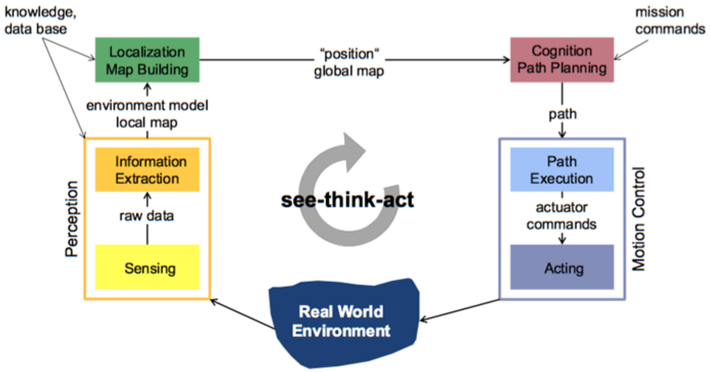
Autonomous Mobile Robots

Fabian Blöchliger, *Spring Semester 2016*

[**Github:** fabianbl/amr2016_summary]

1 Introduction and Motivation



2 Locomotion Concepts

Express point P which is given w.r.t body frame B in inertial frame I:

$$ {}_I\mathbf{r}_{OP} = {}_I\mathbf{r}_{OB} + \mathbf{R}_{IB}\ {}_B\mathbf{r}_{BP}. $$

Equivalent **homogeneous transformation** description:

$$ \begin{bmatrix} {}_I\mathbf{r}_{OP} \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_{IB} & {}_I\mathbf{r}_{OB} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}_B\mathbf{r}_{BP} \\ 1 \end{bmatrix} = \mathbf{H}_{IB} \cdot {}_B\tilde{\mathbf{r}}_{BP}. $$

Velocity of rigid body point P:

$$ {}_I\mathbf{v}_P = {}_I\dot{\mathbf{r}}_{OP} = \dot{\mathbf{r}}_{OB} + {}_I\omega_{IB} \times {}_I\mathbf{r}_{BP}. $$

Differentiation in moving frame (**Coriolis equation**):

$$ {}_B\mathbf{v}_P = {}_B\left[\dot{\mathbf{r}}_{OP}\right] = \frac{\mathrm{d}\,{}_B\mathbf{r}_{OP}}{\mathrm{d}\,t} + {}_B\omega_{IB} \times {}_B\mathbf{r}_{OP} $$

Basic rotation matrices $\mathbf{R}_x(\bullet)$, $\mathbf{R}_y(\bullet)$, $\mathbf{R}_z(\bullet)$:

$$ \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos & -\sin \\ 0 & \sin & \cos \end{bmatrix}, \begin{bmatrix} \cos & 0 & \sin \\ 0 & 1 & 0 \\ -\sin & 0 & \cos \end{bmatrix}, \begin{bmatrix} \cos & -\sin & 0 \\ \sin & \cos & 0 \\ 0 & 0 & 1 \end{bmatrix}. $$

Jacobian (partial derivative of position vector $\mathbf{r}(\mathbf{q})$ w.r.t. **generalized coordinate** vector $\mathbf{q}$):

$$ \mathbf{J} = \frac{\partial \mathbf{r}(\mathbf{q})}{\partial \mathbf{q}} = \begin{bmatrix} \frac{\partial r_1}{\partial q_1} & \cdots & \frac{\partial r_1}{\partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial q_1} & \cdots & \frac{\partial r_m}{\partial q_n} \end{bmatrix}. $$

Left/right pseudoinverse for $m \times n$ matrix $\mathbf{J}$ to solve $\mathbf{r}_F = \mathbf{J}_F\mathbf{q}$ for $\mathbf{q}$:

$$ \mathbf{J}^+ = \begin{cases} (\mathbf{J}^\intercal\mathbf{J})^{-1}\mathbf{J}^\intercal, & m > n \ \text{(overdetermined)}, \\ \mathbf{J}^\intercal(\mathbf{J}\mathbf{J}^\intercal)^{-1}, & m < n \ \text{(underdetermined)}. \end{cases} $$

Iterative approach for **inverse kinematics** of robotic manipulator to find generalized coordinates for end-effector position $\mathbf{r}^{\text{goal}}$ (Newton's method):

$\mathbf{q} = \mathbf{q}^0,\ \mathbf{r} = \mathbf{r}(\mathbf{q})$
while $\|\mathbf{r} - \mathbf{r}^{\text{goal}}\| >$ threshold **do**
  $\mathbf{q} = \mathbf{q} + \mathbf{J}^+(\mathbf{q}) \cdot (\mathbf{r}^{\text{goal}} - \mathbf{r}),\quad \mathbf{r} = \mathbf{r}(\mathbf{q})$

