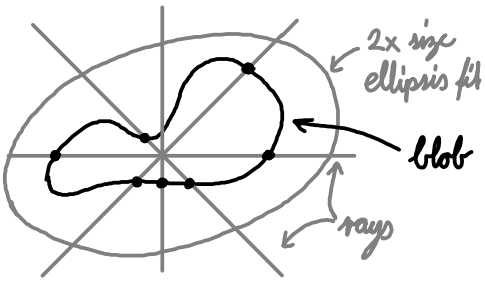
4) Describe patch with e.g. invariant moment D_{02}

$$D_{02} = \frac{(M_{00}^{11} M_{00}^{00} - M_{00}^{10} M_{00}^{01})}{(M_{00}^{20} M_{00}^{00} - (M_{00}^{10})^2)(M_{00}^{02} M_{00}^{00} - (M_{00}^{01})^2)} \quad \text{if } c=0$$

$$M_{pq}^{abc} = \int x^p y^q \cdot r(x,y)^a \cdot g(x,y)^b \cdot b(x,y)^c \, dx dy \qquad \begin{cases} x,y \in \text{patch} \\ r,g,b: \text{pixel color} \end{cases}$$

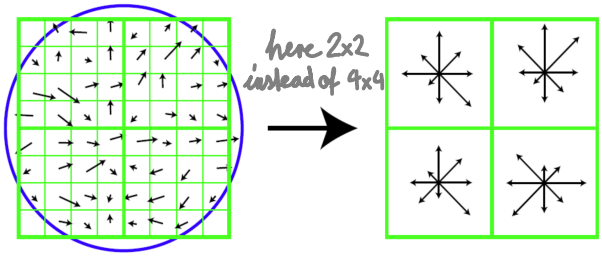
- **intensity extremas (blobs) + ellipse patch** (invariant to: affine transf. + illumination)

1) find blobs / intensity extrema
2) find points of max intensity changes along rays
3) fit ellipse to points and double its size
4) compute invariant of elliptic patch (e.g. D_{02} from above)



- **SIFT** (scale-invariant feature transform) (invariant to similarity + illum.)

1) use LoG method to find blobs
2) get square patch around blob and compute gradients
3) subdivide (square) patch of blob in 4x4 zones
4) bin gradient in each zone into 8 principal directions (4x4x8 bins)
5) normalize + rotate, so strongest dir is up. ↑ (weight with dist. from center)



- **SURF**: faster version of SIFT
- **HoG** (historigram of oriented gradients): like step 4 of SIFT

matching

smallest sum-of-squared-differences (SSD)

- **one way nearest neighbor**: for every descriptor of image 1, find best match in image 2
- **mutual nearest neighbor**: do one-way and one-way with swapped img 1↔2. intersect matches.
- **one-way + ratio test**: do one-way and find also 2nd best. if $\frac{\text{1st best}}{\text{2nd best}} < \sim 0.5$ → discard.

optical flow (find speed of every pixel in a video)

$I(x(t), y(t), t)$: intensity of a pixel that moves ("fixed" to a object) in a video

$u = \frac{dx}{dt}$, $v = \frac{dy}{dt}$: momentan speed of a pixel

- $\frac{dI}{dt} = \frac{\partial I}{\partial x} \cdot u + \frac{\partial I}{\partial y} \cdot v + \frac{\partial I}{\partial t} \approx 0$ (assume that moving pixel does not change intensity)
- $\left(\frac{du}{dx}\right)^2 + \left(\frac{dv}{dx}\right)^2 + \left(\frac{du}{dy}\right)^2 + \left(\frac{dv}{dy}\right)^2 \approx 0$ (assume u,v map is "smooth")

$\frac{\partial I}{\partial x} = \frac{I(x+1,y,t) - I(x-1,y,t)}{2}$

$\frac{\partial I}{\partial y} = \frac{I(x,y+1,t) - I(x,y-1,t)}{2}$

$\frac{\partial I}{\partial t} = \frac{I(x,y,t+1) - I(x,y,t-1)}{2}$

$\frac{du,v}{dx,y}$: same as above

find $u(x,y), v(x,y)$ that minimizes : $\iint (u_x^2 + u_y^2 + v_x^2 + v_y^2) + \lambda (I_x u + I_y v + I_t)^2$

λ : some const.

