FUNDAMENTALS

Grübler's Formula (DoF) [Prac Q1b, Q3d]

dof $= m(N-1-J) + \sum f_i$. m=3 planar, m=6 spatial.
N=#links (incl. ground), J=#joints, f_i=DoF of joint i.

Joint	f	c(2D)	c(3D)
Revolute/Prismatic	1	2	5
Helical	1	–	5
Cylindrical	2	–	4
Universal	2	–	4
Spherical	3	–	3

▶ **Redundant:** more DoF than task needs (Prac Q1b: **C**). ALOHA: N=15, J=16 (12R+4S), dof= 6(14)−(60+12) = 12.

Sensors [Prac Q1a]

	Proprioceptive	**Exteroceptive**
Active	Motor current	**LiDAR**, sonar
Passive	Encoder, IMU	Camera

LiDAR = **active, exteroceptive** (Prac Q1a: **D**).

Gear Ratio [Prac Q3a]

Ratio = $\frac{\text{driven teeth}}{\text{driving teeth}}$. Ratio n:1 → speed/n, torque×n.
Ex: 10 drives 40 → ratio=4:1, speed÷4, torque×4.

DH PARAMETERS [Prac Q4: 10pts]

Frame Assignment

z-axis: along **joint axis** (R: rotation, P: sliding).
x-axis: common normal from z_{i-1} to z_i.
If z's parallel: x_i from z_{i-1} toward z_i.
If intersect: $x_i = z_{i-1} \times z_i$.

4 DH Parameters

Param	**Definition**	**Axis**
α_{i-1}	$\angle$ from z_{i-1} to z_i	around x_{i-1}
a_{i-1}	dist z_{i-1} to z_i	along x_{i-1}
d_i	dist x_{i-1} to x_i	along z_i
θ_i	$\angle$ from x_{i-1} to x_i	around z_i

▶ **R** joint → θ_i variable. **P** joint → d_i variable.

DH Transformation Matrix

$T = \text{Rot}(x,\alpha)\cdot\text{Trans}(x,a)\cdot\text{Trans}(z,d)\cdot\text{Rot}(z,\theta)$

$$^{i-1}T_i = \begin{bmatrix} c\theta & -s\theta & 0 & a \\ s\theta c\alpha & c\theta c\alpha & -s\alpha & -s\alpha d \\ s\theta s\alpha & c\theta s\alpha & c\alpha & c\alpha d \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

FK: $^0T_n = {}^0T_1 \cdot {}^1T_2 \cdots {}^{n-1}T_n$. Last col = position.

RPR Example [Prac Q4a]

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1^*
2	90°	0	d_2^*	0
3	−90°	0	0	θ_3^*

*variable

If α=0: $s\alpha$=0, $c\alpha$=1. If α=90°: $s\alpha$=1, $c\alpha$=0.

DH FK Computation [Prac Q4b]

For joint 2 (α=90°, a=0, d=d_2, θ=0):

$$^1T_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & -d_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (c0{=}1,\ s0{=}0,\ c90{=}0,\ s90{=}1)$$

$^0T_3 = {}^0T_1 \cdot {}^1T_2 \cdot {}^2T_3$. Position = last column of result.

FK & IK [Prac Q4b–c]

2-Link Forward Kinematics

$x = l_1c_1 + l_2c_{12}$, $y = l_1s_1 + l_2s_{12}$
where $c_{12} = \cos(\theta_1{+}\theta_2)$, $s_{12} = \sin(\theta_1{+}\theta_2)$.

Inverse Kinematics

Step 1: $\cos\theta_2 = \frac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2}$ (law of cosines).
Two solutions: θ_2 and $-\theta_2$ (elbow up/down).
Step 2: $\theta_1 = \text{atan2}(y,x) - \text{atan2}(l_2s_2, l_1{+}l_2c_2)$.
▶ IK challenges: multiple solutions, singularities (θ_2=0, π), may be unreachable ($|c\theta_2|>1$).