Inverse **differential kinematics** (get desired end-effector velocity $\dot{\mathbf{r}}_\mathbf{F}$):

$$ \dot{\mathbf{r}}_F = \mathbf{J}_F\dot{\mathbf{q}} \quad \rightarrow \quad \dot{\mathbf{q}} = \mathbf{J}_F^+\mathbf{r}_F. $$

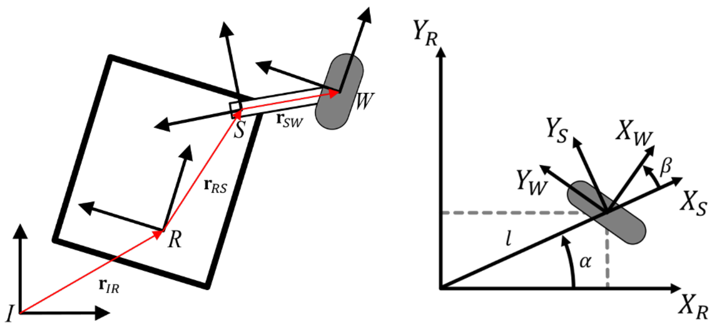
3 Mobile Robot Kinematics

General wheel equation ($\mathbf{v}_{IR}, \omega_{IR}$: linear / angular robot velocity, ω_{RS}: steer rate, $\mathbf{r}_{SW}$: wheel offset):

$$ \mathbf{v}_{IW} = \mathbf{v}_{IR} + \omega_{IR} \times [\mathbf{r}_{RS} + \mathbf{r}_{SW}] + \omega_{RS} \times \mathbf{r}_{SW}. $$

Standard wheel equation (no wheel offset $\mathbf{r}_{SW}$):

$$ \mathbf{v}_{IW} = \mathbf{v}_{IR} + \omega_{IR} \times \mathbf{r}_{RS}. $$



Rolling constraint (${}_W\mathbf{v}_{IW} = [0, -r\dot{\varphi}, 0]^\intercal$):

$$ \begin{bmatrix} \sin(\alpha+\beta) & -\cos(\alpha+\beta) & (-l)\cos(\beta) \end{bmatrix} R(\theta)\dot{\xi}_I = r\dot{\varphi}. $$

No-sliding constraint:

$$ \begin{bmatrix} \cos(\alpha+\beta) & \sin(\alpha+\beta) & l\sin(\beta) \end{bmatrix} R(\theta)\dot{\xi}_I = 0. $$

Robot state: $\xi_I = \begin{bmatrix} x & y & \theta \end{bmatrix}^\intercal$, $\dot{\xi}_R = R(\theta)\dot{\xi}_I$, $R(\theta) = \mathbf{R}_z^\intercal(\theta)$.

Stacked equations of motion for a $(N_f + N_s)$-wheeled robot:

$$ \begin{matrix} \textbf{(rolling)} \\ \textbf{(no-sliding)} \end{matrix} \quad \begin{bmatrix} J_1(\beta_s) \\ C_1(\beta_s) \end{bmatrix} R(\theta)\dot{\xi}_I = \begin{bmatrix} J_2 \\ 0 \end{bmatrix}\dot{\varphi}, \quad \dot{\varphi} = \begin{bmatrix} \dot{\varphi}_1 .. \dot{\varphi}_N \end{bmatrix}, $$

with

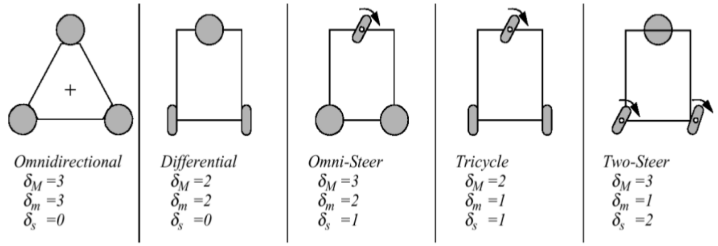
$$ J_1(\beta_s) = \begin{bmatrix} J_{1f} \\ J_{1s}(\beta_s) \end{bmatrix}, J_2 = \mathrm{diag}(r_1..r_N), C_1(\beta_s) = \begin{bmatrix} C_{1f} \\ C_{1s}(\beta_s) \end{bmatrix}. $$

A robot's **degree of maneuverability** φ_M is

$$ \delta_M = \delta_m + \delta_s, $$

which is the sum of its **degree of mobility** φ_m and its **degree of steerability** φ_s:

$\delta_m = \dim \mathrm{N}\left[C_1(\beta_s)\right] = 3 - \mathrm{rank}\left[C_1(\beta_s)\right],\ \varphi_s = \mathrm{rank}\left[C_{1s}(\beta_s)\right].$



