Probabilistic Robotics Course

Least Squares Introduction ICP Optimization

Giorgio Grisetti
grisetti@dis.uniroma1.it

Dept of Computer Control and Management Engineering
Sapienza University of Rome

Maximum Likelihood Estimation

Using

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmax}}\, p(\mathbf{x}|\mathbf{z})$$

- Bayes' Rule

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})}$$

$$\propto p(\mathbf{z}|\mathbf{x})$$

- Independence,

$$= \prod_i p(\mathbf{z}^{[i]}|\mathbf{x})$$

We can further simplify the task

Gaussian Assumption

Measurements affected by Gaussian noise

$$\begin{aligned} p(\mathbf{z}_i|\mathbf{x}) &= \mathcal{N}(\mathbf{z}^{[i]}; \mathbf{h}^{[i]}(\mathbf{x}), \mathbf{\Sigma}^{[i]}) \\ &\propto \exp\left(-(\underbrace{\mathbf{h}^{[i]}(\mathbf{x}) - \mathbf{z}^{[i]}}_{\mathbf{e}^{[i]}(\mathbf{x})})^T \underbrace{\mathbf{\Sigma}^{[i]-1}}_{\mathbf{\Omega}^{[i]}} (\mathbf{h}^{[i]}(\mathbf{x}) - \mathbf{z}^{[i]}) \right) \end{aligned}$$

Gaussian Assumption

Measurements affected by Gaussian noise

prediction

measurement

state

$$
\begin{aligned}
p(\mathbf{z}_i|\mathbf{x}) \quad &= \quad \mathcal{N}(\mathbf{z}^{[i]}; \mathbf{h}^{[i]}(\mathbf{x}), \mathbf{\Sigma}^{[i]}) \\
&\propto \quad \exp\left(-\underbrace{(\mathbf{h}^{[i]}(\mathbf{x}) - \mathbf{z}^{[i]})}_{\mathbf{e}^{[i]}(\mathbf{x})}{}^{T} \underbrace{\mathbf{\Sigma}^{[i]-1}}_{\mathbf{\Omega}^{[i]}} (\mathbf{h}^{[i]}(\mathbf{x}) - \mathbf{z}^{[i]}) \right)
\end{aligned}
$$

error function

Gaussian Assumption

Through Gaussian assumption

- Maximization becomes minimization
- Product turns into sum

$$
\begin{aligned}
\mathbf{x}^* &= \operatorname*{argmax}_x \prod_i p(\mathbf{z}^{[i]} | \mathbf{x}) \\
&= \operatorname*{argmax}_x \prod_i \exp(-\mathbf{e}^{[i]}(\mathbf{x})^T \mathbf{\Omega}^{[i]} \mathbf{e}^{[i]}(\mathbf{x})) \\
&= \operatorname*{argmin}_x \sum_i \mathbf{e}^{[i]}(\mathbf{x})^T \mathbf{\Omega}^{[i]} \mathbf{e}^{[i]}(\mathbf{x})
\end{aligned}
$$

Gauss Method Overview

Iterative minimization of

$$F(\mathbf{x}) \quad = \quad \sum_i \mathbf{e}^{[i]}(\mathbf{x})^T \boldsymbol{\Omega}^{[i]} \mathbf{e}^{[i]}(\mathbf{x})$$

Each iteration refines the current estimate by applying a perturbation

$$\mathbf{x} \quad \leftarrow \quad \mathbf{x} + \boldsymbol{\Delta}\mathbf{x}$$

Perturbation obtained by minimizing a quadratic approximation of the problem in $\boldsymbol{\Delta}\mathbf{x}$

$$F(\mathbf{x} + \boldsymbol{\Delta}\mathbf{x}) \quad \simeq \quad \boldsymbol{\Delta}\mathbf{x}^T \mathbf{H} \boldsymbol{\Delta}\mathbf{x} + 2\mathbf{b}^T \boldsymbol{\Delta}\mathbf{x} + c$$

Linearization

The quadratic approximation is obtained by linearizing the error functions around $\mathbf{x}$

$$\mathbf{e}^{[i]}(\mathbf{x}^* + \mathbf{\Delta x}) \simeq \underbrace{\mathbf{e}^{[i]}(\mathbf{x}^*)}_{\mathbf{e}} + \underbrace{\left.\frac{\partial \mathbf{e}^{[i]}(\mathbf{x})}{\partial(\mathbf{x})}\right|_{\mathbf{x}=\mathbf{x}^*}}_{\mathbf{J}^{[i]}} \mathbf{\Delta x}$$