IK Worked Example

Given l_1=l_2=1, reach (x,y)=(1,1):
$c\theta_2 = \frac{1+1-1-1}{2} = 0 \to \theta_2 = 90°$ (or −90°).
$\theta_1 = \text{atan2}(1,1) - \text{atan2}(1, 1+0) = 45° - 45° = 0°$.
Check: x= cos 0 + cos 90=1+0=1✓, y= sin 0 + sin 90=0+1=1✓.

Jacobian

$\dot{\mathbf{x}} = J\,\dot{\boldsymbol\theta}$. Singularity: $\det(J)$=$l_1l_2\sin\theta_2$=0 at θ_2=0, π.
At singularity: lose a DOF, can't move in some direction.

CAMERA [Prac Q5: 15pts]

Full Projection Equation

$$\boxed{s\begin{bmatrix} u \\ v \\ 1\end{bmatrix} = K\,[R\mid t]\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1\end{bmatrix}}$$

$s = Z_c$ (depth in camera frame).

Intrinsic Matrix K

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad f_x, f_y\text{=focal length (px)},$$

(u_0, v_0)=principal point.
Larger f → more zoomed. K doesn't change when camera moves.

Extrinsic $[R|t]$ **— world TO camera**

▶ $[R|t]$ is **NOT** camera pose! Camera position: $\mathbf{C} = -R^Tt$.

R=3×3 rotation (R^T=R^{-1}), t=3×1 translation.

Projection Steps [Prac Q5a–d]

1. World → Camera: $\mathbf{P}_c = R\,\mathbf{P}_w + t$
2. Normalize: $x_n = X_c/Z_c$, $y_n = Y_c/Z_c$
3. Pixel: $u = f_xx_n + u_0$, $v = f_yy_n + v_0$
Back-proj (need depth Z): $X = (u{-}u_0)Z/f_x$, $Y = (v{-}v_0)Z/f_y$.

Camera Projection Pipeline [Prac Q5a–d]

Given: depth cam pixel (u_d, v_d), depth D, K, $[R|t]$ to RGB cam.
Step 1 — Back-project to 3D: $\mathbf{P}_{3D} = D \cdot K^{-1}[u_d, v_d, 1]^T$
Step 2 — Transform to RGB frame: $\mathbf{P}_{rgb} = R\cdot\mathbf{P}_{3D} + t$
Step 3 — Project to RGB pixel: $s[u', v', 1]^T = K\cdot\mathbf{P}_{rgb}$
Step 4 — Find closest: $\arg\min_i \|(u',v') - (u_i^*, v_i^*)\|$

Depth & Calibration

Stereo: $Z = fB/d$ (disparity). FoV: $2\text{atan}(d/2f)$.
Calibration: min 6 point correspondences (11 unknowns).
Kabsch [Prac Q1f: **B,D**]: SVD aligns two sets of 3D points. H=P^TQ, USV^T=svd(H), R=$V\text{diag}(1,1,\det(VU^T))U^T$, t=$\bar q - R\bar p$.
Need ≥ 3 non-collinear 3D point correspondences.

CONTROL [Prac Q1d]

$$u(t) = K_pe(t) + K_i\int_0^t e(\tau)\,d\tau + K_d\dot e(t)$$

Term	**Effect**	**Too high**
P	Reduces error	Oscillate
I	Kills steady-state err	Overshoot
D	Damps oscillations	**Noise sensitive** (Q1d: **C**)

Open-loop: no feedback, drifts. Closed-loop: sensor feedback corrects.

MOTION PLANNING [PS3 Q3: 18pts]

Configuration Space

C_{free} = collision-free configs, C_{obs} = collision configs.
Planning converts any robot to a **point** in C-space.

PRM (Probabilistic Roadmap) — Multi-query

Phase 1 Build: sample N configs in C_{free}, connect nearby collision-free → undirected graph.
Phase 2 Query: connect start/goal to graph, run Dijkstra/A*.
▶ Prob. complete. Reusable for **many queries**. Bad for dynamic env.