Forward/inverse kinematics of a **differential drive robot**:

$$ \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}_R = \begin{bmatrix} \frac{r}{2} & \frac{r}{2} \\ 0 & 0 \\ \frac{r}{2b} & -\frac{r}{2b} \end{bmatrix} \begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix} \ / \ \begin{bmatrix} \dot{\varphi}_r \\ \dot{\varphi}_l \end{bmatrix} = \begin{bmatrix} \frac{1}{r} & 0 & \frac{b}{r} \\ \frac{1}{r} & 0 & -\frac{b}{r} \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix}_R. $$

4 Perception I

Sensor Type	System	Class
Tactile	Bumpers	EC, P
Wheel/motor	Brush encoders	PC, P
	Optical encoders	PC, A
Heading	Compass	EC, P
	Gyroscope	PC, P
	Inclinometer	EC, A/P
Acceleration	Accelerometer	PC, P
Beacons	GPS	EC, A
	Radio, ultrasonic, Reflective Beacons	EC, A
Motion/speed	Doppler: radar/sound	EC, A
Range	Ultrasound, laser, struct. light, ToF	EC, A
Vision	CCD/CMOS	EC, P

(PC = **proprioceptive**, EC = **exteroceptive**, A = active, P = passive)

Range sensors: Traveled distance d of a sound or electromagnetic wave after a **time of flight** t is given by

$$ d = ct, \quad c = 0.3\,\mathrm{m/ms}\,\text{(sound)}\ /\ 0.3\,\mathrm{m/ns}\,\text{(light)}. $$

Phase-shift measurement between transmitted and reflected laser beam (D': total distance, λ: modulating wavelength, f: modulating frequency, θ: phase difference):

$$ D' = 2D = \tfrac{\theta}{2\pi}\lambda. $$

Optical triangulation (1D): Determine object distance as

$$ D = f\tfrac{L}{x}, \quad L: \text{distance laser/PSD}. $$

The **error propagation law** describes the mapping from input covariance C_X to output covariance C_Y using the Jacobian F_X of the mapping function $f(\bullet): \mathbb{R}^n \to \mathbb{R}^m$ w.r.t. X:

$$ C_Y = F_X C_X F_X^\intercal \quad \text{(linear approximation)}. $$

5 Perception II

Thin lens equation: Voxel at depth z will be **focused** on the **focal plane** at distance e behind the lens for a camera with **focal length** f. If the **image plane** lies at $e \pm \delta$, the voxel image will be a **blur circle** of radius R:

$$\frac{1}{f} = \frac{1}{z} + \frac{1}{e}, \quad R = \frac{L\delta}{2e} \quad (L: \text{ diameter of lens/aperture}).$$

Pinhole approximation: $z \gg f$, therefore $e \approx f$ (lens is approximated as pinhole at distance f from image plane).

Perspective projection: A 3D-point $P_C = [X_C, Y_C, Z_C]^\intercal$ (in camera frame C) projects onto the image location $[u, v]^\intercal$ as

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \alpha_u X_C/Z_C + u_0 \\ \alpha_v Y_C/Z_C + v_0 \end{bmatrix} \quad / \quad \lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}}_{\text{calibr. matrix } K} \begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix},$$

with $[\alpha_u, \alpha_v] = f[k_u, k_v]$, using the inverse of the effective pixel size k_u (k_v) in [pixel/m] and the pixel coordinates of the **optical center** $[u_0, v_0]^\intercal$. K contains the **intrinsic parameters**.

Radial distortion model:

$$\begin{bmatrix} u_\text{d} \\ v_\text{d} \end{bmatrix} = (1 + k_1 r^2) \begin{bmatrix} u - u_0 \\ v - v_0 \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}, \quad r^2 = (u - u_0)^2 + (v - v_0)^2.$$

For a **general perspective projection**, P_W is given w.r.t. world frame W and the transform from W to C is described by the **extrinsic parameters** $[R|T]$.

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \left(R \begin{bmatrix} X_W \\ Y_W \\ Z_W \end{bmatrix} + T \right) = \underbrace{K[R|T]}_{\substack{\text{camera} \\ \text{matrix } P}} \begin{bmatrix} X_W \\ Y_W \\ Z_W \\ 1 \end{bmatrix}.$$

Basic stereo camera setup with **baseline** b and focal length f, the depth Z for a point at left/right coordinate $[u_l, u_r]$ is

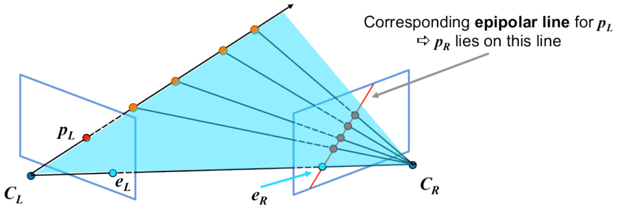
$$Z = bf/d, \quad d = u_l - u_r \text{ (disparity)}.$$

Cross-product written with skew-symmetric matrix:

$$\mathbf{a} \times \mathbf{b} = [\mathbf{a}_\times]\mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}.$$

Epipolar constraint (p_1, p_2: normalized, homogeneous):

$$p_2^\intercal E p_1 = 0, \quad E = [T_\times]R \text{ (essential matrix)}.$$



6 Perception III

Correlation with a filter/kernel/mask F of size $(2N+1)$ is

$$J(x) = F \circ I(x) = \sum_{i=-N}^{N} F(i) I(x+i),$$

Convolution is correlation with a flipped filter/image:

$$J(x) = F * I(x) = \sum_{i=-N}^{N} F(i) I(x-i).$$

In contrast to correlation, convolution is associative.