↓ (calculation of variations → Horn+Schunck algorithm)

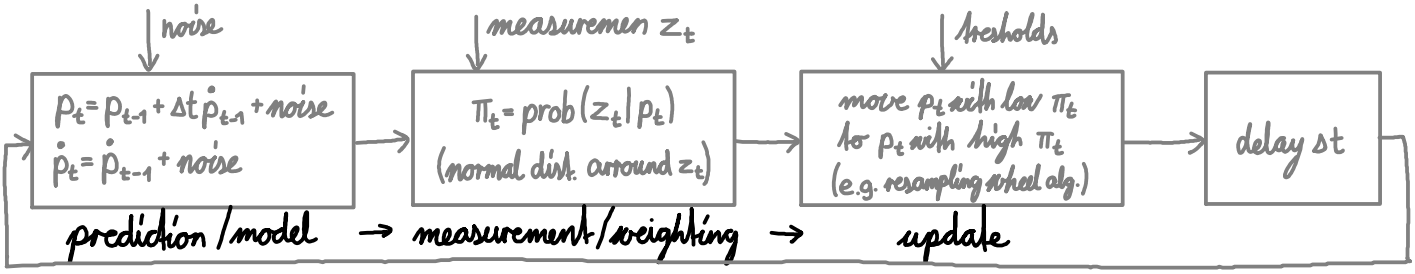
$\partial u / \partial i = \Delta u - \lambda (I_x u + I_y v + I_t) \cdot I_x$; $\partial v / \partial i = \Delta v - \lambda (I_x u + I_y v + I_t) \cdot I_y$

$\Delta[\cdot] = \partial[\cdot]^2/\partial x^2 + \partial[\cdot]^2/\partial y^2$

i : iteration

condensation filter (object tracker for video (similar to Kalman / Bayesian filter))

p_t : many position estimates at time t (mimicking probability distribution) ; π_t : prob. that p_t is true



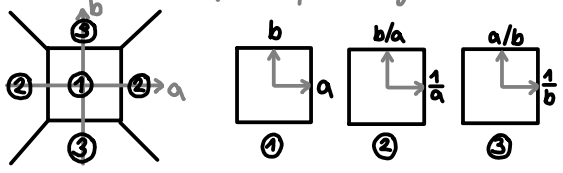
model fitting

- **hough transform :**
  1) discretize model parameter space
  2) count up cells that explain a datapoint
  3) find peaks in cell value

  examples with line fitting :
  - model is $y = ax + b$. data space (x,y), parameter space (a,b)
  - all cells (a,b) that explain a point get counted up +1
  - the cell (a,b) with the highest count is a good model

  ! problem if param. space unbounded !
  ! param. space cell size has to be "right" !
  (guess...)

  - use $x\cos\theta + y\sin\theta = s$ instead (still: bounding in s?)
  - use distorted subspaces for large (a,b)

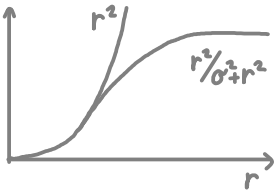


- **K-means line fitting** (for when there are multiple lines in the data)
  1) generate randomly placed lines (guess line count)
  2) assign each point to closest line
  3) refit lines to assigned points
     (least-square?)
  4) go to step 2 many times
  5) go to step 1 many times
  6) pick the iteration with lowest error.

robust estimation (alternative to least-squares, that's better against outliers)

instead of minimizing Σr^2 (r: dist. datapoint to model), minimize $\Sigma \frac{r^2}{\sigma^2+r^2}$