RRT (Rapidly-exploring Random Tree) — Single-query

Tree from q_{start}. Loop: sample q_{rand}, find q_{near}, extend by step ε toward q_{rand} → q_{new}, add if collision-free.
$q_{new} = q_{near} + \varepsilon\cdot\frac{q_{rand}-q_{near}}{\|q_{rand}-q_{near}\|}$.
▶ Prob. complete. **NOT optimal**. Good for **single query**, high-dim.

PRM vs RRT [Prac Q1e: A,B]

	PRM	**RRT**
Structure	Undirected graph	Tree
Query	Multi-query	Single-query
Preprocess	Yes	No
Best for	Static, many queries	Dynamic, one-shot

PARTICLE FILTER [PS3 Q1: 33pts]

4-Step Algorithm

1. Initialize: scatter N particles randomly.
2. Move (Predict) [Q1e]: motion model + noise:
$\Theta' = \Theta + \mathcal{N}(0,\sigma_\Theta)$, $d' = d + \mathcal{N}(0,\sigma_d)$
$x_{t+1} = x_t + d'\cos\Theta'$, $y_{t+1} = y_t + d'\sin\Theta'$
▶ Robot **turns first** (Θ), **then moves** (x,y).
3. Update (Weight) [Q1f]: weight = P(reading | location).
Measurement model [Q1c: **B**] = P(sensor reading | robot at loc).
Gaussian PDF: closer match → higher weight.
Posterior via Bayes [Q1b: **C**]:

$$\boxed{P(\text{loc}|\text{read}) = \frac{P(\text{read}|\text{loc})\,P(\text{loc})}{P(\text{read})}}$$

4. Resample [Q1a: **A**]: draw N new particles ∝ weights.
▶ **Concentrate on high-prob particles** (discard low, duplicate high).
▶ $N_{eff} = 1/\sum w_i^2$. Use $\log(p)$ for underflow. Reset weights to $1/N$.

PF vs KF [PS3 Q1g — 10 pts!]

Concept	**PF**	**KF**
Belief w/ particles (samples)	✓	
Assumes Gaussian + linear		✓
Uses motion & meas. models	✓	✓
Handles nonlinear/non-Gaussian	✓	
Single mean + covariance		✓
Prediction-update (Bayes cycle)	✓	✓
Resampling needed	✓	
Optimal gain (Kalman gain)		✓
Requires initial estimate	✓	✓
Used for localization & SLAM	✓	✓

▶ PF advantage [Q1d: **A**]: handles **nonlinear** + **non-Gaussian**.

SLAM [PS3 Q2: 20pts, Prac Q6: 15pts]

What is SLAM? [Q2a: C]

Map unknown environment **while** tracking robot pose.
Chicken-and-egg: need map to localize, need location to map.

Why EKF, not KF? [Q2b: B, Q2c: B]

SLAM models are **nonlinear** → basic KF fails.
EKF = recursive estimator using **Jacobians** to **linearize**.

Loop Closure [Q2d: C]

Recognize previously visited location → **reduces uncertainty** in pose and map.

EKF-SLAM [Q2e, Prac Q2d fill-in]

State: $[x, y, \theta, x_1, y_1, \ldots, x_n, y_n]$. Cov: $(2n{+}3)\times(2n{+}3)$.
Prediction: uses **motion model** (fill-in: **L**).
Update: incorporates **sensor data** (fill-in: **K**).
Belief updated using **Bayes** rule (fill-in: **N**).

▶ Motion update: $\boxed{O(n^2)}$. Overall: $O(n^3)$.

FastSLAM [Prac Q6: 15 pts!]

Particle filter (trajectory) + local EKFs (landmarks).
Each particle has 1 EKF per landmark.
Total # EKFs = $M\times N$ [Q6b]. Ex: 50 × 20 = **1000**.
Complexity: $\boxed{O(M\cdot n)}$ per step [Q6c].

▶ Faster than EKF-SLAM when $\boxed{M \ll n}$ [Q6d].
Because $O(Mn) < O(n^2)$ when $M < n$.