...expanding the products

$$\mathbf{e}^{[i]}(\mathbf{x}^* + \mathbf{\Delta x})^T \mathbf{\Omega}^{[i]} \mathbf{e}^{[i]}(\mathbf{x}^* + \mathbf{\Delta x}) \simeq$$

$$\mathbf{\Delta x}^T \underbrace{\mathbf{J}^{[i]T} \mathbf{\Omega}^{[i]} \mathbf{J}^{[i]}}_{\mathbf{H}^{[i]}} \mathbf{\Delta x} + 2 \underbrace{\mathbf{J}^{[i]T} \mathbf{\Omega}^{[i]} \mathbf{e}^{[i]}}_{\mathbf{b}^{[i]T}} \mathbf{\Delta}_x + \underbrace{\mathbf{e}^{[i]T} \mathbf{\Omega}^{[i]} \mathbf{e}^{[i]}}_{c^{[i]}}$$

...and grouping the terms

$$\mathbf{H} = \sum_i \mathbf{H}^{[i]} \qquad \mathbf{b} = \sum_i \mathbf{b}^{[i]} \qquad c = \sum_i c^{[i]}$$

Quadratic form

Find the $\mathbf{\Delta x}$ that minimizes the quadratic approximation of the objective function

$$\begin{aligned}\mathbf{\Delta x}^* &= \operatorname*{argmin}_{\mathbf{\Delta x}} F(\mathbf{x}^* + \mathbf{\Delta x}) \\ &\simeq \operatorname*{argmin}_{\mathbf{\Delta x}} \mathbf{\Delta x}^T \mathbf{H} \mathbf{\Delta x} + 2\mathbf{b}^T \mathbf{\Delta x} + c\end{aligned}$$

Find $\mathbf{\Delta x}$ that nulls the derivative of quadratic form

$$\begin{aligned}\mathbf{0} &= \frac{\partial \left[\mathbf{\Delta x}^T \mathbf{H} \mathbf{\Delta x} + 2\mathbf{b}^T \mathbf{\Delta x} + c\right]}{\partial \mathbf{\Delta x}} \\ -\mathbf{b} &= \mathbf{H} \mathbf{\Delta x}\end{aligned}$$

Algorithm (one Iteration)

Clear **H** and **b**

$$\mathbf{H} \leftarrow 0 \qquad \mathbf{b} \leftarrow 0$$

For each measurement, update h and b

$$\begin{aligned}
\mathbf{e}^{[i]} &\leftarrow \mathbf{h}^{[i]}(\mathbf{x}^*) - \mathbf{z}^{[i]} \\
\mathbf{J}^{[i]} &\leftarrow \left.\frac{\partial \mathbf{e}^{[i]}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\mathbf{x}^*} \\
\mathbf{H} &\leftarrow \mathbf{H} + \mathbf{J}^{[i]T}\boldsymbol{\Omega}^{[i]}\mathbf{J}^{[i]} \\
\mathbf{b} &\leftarrow \mathbf{b} + \mathbf{J}^{[i]T}\boldsymbol{\Omega}^{[i]}\mathbf{e}^{[i]}
\end{aligned}$$

Update the estimate with the perturbation

$$\begin{aligned}
\boldsymbol{\Delta}\mathbf{x} &\leftarrow \mathrm{solve}(\mathbf{H}\boldsymbol{\Delta}\mathbf{x} = -\mathbf{b}) \\
\mathbf{x}^* &\leftarrow \mathbf{x}^* + \boldsymbol{\Delta}\mathbf{x}
\end{aligned}$$