$\sigma^2 = 1.4826 \cdot \text{median}(r^2)$



RANSAC:

n: min. # of points needed to determine model
k: # of algorithm iterations
t: treshold dist. to determine inliers of model

while less than k iterations
  take n randomly sampled points from data
  fit model to n points
  find all other datapoints within t of model
  refit model to all these points
pick model with smallest fitting error

k: $k = \log(1-p)/\log(1-(1-e)^n)$ ($p \approx 0.99$)
p: probability 1+ sample has all inliers
e: approximate proportion outlier/total
(recalc. k each iteration if e unknown. e = lowest outlier/total up to that moment)

t: empirical / so that X% data included

improvements:
- minimize median residual instead of maximizing inliner count (LMedS)
- sample 'good' points more frequently
- if quasi-degenerate data: do RANSAC reducing n until model starts getting bad → sort out explainable data → use remaining DoF to fit remaining data

Segmentation (find regions of an image that belong together)

k-means:

1) give features to each pixel (1D: intensity, 3D: color, 5D: r,g,b,x,y,...)
2) randomly initialize k cluster centers $c_1,..,c_k$ in the feature space
3) for each pixel p, put it in the closest cluster c_i with $\min \|p-c_i\|^2$
4) move each cluster to be at the mean of all contained pixels.
5) repeat steps 3,4 until clusters stop moving.

cons:
- how to choose k?
- can be "local minimum"
- detects spherical clusters only
- sensitive to outliers
- sensitive to initial cluster locations

mixture of gaussians, EM:

similar to k-means, but clusters are defined by $\theta_b = (\alpha_b, \mu_b, V_b)$

$$R(x,\theta_b) = \alpha_b / \sqrt{(2\pi)^d |V_b|} \cdot \exp\left(-\tfrac{1}{2}(x-\mu_b)^T \cdot V_b^{-1}(x-\mu_b)\right)$$

$$P(x|\theta_b) = R(x,\theta_b) / \textstyle\sum_{i=1}^{k} R(x,\theta_i)$$

(μ_b: mean vector, V_b: covariance matrix, $P(x|\theta_b)$: prob. that x is in θ_b)

cons:
same as k-mean, but better with outliers.

assign every pixel to the θ_b with highest $P(x|\theta_b)$
given the assigned pixels of θ_b, change μ_b, V_b to maximize $P(x|\theta_b)$

$$\alpha_b^{new} = \frac{1}{N} \sum_{i=1}^{N} P(x_i|\theta_b)$$

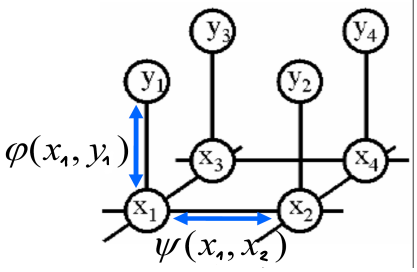
$$\mu_b^{new} = \sum_{i=1}^{N} x_i \cdot P(x_i|\theta_b) \Big/ \sum_{i=1}^{N} P(x_i|\theta_b)$$

$$V_b^{new} = \sum_{i=1}^{N} (x_i-\mu_b^{new})(x_i-\mu_b^{new})^T P(x_i|\theta_b) \Big/ \sum_{i=1}^{N} P(x_i|\theta_b)$$

- **mean-shift**
  1) define feature space like in k-mean
  2) place windows (e.g.: circle, sphere) on every pixel in the feature space
  3) for every window: move it to the mean of all contained pixels
  4) do step 3 until all windows stop moving
  5) pixels with windows that end up in the same place are in one segment.

  cons:
  - window size selection difficult
  - computationally expensive

- **Graph-cuts:**
  1) pick k key-features x
  2) for every pixel/feature y calculate the "dist." to every key-feature x → $\varphi(x,y)$
  3) for every combination of 2 key-features define a "distance" → $\psi(x_i,x_j)$
  4) assign every pixel to a key-feature while minimizing $E = \sum\varphi + \sum\psi$
     ↳ use graph-cuts to find division (cuts) of regions with same key-feature

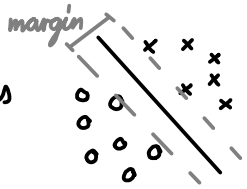


  cons:
  - how to pick key-features x

- **k-nearest neighbor (KNN)** (!needs training dataset!)
  for every pixel find the nearest k points of the training dataset in featurespace, and assign it to a segment based on the k points.

  cons:
  - hard to choose k
  - needs large training dataset

- **support vector machine (SVM):**
  find hyperplane (line in 2D case) that divides the featurespace and has maximal margin



  cons:
  - only 2 clusters
  - not always solvable

- **random forest**
  1) build many random binary decision trees (= forest)

  (feature-space description of pixel is input to the root. each node has a feature dimension and a treshold to decide if to go right or left. the leafs corrispond to segments that the tree chooses.)