	EKF-SLAM	**FastSLAM**
State	Joint (pose+map)	Factored
Belief	One Gaussian	Particles + EKFs
Per-step	$O(n^2)$	$O(M\cdot n)$
# EKFs	1 (size $2n$+3)	$M\times n$ (size 2)
Scales to	Small n	Large n

M=#particles, n=#landmarks.

MDPs [PS3 Q4: 20pts, Prac Q7: 15pts]

MDP = (S, A, P, R, γ)
S=states, A=actions, $P(s'|s,a)$=transition, $R(s,a,s')$=reward, γ=discount.
▶ **Markov Property** [Q4a: **A**, Q3b]: future depends **only on current state**.
Policy [Q4b: **C**]: mapping from states to actions. $\pi: S\to A$.
Goal [Q4c: **B**]: find π^* that maximizes expected cumulative reward.
Discount [Q4d: **A**]: γ<1 makes immediate rewards more important.
$\sum_{t=0}^{\infty}\gamma^t = \frac{1}{1-\gamma}$.

Value Functions

$V^\pi(s)$=expected return from s following π.
$Q^\pi(s,a)$=start at s, take a, then follow π.
$V^*(s) = \max_a Q^*(s,a)$. $\pi^*(s) = \arg\max_a Q^*(s,a)$.

Bellman Equations [Prac Q7a, PS3 Q4e]

Fixed π: $V^\pi(s) = \sum_{s'} P(s'|s,\pi(s))[R + \gamma V^\pi(s')]$

Optimal: $\boxed{V^*(s) = \max_a \sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V^*(s')}$

Value Iteration [Prac Q2c fill-in: I]

Init V_0=0. Repeat: $V_{k+1}(s) = \max_a \sum_{s'} P[R + \gamma V_k(s')]$.
Extract: $\pi^*(s) = \arg\max_a \sum P[R + \gamma V^*]$.
Cost/iter: $O(|S|^2|A|)$. ▶ Supports **early stopping** (monitor V convergence).

Policy Iteration [Prac Q2c fill-in: J]

(a) Evaluation: solve V^π exactly: $O(|S|^3)$ (linear system).
(b) Improvement: $\pi_{new}(s) = \arg\max_a \sum P[R + \gamma V^\pi(s')]$: $O(|S|^2|A|)$.
▶ Requires **full policy eval** before assessing convergence.
Fewer iterations but each costlier. Often **faster in practice**.

Matching [Prac Q7a — 5 pts]

Term	**Equation form**
Bellman Eq	$V(s) = \max_a \sum P[R + \gamma V(s')]$
Value Iter	$V_k(s) = \max_a \sum P[R + \gamma V_k(s')]$
Policy Extract	$\pi(s) = \arg\max_a \sum P[R + \gamma V^\pi(s')]$
Policy Eval	$V_{k+1}^\pi(s) = \sum P(s'|s,\pi(s))[R + \gamma V_k^\pi(s')]$
Policy Improve	$\pi_{new}(s) = \arg\max_a \sum P[R + \gamma V^{\pi old}(s')]$

Worked Examples [Prac Q7c–d]

Q-value (deterministic): $Q(s,a) = R(s,a) + \gamma V(s')$.
Ex: $Q(s,A)$=3+0.8×10=11, $Q(s,B)$=2+0.8×20=18. Best: **B**.

Self-loop: $V(s) = R + \gamma V(s) \to \boxed{V(s) = \frac{R}{1-\gamma}}$.

Ex: R=4, γ=0.95 → V = 4/0.05 = **80**.
Stochastic Q: $Q(s,a)$=$\sum_{s'} P(s'|s,a)[R(s,a,s') + \gamma V(s')]$.
Ex: $P(s_1)$=0.7, $P(s_2)$=0.3: Q=0.7[R_1+γV_1] + 0.3[R_2+γV_2].
▶ Smaller γ → myopic (near-sighted). Larger γ → values future more.

