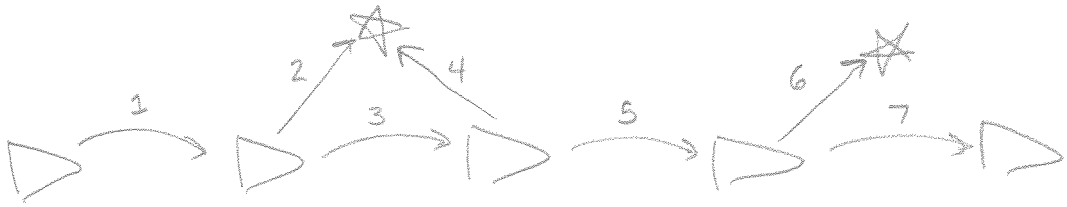
SLAM example



Our approach today:

- Incrementally update $J^T W J$, use dense methods.
  - Pros: makes it "easy" to forget old things.
  - Cons: can't relinearize. Sparsity will be compromised if we marginalize.

$$\chi^2 = \sum_i (z_i(x) - \hat{z}_i)^T \Sigma_i^{-1} (z_i(x) - z_i)$$

Sum of squared errors.

$$z_i(x) \approx J_i \Delta x + b_i$$

1st-order Taylor expn.

so:

$$z_i(x) - z_i = J_i \Delta x + b_i - z_i$$

let $r_i = z_i - b_i$ (our residual obs. − predicted)

$$= J_i \Delta x - r_i$$

Note: J_i is our hyperplane! A robot is a hyperplane collecting machine

$$\chi^2 = \sum_i (J_i \Delta x - r_i)^T \Sigma_i^{-1} (J_i \Delta x - r_i)$$

...(minimize by diff. WRT Δx, set to zero)...

$$\underbrace{\left(\sum_i J_i^T \Sigma_i^{-1} J_i\right)}_{\text{A matrix we can build up one hyperplane (or hyperplanes corresponding to 1 obs) at a time}} \Delta x = \underbrace{\left(\sum_i J_i^T \Sigma_i^{-1} r_i\right)}_{\text{A vector.}}$$

We will maintain 3 quantities:

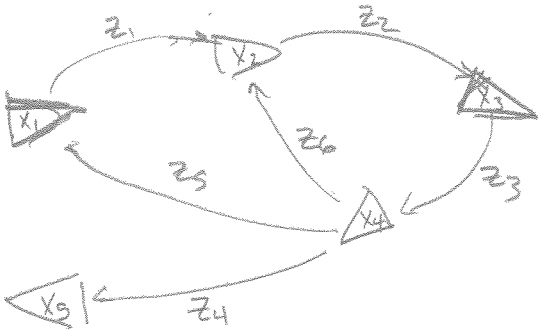
> the info matrix

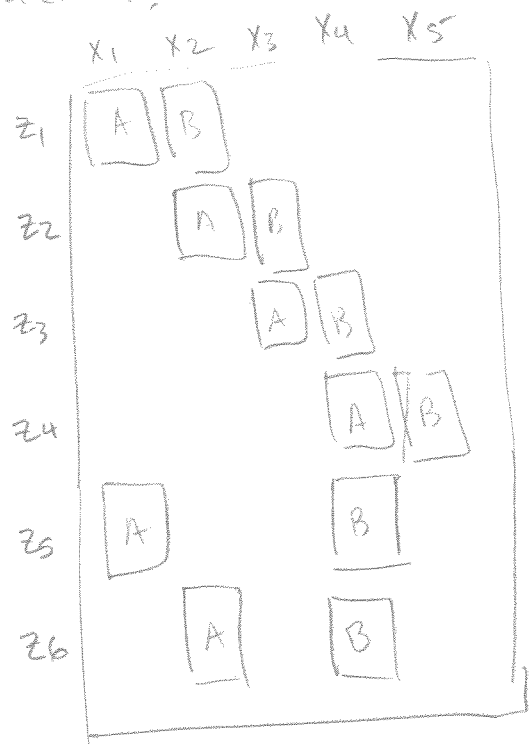
> the RHS

> x.