Linear filters replace every pixel by a linear combination of its neighbors. **Shift-invariant filters** perform the same operation on every point of the image.

The **2D Gaussian kernel** is a **separable** filter (width σ):

$$G_\sigma = (x, y) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-\frac{x^2+y^2}{2\sigma^2}} = \underbrace{\frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{-x^2}{2\sigma^2}}}_{g_\sigma(x)} \cdot \underbrace{\frac{1}{\sigma\sqrt{2\pi}} \mathrm{e}^{-\frac{-y^2}{2\sigma^2}}}_{g_\sigma(y)}.$$

Laplacian of Gaussian (second derivative operator):

$$\text{LoG} = \nabla^2 G_\sigma(x, y) = \frac{\partial^2 G_\sigma(x,y)}{\partial x^2} + \frac{\partial^2 G_\sigma(x,y)}{\partial y^2}.$$

Difference of Gaussian is an approximation of LoG:

$$\text{DoG} = G_{k\sigma}(x, y) - G_\sigma(x, y).$$

Template matching using **sum of squared differences**:

$$\text{SSD}(x) = \sum_{i=-N}^{N} [F(i) - I(x+i)]^2,$$
$$= \sum [F(i)]^2 + \sum [I(x+i)]^2 - 2\sum [F(i) I(x+i)].$$

Zero-mean normalized cross correlation (ZNCC): Invariant to local average intensity. Maximize:

$$\text{ZNCC}(x) = \frac{\sum_i [F(i) - \mu_F][I(x+i) - \mu_{I_x}]}{\sqrt{\sum_i [F(i) - \mu_F]^2} \sqrt{\sum_i [I(x+i) - \mu_{I_x}]^2}},$$

$$\mu_F = \frac{\sum_i F(i)}{2N+1}, \quad \mu_{I_x} = \frac{\sum_i I(x+i)}{2N+1}, \quad i = -N..N.$$

Roberts/Prewitt/Sobel masks (approx. derivatives):

$$I_x = \begin{array}{|c|c|} \hline 0 & -1 \\ \hline 1 & 0 \\ \hline \end{array} \Big/ \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \Big/ \begin{array}{|c|c|c|} \hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}, \quad I_y = \begin{array}{|c|c|} \hline -1 & 0 \\ \hline 0 & 1 \\ \hline \end{array} \Big/ \begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array} \Big/ \begin{array}{|c|c|c|} \hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}.$$

The **second order matrix** used by the **Harris corner detector** (image patch size P):

$$M = \sum_{x,y \in P} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}, \quad \text{SSD}(\Delta x, \Delta y) \approx [\Delta x \; \Delta y] \, M \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix}.$$

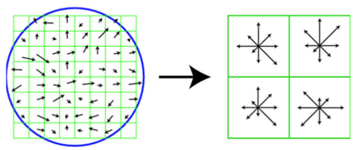
Corners: Local maxima in the **cornerness function**

$$C = \lambda_1 \lambda_2 - \kappa(\lambda_1 + \lambda_2)^2 = \det M - \kappa \operatorname{trace}^2 M.$$

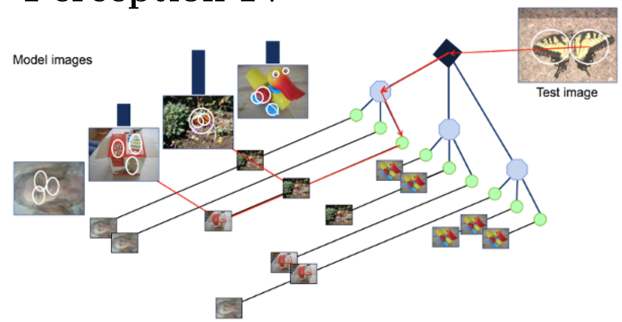
Main **SIFT** stages:

1. Extract keypoints + scale.
2. Assign keypoint orientation.
3. Generate descriptor (4·4·8).

[4. Matching (L_2 distance).]