IMITATION LEARNING [PS3 Q5: 9pts]

Three Approaches [Prac Q2a fill-in]

Method	Learns	Fill-in
BC	Policy $\pi(a\|s)$ directly	**A**
DMP	Trajectory (attractor sys.)	**C**
IRL	Reward $R(s)$	**B**

Behavioral Cloning

Supervised learning: $\pi_\theta(s) \approx \pi^*(s)$.
▶ **Compounding error** [PS3 Q5a: **A**]: small mistakes → unseen states → more errors → crash. Error grows $O(\varepsilon T^2)$.
DAgger: run learned π, query expert at visited states, retrain.

BC vs IRL [PS3 Q5b: D, Q5c: C]

BC copies **actions** directly (supervised). Assumes expert optimal.
IRL infers **reward function**, then optimizes policy via RL. IRL more robust, transfers better.
▶ Q5c: BC assumes **expert actions are optimal**; IRL relaxes this.

DMPs (Dynamic Movement Primitives)

$\tau\dot{v} = K(g-x) - Dv + (g-x_0)f(s)$, $\tau\dot{s} = -\alpha s$ (s: $1 \to 0$).
$f(s) = \frac{\sum w_i \psi_i(s) s}{\sum \psi_i(s)}$. As $s \to 0$: $f \to 0$ → spring-damper converges to g.
Weights learned via linear regression. Temporal/spatial invariance.

Imitation Challenges [Prac Q3c — 4 pts]

1. Compounding error / distribution shift (see above).
2. Multimodal demonstrations: different demos show different strategies for same state (e.g., go left OR right around obstacle). Unimodal policy (Gaussian) → **averages modes** → dangerous (goes straight into obstacle).
Solutions: GMMs, **Diffusion Policy** (generative, handles multiple modes).

Monte Carlo [Prac Q2b fill-in]

Estimate via **random sampling** (**E**) + **statistical analysis** (**F**).

CONVOLUTION [Prac Q1c]

Slide kernel over image, element-wise multiply, sum → 1 output pixel.
$h[m,n] = \sum_{k,l} g[k,l]\, f[m+k, n+l]$. Output: $(W-K+1)\times(H-K+1)$.
Box $\frac{1}{9}\begin{pmatrix}1&1&1\\1&1&1\\1&1&1\end{pmatrix}$ → blur. **Sobel V:** $\begin{pmatrix}1&0&-1\\2&0&-2\\1&0&-1\end{pmatrix}$ detects vertical edges.
Sobel H: $\begin{pmatrix}1&2&1\\0&0&0\\-1&-2&-1\end{pmatrix}$ detects horizontal edges.
Sharpening: 2×original − blurred → amplifies edges.
Gaussian: $G_\sigma = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}$, smoother blur than box.
▶ Prac Q1c kernel: negative top, positive bottom → **horizontal edge detect** (Sobel-H variant) → answer shows horizontal edges.
Conv ex: 3×3 image with Sobel-H center pixel:
$o = (-1)(a_{00}) + (-2)(a_{01}) + (-1)(a_{02}) + (1)(a_{20}) + (2)(a_{21}) + (1)(a_{22})$
Large $|o|$ → strong horizontal edge. $o>0$ → dark-to-light going down.

VI vs PI COMPARISON [PS3 Q4f]

	Value Iteration	**Policy Iteration**
Updates	V values only	π and V
Per-iter	$O(\|S\|^2\|A\|)$	$O(\|S\|^3 + \|S\|^2\|A\|)$
# Iters	Many (slow conv.)	Few (fast conv.)
Converges to	V^* directly	π^* directly
Simplicity	Simpler	More complex
In practice	—	Often **faster**
Early stop	**Yes** (monitor V)	**No** (need full eval)

▶ VI: one Bellman backup per iteration (approx.).
PI: **exact** policy eval + greedy improvement, guaranteed to converge in finite steps (finite # policies).

ROTATION & TRIG REFERENCE

2D Rotation Matrix

$R(\theta) = \begin{bmatrix}\cos\theta & -\sin\theta\\ \sin\theta & \cos\theta\end{bmatrix}$ Rotates point by θ CCW. $R^{-1} = R^T = R(-\theta)$.