Methodology

Identify the state space **X**

- Qualify the domain
- Find a locally Euclidean parameterization

Identify the measurement space(s) **Z**

- Qualify the domain
- Find a locally Euclidean parameterization

Identify the prediction functions **h(x)**

Gauss-Newton in SLAM

Typical problems where GN is used

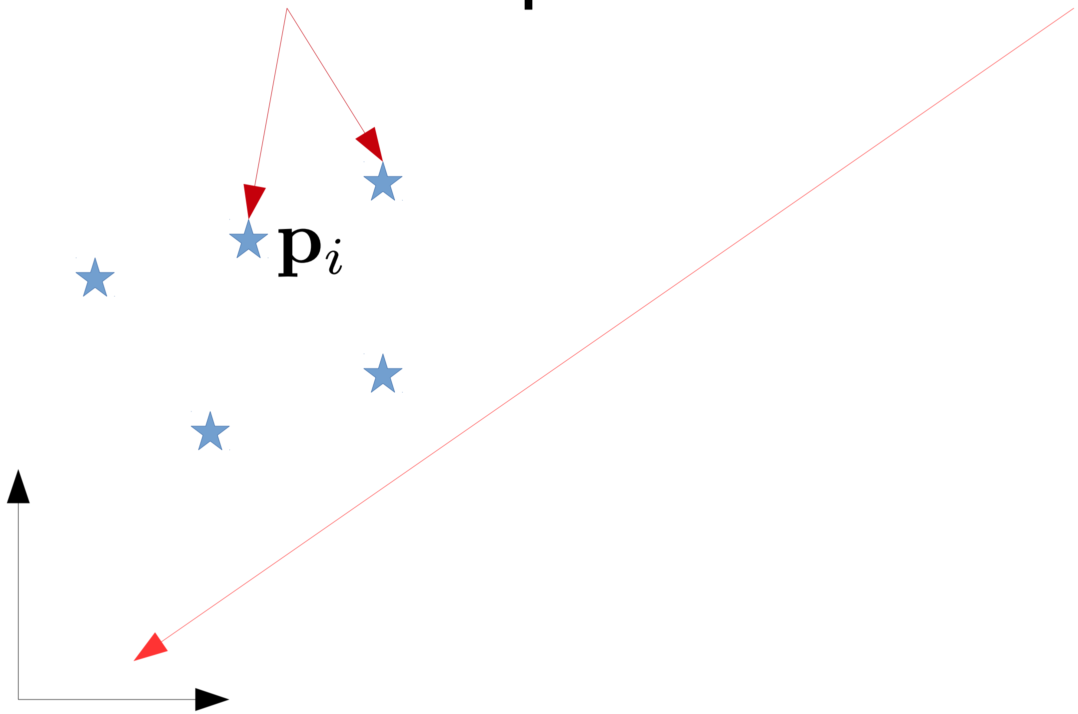
- Calibration
- Registration
  - Cloud to Cloud (ICP)
  - Image to Cloud (Posit)
- Global Optimization
  - Pose-SLAM
  - Bundle Adjustment

Warning

- **Data association is assumed to be known known**
- **Gauss-Newton alone is not sufficient to solve a full problem**
- **One needs a strategy to compute data association**