7 Perception IV



Vector quantization by **k-means clustering**, minimizes squared Euclidean distance between points and their nearest cluster-centers:

```
randomly initialize k cluster centers
while not converged do
    assign each vector to nearest center
    re-compute cluster centers
```

Term frequency-inverse document frequency (tf-idf): Measures the importance of visual word inside database as $\mathit{tf\text{-}idf} = \mathit{tf}_{ij} \cdot \mathit{idf}_i$ with

$$\mathit{tf}_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}}, \quad \mathit{idf}_i = \log \frac{|\text{num. images}|}{|\text{num. images with } w_i|}.$$

Line extraction algorithms:

A1) **Split-and-merge:**
1. Fit line through point set, find most distant point P.
2. If $d_P >$ threshold, split set at P. Repeat for all sets.
3. If two consecutive segments are collinear enough, merge.

A2) **Line regression:**
1. Initialize sliding window size N_f.
2. Fit a line to every N_f consecutive points.
3. Merge overlapping line segments and re-compute line parameters for each segment.

A3) **RANSAC:**
1. Randomly select 2 points and fit a line through them.
2. Compute distances of all points to this line, select inliers.
3. Iterate k times. Estimate $k = \log(1-p)/\log(1-w^2)$ (p: probability of finding set free of outliers, w: fraction of inliers).

A4) **Hough transform:**
1. For each point (x, y), compute $\rho = x\cos\theta + y\sin\theta$ for $\theta = [0..180]$. Increment according array entries.
2. Find local maxima (θ, ρ).

8 Localization I

The **mean/expectation value** $\mathrm{E}[x] = \mu$ and the **variance** $\mathrm{Var}[x] = \sigma^2$ of a **continuous random variable** x with **probability density function (PDF)** $p(x)$ are computed as

$$\mu = \int_{-\infty}^{\infty} x p(x) \mathrm{d}x, \quad \sigma^2 = \int_{-\infty}^{\infty} = (x-\mu)^2 p(x) \mathrm{d}x.$$

Sum rule (1) and **product rule** (2):

$$(1)\ p(x) = \textstyle\sum_y p(x,y), \quad (2)\ p(x,y) = p(y|x)p(x).$$

Combine them to get the **theorem of total probability**:

$$p(x) = \textstyle\sum_y p(y|x)p(x).$$

Assuming that $p(y) > 0$, **Bayes' rule** is

$$p(x|y) = \tfrac{p(y|x)p(x)}{p(y)} = \eta\, p(y|x)p(x), \quad \eta = p(y)^{-1}.$$

PDF for one-dimensional **Gaussian distribution** $\mathcal{N}(\mu, \sigma^2)$:

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

Multivariate Gaussian distribution $\mathcal{N}(\mu, \Sigma)$ for dimension k with (symmetric) covariance matrix Σ:

$$p(x) = \tfrac{1}{(2\pi)^{k/2}\det(\Sigma)^{1/2}} \exp\left[-\tfrac{1}{2}(x-\mu)^\intercal \Sigma^{-1}(x-\mu)\right].$$

Combination of GRVs: Let $y = Ax_1 + Bx_2$ be a linear function of $x_i = \mathcal{N}(\mu_i, \Sigma_i)$. Then, $p(y)$ is

$$p(y) = \mathcal{N}(A\mu_1 + B\mu_2,\ A\Sigma_1 A^\intercal + B\Sigma_2 B^\intercal).$$

If $y = f(x_1, x_2)$ is non-linear, approximate y and $p(y)$ as

$$y \approx f(\mu_1, \mu_2) + F_{x_1}(x_1 - \mu_1) + F_{x_2}(x_2 - \mu_2),$$
$$p(y) \approx \mathcal{N}(f(\mu_1, \mu_2),\ F_{x_1}\Sigma_1 {F_{x_1}}^\intercal + F_{x_2}\Sigma_2 {F_{x_2}}^\intercal).$$

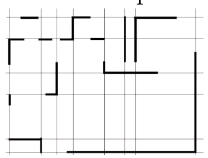
A robot's **belief** about its state x_t before/after measurement z_t is represented as probability distribution:

$$\overline{bel}(x_t) = p(x_t|z_{1\to t-1}, u_{1\to t}), \quad bel(x_t) = p(x_t|z_{1\to t}, u_{1\to t}).$$

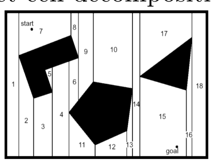
Classification of localization problems:

- **Position tracking:** $bel(x_0)$ is **Dirac delta** function.
- **Global localization:** Uniform distribution for $bel(x_0)$).
- **Kidnapped robot problem:** Does the robot realize?

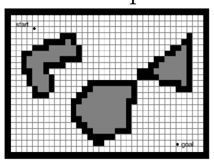
Architecture map:



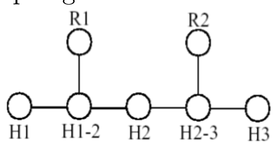
Exact cell decomposition:



Approx. decomposition:



Topological:



9 Localization II

Ingredients of probabilistic **map-based localization**:

1. The initial probability distribution $bel(x_0)$.
2. *True* **map** $M = \{m_0..m_n\}$ of the environment.
3. Data: u_t (proprioceptive, control), z_t (exteroceptive).
4. Probabilistic **motion model** $p(x_t|u_t, x_{t-1})$, e.g. based on noise-free model $x_t = f(x_{t-1}, u_t)$.
5. Probabilistic **measurement model** $p(z_t|x_t, M)$, e.g. based on noise-free model $z_t = h(x_t, M)$.