3D Elementary Rotation Matrices

$R_x(\alpha) = \begin{bmatrix}1&0&0\\0&c\alpha&-s\alpha\\0&s\alpha&c\alpha\end{bmatrix}$ $R_y(\beta) = \begin{bmatrix}c\beta&0&s\beta\\0&1&0\\-s\beta&0&c\beta\end{bmatrix}$
$R_z(\gamma) = \begin{bmatrix}c\gamma&-s\gamma&0\\s\gamma&c\gamma&0\\0&0&1\end{bmatrix}$
▶ R_y has $+s\beta$ top-right, $-s\beta$ bottom-left (opposite of R_x, R_z!).
▶ Trick: the "1" is on the axis you rotate **about** (x → row/col 1, y → 2, z → 3).

Rotation Properties

$\det(R)=1$, $R^TR=I$, $R^{-1}=R^T$. Group: SO(3).
Composition: $R_{total} = R_z \cdot R_y \cdot R_x$ (right-to-left = intrinsic).
▶ $R_A \cdot R_B \neq R_B \cdot R_A$ — rotations **don't commute!**

Euler Angles (ZYX convention)

Any 3D rotation = $R = R_z(\alpha)\, R_y(\beta)\, R_x(\gamma)$ (yaw-pitch-roll).
Yaw α: rotation about z (heading).
Pitch β: rotation about y (nose up/down).
Roll γ: rotation about x (tilt left/right).
▶ **Gimbal lock** at pitch = ±90°: lose 1 DoF (yaw and roll become same axis).

Axis-Angle (Rodrigues' Formula)

Rotate by angle θ about unit axis $\hat{\omega} = [\omega_x, \omega_y, \omega_z]^T$:
$R = I + \sin\theta\,[\hat{\omega}]_\times + (1-\cos\theta)\,[\hat{\omega}]_\times^2$
where skew-symmetric: $[\hat{\omega}]_\times = \begin{bmatrix}0&-\omega_z&\omega_y\\ \omega_z&0&-\omega_x\\ -\omega_y&\omega_x&0\end{bmatrix}$
Special cases: $\theta=0 \to R=I$. $\theta=\pi \to R=2\hat{\omega}\hat{\omega}^T - I$.

Homogeneous Transform (4×4)

$T = \begin{bmatrix}R & t\\ 0 & 1\end{bmatrix}$, $T^{-1} = \begin{bmatrix}R^T & -R^Tt\\ 0 & 1\end{bmatrix}$.
Chain: ${}^AT_C = {}^AT_B \cdot {}^BT_C$.

Trig Values (no calculator!)

θ	0	30	45	60	90
sin	0	$\frac{1}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$	1
cos	1	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$	0

$\sin(-\theta) = -\sin\theta$, $\cos(-\theta) = \cos\theta$.
$\sin(90°) = 1$, $\cos(90°) = 0$, $\sin(-90°) = -1$.

KEY DEFINITIONS [Fill-in Targets]

EKF [PS3 Q2b]: Recursive estimator for **nonlinear** systems; linearizes models via **Jacobians** to apply Kalman filter updates.
Markov [PS3 Q4a]: Given current state, future is **independent of history**.
IK challenges [Prac Q4c]: Joint limits, singularities, arm redundancy, multiple solutions, workspace limits.
Meas. model [PS3 Q1c]: P(sensor reading | robot at location).
Loop closure [PS3 Q2d]: Re-observing known landmark → corrects accumulated drift.
C-space [PS3 Q3d]: Space of all robot configurations (joint angles). Obstacles mapped to C-space.

RRT step: $q_{new} = q_{near} + \varepsilon \cdot \frac{q_{rand} - q_{near}}{\|q_{rand} - q_{near}\|}$.