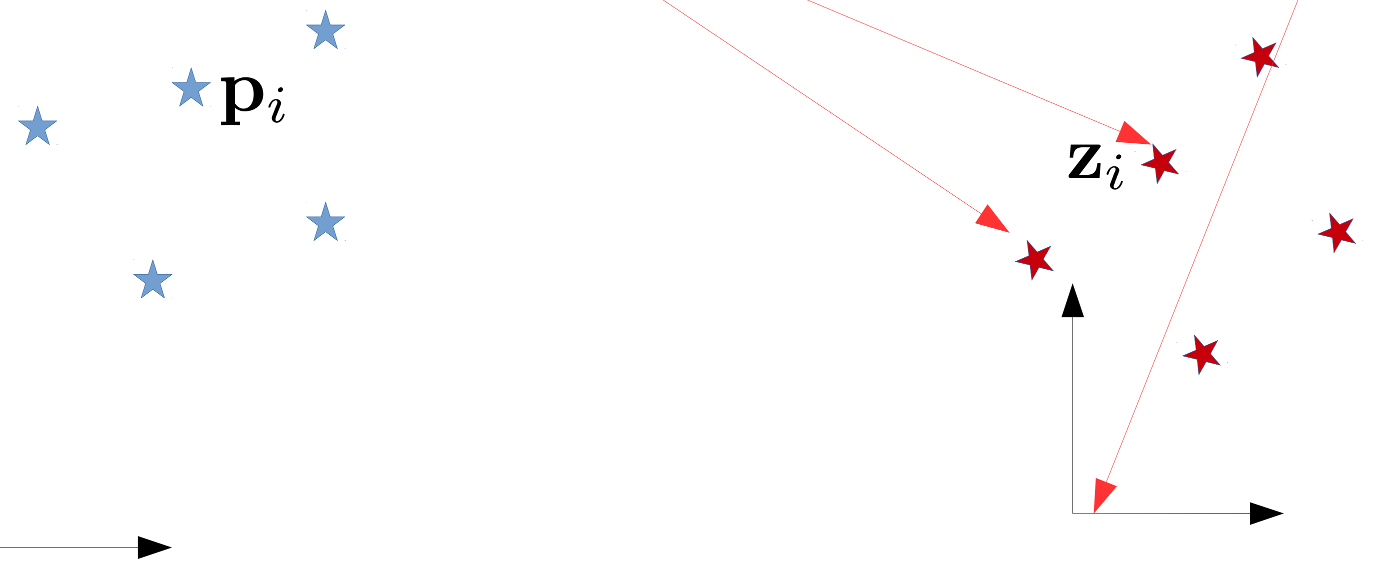
Example ICP Optimization in 2D

Given a set of points in the world frame



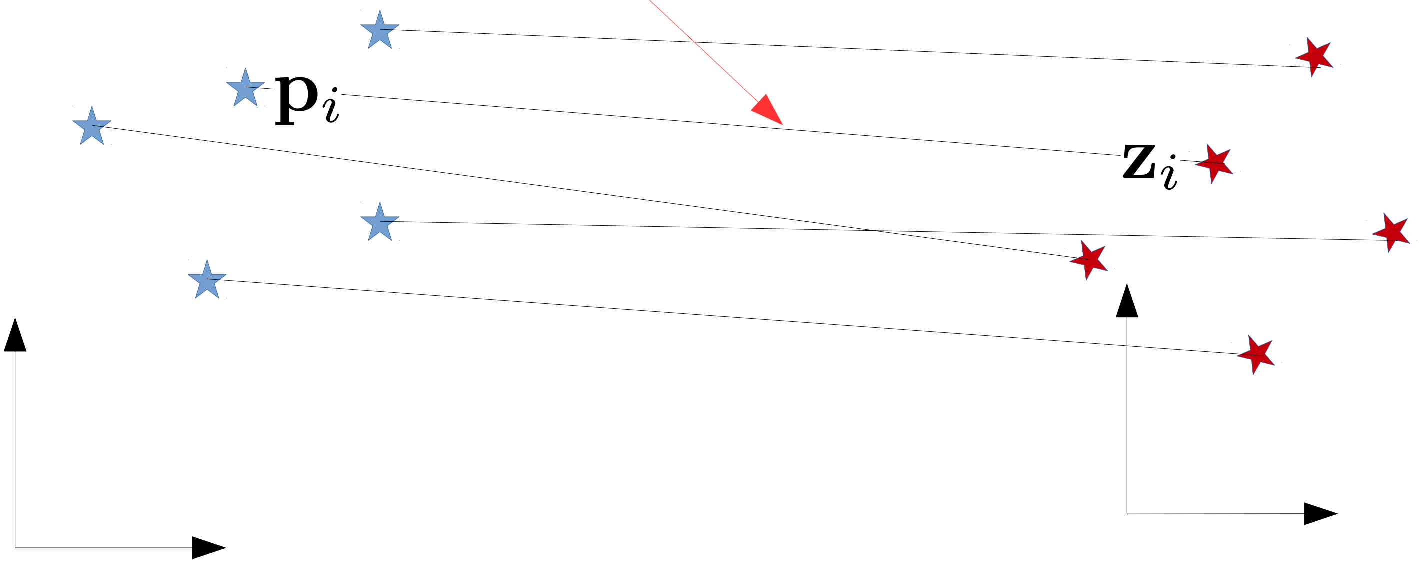
Example ICP Optimization in 2D

A set of 3D measurements in the robot frame



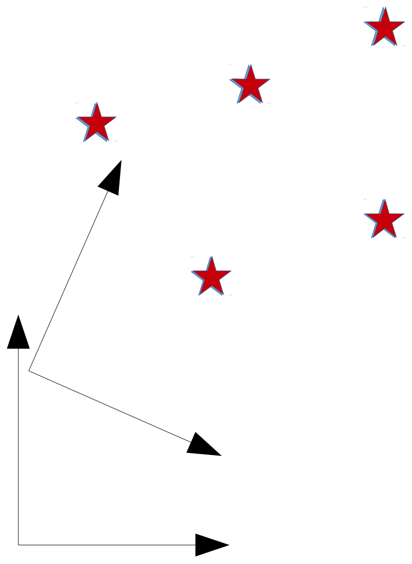
Example ICP Optimization in 2D

Roughly known correspondences



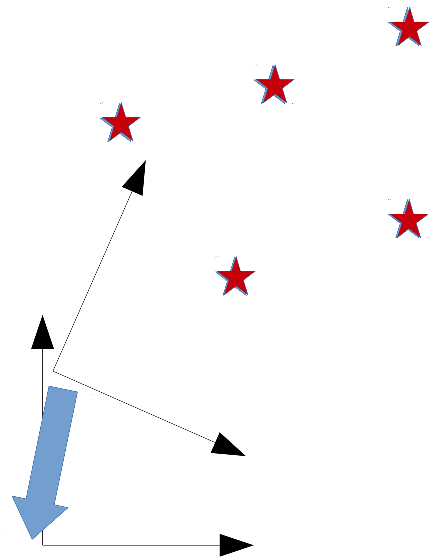
Example ICP Optimization in 2D

We want to find a transform that minimizes distance between corresponding points



Example ICP Optimization in 2D

Such a transform will be the pose of world w.r.t. robot



Note: we can also estimate robot w.r.t world, but it leads to longer calculations

ICP: State and Measurements

State

$$\mathbf{x} \in SE(2)$$
$$\mathbf{x} = (\underbrace{x\ y}_{\mathbf{t}}\ \theta)^T$$

Measurements

$$\mathbf{z} \in \Re^2$$
$$\mathbf{h}_i(\mathbf{x}) = \mathbf{R}(\theta)\mathbf{p}_i + \mathbf{t}$$

On Rotation Matrices

Recall a rotation matrix in 2D and the corresponding derivative w.r.t. the rotation angle