  2) train the trees by changing node parameters / killing, growing branches to make them predict a testing dataset. how?
  3) use the average output of the trees to predict the segment of new pixels.

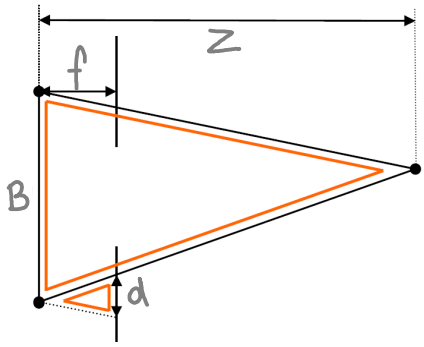
  cons:
  - lots of parameters to choose
  - needs large training dataset
  - training the forest is slow

non-linear least-squares (find solution to eq. that minimizes error)

$X = f(P)$ (X: solution, f: some non-linear function, P: variable for solving the equation)

- Newton method: P_0 = guess → $P_{i+1} = P_i + \Delta$, $\Delta = (J^TJ)^{-1}J^T(X - f(P_i))$, $J = \frac{\partial f}{\partial P}$ ← $(f(P_0+\Delta) \approx f(P_0) + J\cdot\Delta)$
- Levenberg-Marquardt method: same except $\Delta = (J^TJ + \lambda\,\mathrm{diag}(J^TJ))^{-1}J^T(X - f(P_i))$ $\lambda_0 = 10^{-3}$ → how to change λ ?!?

Stereo matching (find point correspondences in 2 images to compute depth + 3D model)



(cameras are identical and point in same direction)

$F = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{pmatrix}$ (fundamental matrix) (defining B line as x-axis) → epipolar lines = horizontal

for every feature in camera 1 search for a match on the corrisponding epipolar line and get the offset d. (see below for methods)

→ depth $Z = B \cdot f / d$

- **feature match constraint**: 1 feature should have 1 or less matches. see above chapter for details.
- **ordering constraint**: order of matched features along epipolar lines should not be allowed to change. (only allow to skip features, not reorder!)
- **disparity constraint**: only realistic depths + depth gradients need to be considered.

- foreshortening distortion of feature windows can be compensated after a rough depth map estimate
- feature matches along epipolar lines can be skipped due to occlusion.
- feature window matching improves by considering interpolated windows (between pixel)
- window on background pixel containing foreground ends up in foreground! try shifting window off-center, then taking best offset (not always, since depth accuracy loss)
- compact windows, rod-shaped windows, weight window based on similarity + proximity
- match each pixel by time varying illumination of the 2 cameras (if available)
- depth map with semi-global optimization: graph cuts, belief propagation, dynamic programing
- image pair rectification can make epipolar lines of any 2 img become horizontal = scan lines = fast.
- a third camera is usefull for verification...
- if baseline dist. B can be choosen: too small → dept error, too large → much occlusion (use multiple helps)
- plane-sweep stereo matching: assume depth → render images on top each other → find where result is sharp → assign the assumed depth to that region.

Structure from motion (pipeline to get dense 3D point cloud from images)

- do feature detection + matching (with segmentation) to get sparse 2D point matches
- do factorization to get camera projections P + sparse 3D points between all image pairs
- discard outlier cameras + improve P matrices by enforcing: 3D points of same feature in different views match + loops of relative projections have to return to initial camera pos.
- do stereo matching with each image pair along its epipolar lines to get dense 3D points
- somehow combine all stereo match results into one consistent dense 3D point cloud

Specific object recognition (find pose+position of an object in an image if it exists)

(need decent slides to be uploaded!)