Structure of SLAM

Jacobian has a lot of structure.





Suppose we're in 2D $\begin{bmatrix} x \\ y \\ \theta \end{bmatrix}$ — what are the dimensions of the blocks & what do they mean?

(3x3, & Jacobian of obs WRT variables).

We can draw the SLAM graph from the Jacobian!

Information matrix — sum of Inf matrix of each eqn. Each eqn gives 4 non-zero blocks

$$J = \begin{bmatrix} A & B \end{bmatrix} \qquad J^TJ = \begin{bmatrix} A^TA & A^TB \\ B^TA & B^TB \end{bmatrix}$$

$$J^TJ = \begin{bmatrix} A^T \\ B^T \end{bmatrix} \begin{bmatrix} A & B \end{bmatrix}$$

$$A = \frac{\partial z}{\partial x_i} \qquad B = \frac{\partial z}{\partial x_j}$$

Step 0.

Initialization. Approach 1: Include start state.

$$x \sim N(0, 0)$$

This is problematic in general. Suppose we're just "awfully sure" so:

$$x \sim N(0, 0.01).$$

Don't get carried away — condition # of matrix can get bad!

Approach 2: Skip initialization — anytime you have a reference to x_0, just plug in $x_0 = 0$.

(equiv. to computing $p(x \mid x_0 = 0)$

all the state except x_0.

Pro: avoids singularity problems.

Con: special cases all over (whenever ref state x_0).

We'll use approach 1.

Step 1.

Odometry.

- Model? $z = x_1 - x_0 + \omega_1 \quad = 1$ Actual observed value.

  $\omega_1 \sim N(0, \Sigma_1) = N(0, 1)$

This is the first time x_1 appears. Must <u>Grow</u> the state estimate.

enlarge state vector

$$\begin{bmatrix} \boxed{} \\ \boxed{} \end{bmatrix}$$

old state (just 0 for us)

initial guess for x_1 (how about 1!)

$$= \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

enlarge info. matrix

$$\begin{bmatrix} \text{old info matrix, (100)} & 0 \\ 0 & \boxed{0} \end{bmatrix}$$

Exactly as though $\vec{x}$ was longer & J_i's had extra zeros tacked on. At this point, it's rank-deficient.

- Now, add new hyperplanes by linearizing model WRT Δx.

$$\underbrace{\begin{bmatrix} -1 & 1 \end{bmatrix}}_{J_i} \Delta x = \overbrace{\underbrace{1}_{\text{obs}} - \underbrace{1}_{\text{initial guess}}}^{r_i}$$

(we got lucky!)

$$J_i^T \Sigma_i^{-1} J_i = \begin{bmatrix} 1 & -1 \\ -1 & +1 \end{bmatrix} \quad \text{so, } \Sigma^{-1} = \begin{bmatrix} 101 & -1 \\ -1 & 1 \end{bmatrix}$$

(assume $\Sigma_i^{-1} = 1$ for laziness)

Similarly, update RHS: $J_i^T \Sigma_i^{-1} r_i$

Step 1, with cov. projection

Also could have done this via cov. projection!

$$x' = f(x) = \begin{bmatrix} x_0 \\ 1 + x_0 - w_1 \end{bmatrix} \quad \leftarrow \text{solve obs. model for } x_1$$

$$E[x'] = E\begin{bmatrix} x_0 \\ 1 + x_0 - w_1 \end{bmatrix} = E\begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\mathrm{VAR}[x'] = J_x^f \Sigma_x J_x^{f\,T} + J_w^f \Sigma_w J_w^{f\,T}$$

$$J_x^f = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \qquad \Sigma_x = [100]^{-1} \qquad J_w^f = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \qquad \Sigma_w = 1$$

$$= \begin{bmatrix} 100^{-1} & 100^{-1} \\ 100^{-1} & 100^{-1} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 0{,}01 & 0{,}01 \\ 0{,}01 & 1{,}01 \end{bmatrix}}_{\Sigma_{x'}}$$

And yes, $\Sigma_{x'}^{-1} = \begin{bmatrix} 101 & -1 \\ -1 & 1 \end{bmatrix}$, from the other method,

Step 2

We observe a landmark.