According to the **Markov assumption**, the robot's belief state $bel(x_t)$ is a function only of robot's previous state x_{t-1} and its most recent actions u_t and observations z_t:

$$p(x_t|x_0, u_0 \cdots u_t, z_0 \cdots z_t) = p(x_t|x_{t-1}, u_t, z_t).$$

The general algorithm for **Markov localization**:

for all x_t **do**
$\overline{bel}(x_t) = \sum_{x_{t-1}} p(x_t|u_t, x_{t-1}) bel(x_{t-1})$ (prediction)
$bel(x_t) = \eta p(z_t|x_t, M)\overline{bel}(x_t)$ (measurement)

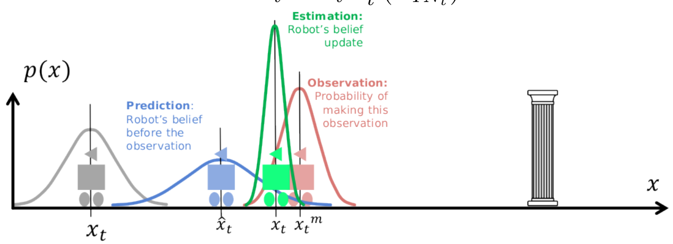
Kalman filter localization assumes $bel(x_t) = \mathcal{N}(x_t, P_t)$.

S1) The **prediction update** is (Q_t: covariance of motion model noise, $F_{x/u}$: jacobian w.r.t. x/u):

$$\hat{x}_t = f(x_{t-1}, u_t), \quad \hat{P}_t = F_x P_{t-1} F_x^\intercal + F_u Q_t F_u^\intercal.$$

S2) The **measurement update** consists of four steps:

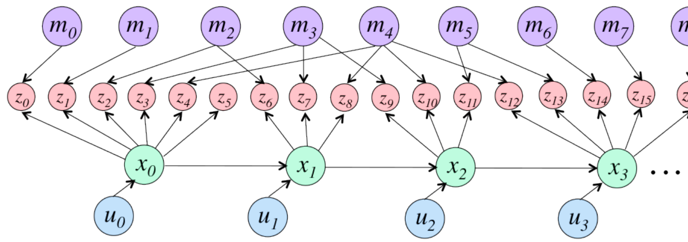
1. *Observation*: Obtain z_t^i with covariance R_t^i $(i = 1..n)$.
2. *Measurement prediction*: Predict $\hat{z}_t^j = h^j(\hat{x}_t, m^j)$, compute its jacobian H^j w.r.t $\hat{x}_t$.
3. *Matching step*: Compute the **innovation** (**covariance**)
$$v_t^{ij} = [z_t^i - \hat{z}_t^j], \quad \Sigma_{IN_t}^{ij} = H^j \hat{P}_t {H^j}^\intercal + R_t^i,$$
Find matches with a **validation gate** g, e.g. Mahalanobis distance: ${v_t^{ij}}^\intercal (\Sigma_{IN_t}^{ij})^{-1} v_t^{ij} \le g^2$.
4. *Estimation step*: Stack validated observations into z_t, corresponding innovations into v_t, measurement jacobians into H_t and $R_t = \mathrm{diag}(R_t^i)$, compute Σ_{IN_t}. Update the robot's state estimate as
$$x_t = \hat{x}_t + K_t v_t, \quad P_t = \hat{P}_t - K_t \Sigma_{IN_t} K_t^\intercal,$$
with the **Kalman gain**
$$K_t = \hat{P}_t H_t^\intercal (\Sigma_{IN_t})^{-1}.$$



10 SLAM I

Predecessors of SLAM:

- **Photogrammetry:** Use (aerial) photographs to make measurements between points, recover exact positions.
- **Structure from Motion (SfM):** Estimate 3D structure from a sequence of images.



Full SLAM: Given the the landmark observations $\{z_0..z_k\}$ and the control inputs $\{u_0..u_t\}$, estimate the joint posterior probability over the robot path $\{x_0..x_t\}$ and the *true* map $\{m_0..m_{n-1}\}$, i.e. find

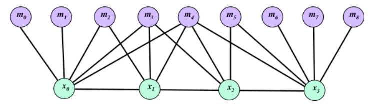
$$p(x_{0:t}, m_{0:n-1}|z_{0:k}, u_{0:t}).$$

Online SLAM: Recover the posterior for the current pose

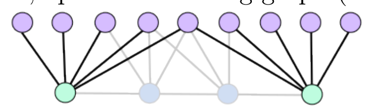
$$p(x_t, m_{0:n-1}|z_{0:k}, u_{0:t}).$$

Approaches:

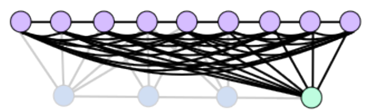
1) **Full graph optimization** (bundle adjustment): Eliminate observations and control inputs, solve for constraints between poses and landmarks. Sparsify graph for real-time application.