$$\mathbf{R}(\theta) = \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \qquad \mathbf{R}'(\theta) = \begin{pmatrix} -s & -c \\ c & -s \end{pmatrix}$$

```
function
R=rotation2D(theta)
  s=sin(theta);
  c=cos(theta);
  R=[c -s;
     s  c];
endfunction
```

```
function
Rp=rotation2Dgradient(theta)
  s=sin(theta);
  c=cos(theta);
  Rp=[-s -c;
       c -s];
endfunction
```

ICP: Jacobian

$$\mathbf{h}_i(\mathbf{x}) = \mathbf{R}(\theta)\mathbf{p}_i + \mathbf{t}$$

measurement function

$$\frac{\partial \mathbf{h}_i(\mathbf{x})}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial \mathbf{h}_i(\mathbf{x})}{\partial \mathbf{t}} & \frac{\partial \mathbf{h}_i(\mathbf{x})}{\partial \theta} \end{pmatrix}$$

measurement Jacobian

$$\frac{\partial \mathbf{h}_i(\mathbf{x})}{\partial \mathbf{t}} = \mathbf{I}$$

$$\frac{\partial \mathbf{h}_i(\mathbf{x})}{\partial \theta} = \mathbf{R}'(\theta)\mathbf{p}_i$$

ICP: Octave Code

```
function [e,J]=errorAndJacobian(x,p,z)
 t=x(1:2);
 theta=x(3);
 R=rotation2D(theta);
 z_hat=R*p+t;
 e=z_hat-z;
 J=zeros(2,3);
 J(1:2,1:2)=eye(2);
 J(1:2,3)=rotation2Dgradient(theta)*p;
endfunction;
```