FORMULA QUICK REF

DH: $T = \text{Rot}(x,\alpha)\text{Trans}(x,a)\text{Trans}(z,d)\text{Rot}(z,\theta)$
IK: $c\theta_2 = \frac{x^2+y^2-l_1^2-l_2^2}{2l_1l_2}$; $\theta_1 = \text{atan2}(y,x) - \text{atan2}(l_2s_2, l_1+l_2c_2$
Camera: $s[u,v,1]^T = K[R|t][X,Y,Z,1]^T$. Pose: $\mathbf{C} = -R^Tt$
Bellman: $V^* = \max_a \sum_{s'} P(s'|s,a)[R+\gamma V^*(s')]$
Q-val: $Q(s,a) = R(s,a) + \gamma V(s')$. **Self-loop:** $V = R/(1-\gamma)$
PF Bayes: $P(\text{loc}|\text{r}) = \frac{P(\text{r}|\text{loc})P(\text{loc})}{P(\text{r})}$. Meas. model $= P(\text{r}|\text{loc})$
PF motion: $x' = x + d'\cos\Theta'$, $y' = y + d'\sin\Theta'$ (turn first, then move)
SLAM: EKF $O(n^2)$. Fast $O(Mn)$. EKFs= $M\times n$. Fast wins: $M \ll n$
Stereo: $Z = fB/d$. **FoV:** $2\text{atan}(d/2f)$. **Gear:** $n{:}1 \to \text{spd}/n$, $\text{trq}\times n$
Grübler: $\text{dof} = m(N-1-J) + \sum f_i$
DMP: $\tau\dot{v} = K(g-x) - Dv + (g-x_0)f(s)$; $f = \frac{\sum w_i\psi_i s}{\sum \psi_i}$; $\tau\dot{s} = -\alpha s$
Kabsch: $H = P^TQ$, svd→ $R = V\text{diag}(1,1,\det(VU^T))U^T$, $t = \bar{q} - R\bar{p}$
Singularity: $\det(J) = l_1l_2\sin\theta_2 = 0$ at $\theta_2 = 0, \pi$
Conv: $h[m,n] = \sum g[k,l]f[m+k,n+l]$. **Gauss:** $\frac{1}{2\pi\sigma^2}e^{-(x^2+y^2)/2\sigma^2}$

Common Pitfalls & Exam Tips

▶ **DH:** R joint → θ varies; P joint → d varies. Don't mix up!
▶ **Camera:** $[R|t]$ transforms world → camera, NOT camera pose. Camera position $= -R^Tt$.
▶ **IK:** Always check $|c\theta_2| \leq 1$ (reachable?). Two solutions: elbow up/down.
▶ **PF motion:** Robot **turns first** (θ), **then moves** (x,y). Each particle gets different noise.
▶ **Resampling:** Concentrates particles, does NOT increase total count. Weights reset to $1/N$.
▶ **MDP:** V^* gives value of best action. Q^* gives value of specific action. π^* tells you which action.
▶ **VI vs PI:** VI can stop early (check ΔV). PI must finish full evaluation before checking convergence.
▶ **Discount:** γ close to 0 → greedy/myopic. γ close to 1 → values future. γ=1 → may not converge.
▶ **SLAM:** EKF-SLAM = one big joint state. FastSLAM = factored (particles + small EKFs).
▶ **BC vs IRL:** BC copies actions (fast but brittle). IRL learns **why** (reward), then finds new policy.

How to Solve Common Problem Types

Grübler: Count links (incl. ground), count joints, identify types → plug in.
DH table: Assign frames (z=joint axis, x=common normal), read off α, a, d, θ.
FK: Plug DH params into T matrix for each joint, multiply ${}^0T_1 \cdot {}^1T_2 \cdots$
IK: Use law of cosines for θ_2, then atan2 for θ_1. Check reachability.
Camera proj: Back-project (need depth!) → transform frame → project with K → divide by Z_c.
PF predict: Apply motion + noise to each particle independently.
PF update: Weight = how well particle's expected reading matches actual reading.
MDP Q-value: $Q = R + \gamma V(s')$ if deterministic. $Q = \sum P(s')[R+\gamma V(s')]$ if stochastic.
Self-loop: $V = R + \gamma V \to V = R/(1-\gamma)$. Always check for this pattern!
Kabsch: Center points, compute $H = P^TQ$, SVD, assemble R, then $t = \bar{q} - R\bar{p}$.