2) **Key frames:** Retain most representative poses and their dependency links, optimize resulting graph (e.g. PTAM).



3) **Filtering:** Summarize all experience w.r.t. last pose with state vector and covariance matrix (e.g. MonoSLAM, FastSLAM).



11 SLAM II

EKF SLAM summarizes past experience in an extended state vector y_t and a corresponding covariance P_{y_t}:

$$y_t = \begin{bmatrix} x_t \\ m_1 \\ \cdots \\ m_{n-1} \end{bmatrix}, \quad P_{y_t} = \begin{bmatrix} P_{xx} & \cdots & P_{xm_{n-1}} \\ \vdots & \ddots & \vdots \\ P_{m_{n-1}x} & \cdots & P_{m_{n-1}m_{n-1}} \end{bmatrix}.$$

S1) Prediction according to EKF equations:

$$\hat{y}_t = \begin{bmatrix} \hat{x}_t \\ m_i \end{bmatrix} = \begin{bmatrix} f(x_{t-1}, u_t) \\ \mathbf{0} \end{bmatrix}, \quad \hat{P}_{y_t} = F_y P_{y_{t-1}} F_y^\intercal + F_u Q_t F_u^\intercal.$$

S2) Measurement model $\hat{z}_i = h^i(\hat{x}_t, m_i)$ as in EKF localization. Update the state with actual observations $z_{0:n-1}$:

$$y_t = \hat{y}_t + K_t v_t, \; P_{y_t} = \hat{P}_{y_t} - K_t \Sigma_{IN} K_t^\intercal,$$

$$v_t = z_{0:n-1} - h_{0:n-1}(\hat{x}_t, m_{0:n-1}), \; \Sigma_{IN} = H \hat{P}_{y_t} H^\intercal + R,$$

using the Kalman gain $K_t = \hat{P}_{y_t} H (\Sigma_{IN})^{-1}$.

MonoSLAM: EKF SLAM implementation. Motion model:

$$\mathbf{f}_v = \begin{bmatrix} \mathbf{r}^W_{\text{new}} \\ \mathbf{q}^{WR}_{\text{new}} \\ \mathbf{v}^W_{\text{new}} \\ \omega^W_{\text{new}} \end{bmatrix} = \begin{bmatrix} \mathbf{r}^W + (\mathbf{v}^W + \mathbf{V}^W)\Delta t \\ \mathbf{q}^{WR} \times \mathbf{q}((\omega^W + \mathbf{\Omega}^W)\Delta t) \\ \mathbf{v}^W + \mathbf{V}^W \\ \omega^W + \mathbf{\Omega}^W \end{bmatrix},$$

where the unknown linear and angular accelerations $\begin{bmatrix} \mathbf{V}^W & \mathbf{\Omega}^W \end{bmatrix} = \begin{bmatrix} \mathbf{a}^W \Delta t & \alpha^W \Delta t \end{bmatrix}$ cause an impulse in velocity.

Current challenges in vision-based robotic perception:

1. High-fidelity localization and mapping.
2. Dense scene reconstruction.
3. Place recognition.
4. Collaborative robot sensing and mapping.
5. Navigation (obstacle avoidance, path planning).

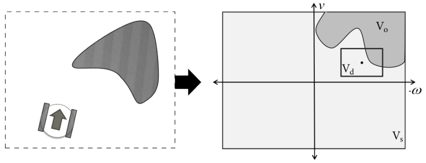
12 Planning I

Dynamic window approach (DWA): Acts in input space (v, ω). Set of admissible velocities within next time frame is

$$V_r = \underbrace{V_o}_{\substack{\text{grid, obstacles} \\ \text{(in input-space)}}} \cap \underbrace{V_s}_{\substack{\text{static window} \\ \text{(motor limits etc.)}}} \cap \underbrace{V_d}_{\substack{\text{dynamic window} \\ \text{(feasible within next frame)}}}.$$

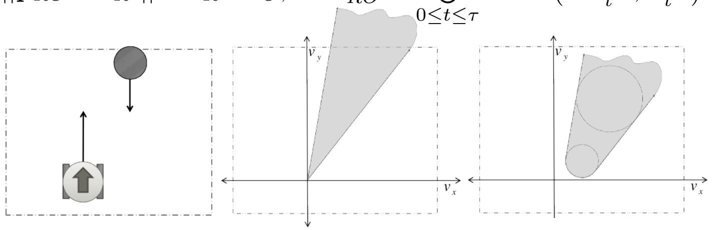
Maximize $(v, \omega) \in V_r$, subject to (local) objective function

$$\begin{aligned} O = & a \cdot \text{heading}(v, \omega) \\ & + b \cdot \text{velocity}(v, \omega) \\ & + c \cdot \text{obst\_dist}(v, \omega). \end{aligned}$$