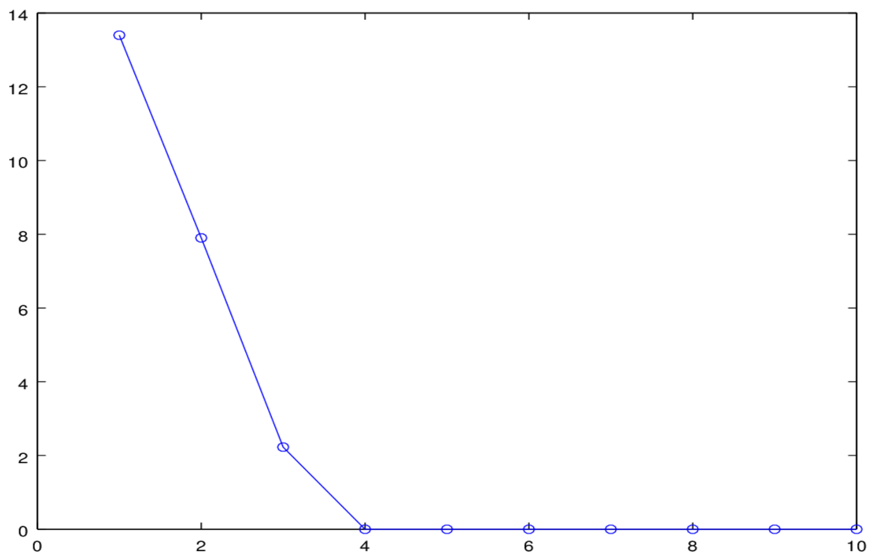
ICP: Octave Code

```
function [chi, x_new]=icp2d(x,P,Z)
  chi=0; %cumulative chi2
  H=zeros(3,3);  b=zeros(3,1); %accumulators for H and b
  for(i=1:size(P,2))
     p=P(:,i); z=Z(:,i); % fetch point and measurement
     [e,J]=errorAndJacobian(x,p,z); %compute e and J
     H+=J'*J;                  %assemble H and B
     b+=J'*e;
     chi+=e'*e;                %update cumulative error
  endfor
  dx=-H\b;                     %solve the linear system
  x_new=x+dx;                  %apply update

  %normalize theta between -pi and pi
  theta=x(3);
  s=sin(theta); c=cos(theta);
  x(3)=atan2(s,c);
endfunction
```

Testing, good initial guess

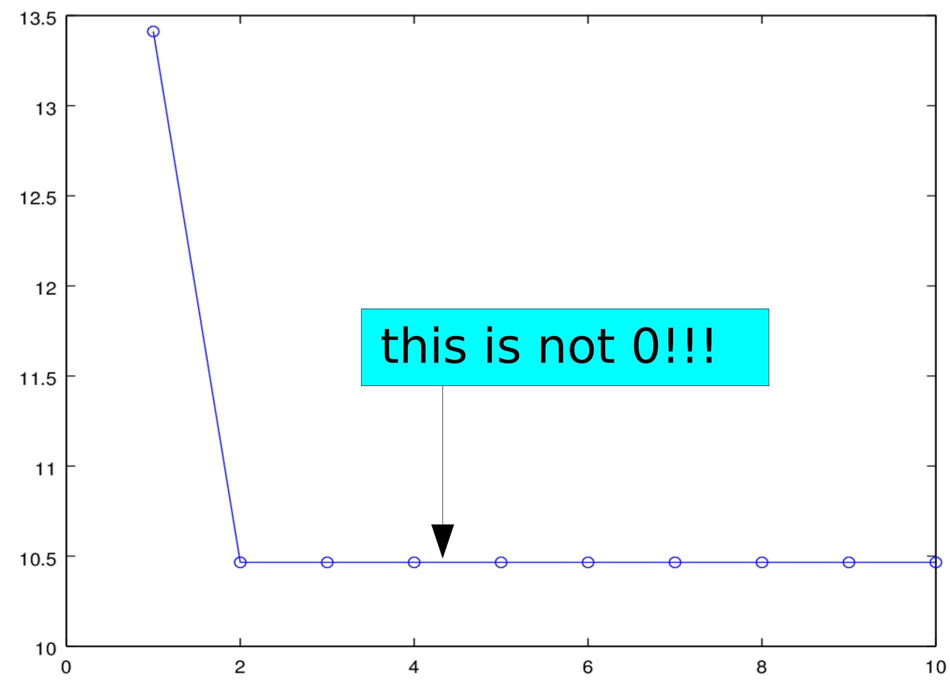
- Spawn a set of random points in 2D
- Define a location of the robot
- Compute synthetic measurements from that location
- Set the a point close to the true location as initial guess
- Run ICP and plot the evolution of the error



- When started from a good guess, the system converges nicely

Testing, bad initial guess

- Spawn a set of random points in 2D
- Define a location of the robot
- Compute synthetic measurements from that location
- Set the origin as initial guess
- Run ICP and plot the evolution of the error



- If the guess is poor, the system might take long to converge or **not** reach the minimum