- **model based** (use a 3D mesh or feat. desc. model)
  1) get features (best if invariant) from image
  2A) try to position the model such that similar features appear
  2B) or try to describe object with one feature descriptor and use that as model/comparison
  ↳ compact model + good with clutter

- **image based** (use images of object in different poses)
  directly compare image pixels to all possible pose images to determine pose (position determined a priori with clear background + bounding box)
  ↳ fast evaluation, easy to produce model

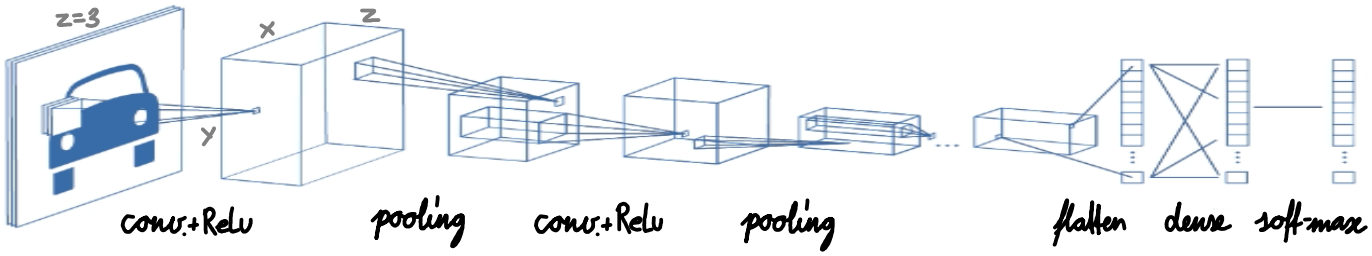
- **hybrid techniques** (combination of model based + image based)
  - use features (best if invariant) of a few training images in different poses as model.
  - detect features in image and determine match to model by counting feature matches

Object class recognition (recognize class of object in image, e.g. cow, car, ..)

- **bag of words:**
  training: 1) extract features (e.g. SIFT) from training images and assign each to its object class
  2) segment feature space (e.g. mean shift) and save cluster centers = words
  usage: 1) get features from target image and assign them the class of a near word (e.g. k-nearest-neighbor)
  2) let all feature classes vote on what is the most probable object class

! spatial information is not regarded ! :  all match. "solutions":
- include pos. in feat. desc.
- phrases (co-occuring words)
- evaluate config. afterwards
- vector from word to obj center

- **convolutional neural network (CNN):** does feature extraction + image segmentation in one.



- (conv.) convolution layer: apply many conv. kernels to each previous channel pixel (grow in z)
  kernel parameters are trained by backpropagation
- (ReLu) rectified layer: apply non-linear func to everything (so that CNN can capture non-linearities)
  e.g. sigmoid, $x = x$ if $x > 0$; 0 else, ...
- pooling: reduce x,y size by some factor by taking e.g. only the max value. (↑ compensate for z growth)
- flatten: flatten x·y·z cube to 1×(x·y·z) (necessary for dense layers)
- dense layer: multiply input with a dense matrix (linear combination of all inputs to output)
  the dense matrix is trained by backpropagation like kernels
- soft-max: normalize input to sum = 1 (so the output can be interpreted as % probability)

→ if trained correctly CNN outputs probability of object class (e.g. 70% cat, 30% dog)
↳ needs MANY training samples (can use pretrained kernels of different CNN to speed up training/require less samples!)

- boosting classifier: use many classifiers that are better than chance to build a new one that's better than all other
  ↳ how? see Ada-Boost algorithm / Viola-Jones face detector (not in these notes!)

Object detection (like classification, but also determine location in image)

- sliding window: move detection window around in image and do class recognition at every move (SLOW!)
- R-CNN / fast R-CNN:
  1) use selective search to get ~2000 bounding box candidates
     do image segmentation with many techniques
     → generate box candidates from segments.
  2) run each bounding box candidate trough a CNN like for object class recognition.
     (optionally add bounding box regression where CNN is set up to propose corrections to bbox)

fast R-CNN: run most of the CNN on the whole image and only then cut out bbox candidates and run rest of CNN on these sub-"images"