Velocity obstacles (VO): 1) Compute set of colliding velocities. 2) Restrict to set of colliding velocities within time horizon τ. 3) Shift VO by instantaneous obstacle velocity.

$$||\mathbf{p}_{RO} + \mathbf{v}_R t|| < r_R + r_O, \quad \text{VO}^\tau_{RO} = \bigcup_{0 \le t \le \tau} \text{Disks}\left(-\tfrac{\mathbf{p}_{RO}}{t}, \tfrac{r_{RO}}{t}\right).$$



Interactive collision avoidance for multiple decision-making agents without explicit communication, using a fairness property: **reciprocal velocity obstacles**.

Local potential fields: Robot (position $\mathbf{q}$) follows gradient of potential field. Potential attractive at goal, repulsive at obstacles ($U_{\text{rep}} = 0$ if distance to obstacle $\rho(\mathbf{q}) > \rho_{\text{lim}}$):

$$U_{\text{att}}(\mathbf{q}) = \tfrac{1}{2} k_{\text{att}} (\mathbf{q} - \mathbf{q}_{\text{goal}})^2, \; U_{\text{rep}}(\mathbf{q}) = \tfrac{1}{2} k_{\text{rep}} \left(\tfrac{1}{\rho(\mathbf{q})} - \tfrac{1}{\rho_{\text{lim}}}\right)^2.$$

Harmonic potential fields: Iteratively solve the **Laplace equation** $\triangle U = \sum \frac{\partial^2 U}{\partial q_i^2} = 0$ with the approximation $\nabla U(\mathbf{q})_i \approx [U(\mathbf{q} + \delta e_i) - U(\mathbf{q})]/\delta$ as

$$U^{k+1}(\mathbf{q}) = \tfrac{1}{2n} \sum_{i=1}^n (U^k(\mathbf{q} + \delta \mathbf{e}_i) + U^k(\mathbf{q} - \delta \mathbf{e}_i)),$$

with a mixture of the following boundary conditions concerning the obstacle boundaries:

- **Neumann:** Equipotential lines orthogonal.
- **Dirichlet:** Constant potential.

Additional obstacle avoidance examples:

- **Bug algorithm:** Follow the boundary of an obstacle, depart from point of shortest distance to the goal.
- **Vector field histogram (VFH):** Use polar histogram showing probability of obstacle occurrence.
- **Bubble band technique:** Compute bubbles representing max. free space around given robot configuration.

13 Planning II

Road map approaches for construction of graph $G(N, E)$ with nodes N and edges E:

- **Visibility graph:** Connect nodes (obstacle corners) that see each other. Optimal solutions w.r.t path length.
- **Voronoi diagram:** Evaluate equidistant paths to obstacles, workspace needs to be closed.

General **deterministic graph search** algorithm:

Queue.init(), Queue.push(Start)
while Queue **is not** empty **do**
  Node curr = Queue.pop()
  if curr **is** Goal **return**
  Closed.push(curr)
  Nodes next = expand(curr)
  for all next **not in** Closed **do**
    Queue.push(next)

Total expected cost from start to goal via node N:

$$f(N) = \underbrace{g(N)}_{\substack{\text{cost so far} \\ \text{(from start)}}} + \varepsilon \cdot \underbrace{h(N)}_{\substack{\text{heuristic} \\ \text{cost-to-go}}}.$$

Examples:

1) **Breadth-first search**: FIFO queue, solution optimal for uniform edge costs, complexity: $\mathcal{O}(|N| + |E|)$.
2) **Depth-first search**: LIFO queue, solution not optimal.
2) **Dijkstra's search**: ordering according to $f(N)$ with $\varepsilon = 0$, complexity: $\mathcal{O}(|N| \log |N| + |E|)$.
3) **A* algorithm**: extension of Dijkstra with $\varepsilon = 1$. Possible heuristic $h(N)$: Euclidean distance to goal (underestimation).
4) **D* algorithm**: incremental replanning version of A*.

Binary min-heap: Top node has minimum value.

- *Push(N)*: Insert new node N at end of heap. While $N <$ parent(N), swap N and parent(N).
- *Pop()*: Return top element of heap. Move bottom node N to top. While $N <$ min(child(N)), swap these nodes.

Rapidly exploring random tree (RRT) algorithm as an example of a randomized graph search method:

Graph.init(Start)
while Graph.size() **is less than** threshold **do**
  Node rand = **rand()**
  Node near = Graph.nearest(rand)
  try
    Node new = Sys.propagate(near, rand)
    Graph.addNode(new)
    Graph.addEdge(near, new)