Model? $z_2 = f - x_1 + w_2 = 0.6 \quad w_2 \sim N(0, 1)$

x_1: our current position

f: landmark (feature) position

Just like step 1, this is our first reference to state element f. Expand the state info:

$$x' = \begin{bmatrix} 0 \\ 1 \\ \boxed{} \end{bmatrix}$$

0: our previous state mean

$\boxed{}$: initial guess for f. (1.6!)

Note it's up to us to keep track of which indices in x are robot positions and/or landmarks.

$$\Sigma_{x'}^{-1} = \begin{bmatrix} \Sigma_x^{-1} & 0 \\ 0 & 0 \end{bmatrix}$$

old Σ_x^{-1}

Add new hyperplane by linearizing WRT Δx:

$$J_2 = \begin{bmatrix} 0 & -1 & 1 \end{bmatrix} \qquad r_2 = \begin{bmatrix} 0 \end{bmatrix}$$

our initial estimate again matches the obs. (by construction. This generally happens when expanding the state vector)

$$J_2 \Sigma_2^{-1} J_2^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

$$\text{So } \Sigma^{-1} = \begin{bmatrix} 101 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

Consider Σ: $\begin{bmatrix} .01 & .01 & .01 \\ .01 & 1.01 & 1.01 \\ .01 & 1.01 & 2.01 \end{bmatrix}$

f "inherited" the uncertainty of x_1. Makes sense!

(could also have used cov. projection)

Step 3) another odom, handled just like step 1.

$$x = \begin{bmatrix} 0 \\ 1 \\ 1.6 \\ 2 \end{bmatrix} \qquad \Sigma_x = \begin{bmatrix} .01 & .01 & 0.01 & .01 \\ .01 & 1.01 & 1.01 & 1.01 \\ .01 & 1.01 & 2.01 & 1.01 \\ .01 & 1.01 & 1.01 & 2.01 \end{bmatrix}$$

Note x_2 "inherits" x_1's uncertainty, not f's!

Step 4) We observe a feature!

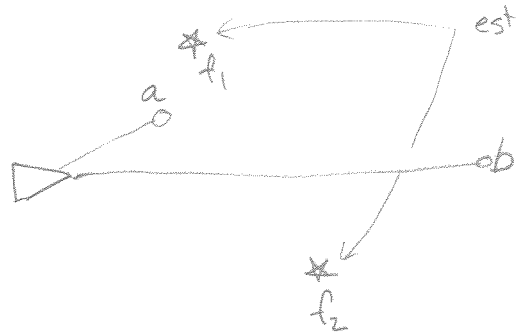
$$\text{Model} = f - x_2 = \underbrace{-0.5}_{\text{observed value}}$$

$$\text{Predicted location} = x_2 - 0.5 = 1.5.$$

Critical Question: is this the same feature we saw before? A new feature?

Data-Association

We've mostly assumed that we know which feature is which. What if we don't?



a & b are obs without known feature association.

Euclidean nearest-neighbor

- match each obs to closest landmark.

+ easy / fast
- doesn't use uncertainties
- would a & b both be assoc. w the same feature?
  - sometimes, this is sensible. Depends on sensor properties.
  - when it's not, use greedy matching or stable marriage.

Is it a new landmark?

Greedy Matching

1. Consider all possible matches between A & B. (n^2 distances).
2. Sort.
3. while (more matches)
   - m = next best match,
   - if m_a and m_b have not yet been matched to any other features, accept m.

Robust? No

Doing Assoc "Right"

$p(z_1 \,\&\, z_2 \text{ are same object})$

$p(\text{assoc})$

$$\iint p(\text{assoc}, z, m) \, dz \, dm$$

$$\iint p(\text{assoc} \mid z, m) \, p(z \mid m) \, p(m) \, dz \, dm$$

Hopeless!

Forgetting

Marginalize