Probabilistic Robotics Course

Multi-Point Registration

Giorgio Grisetti
grisetti@diag.uniroma1.it

Department of Computer, Control, and Management Engineering
Sapienza University of Rome

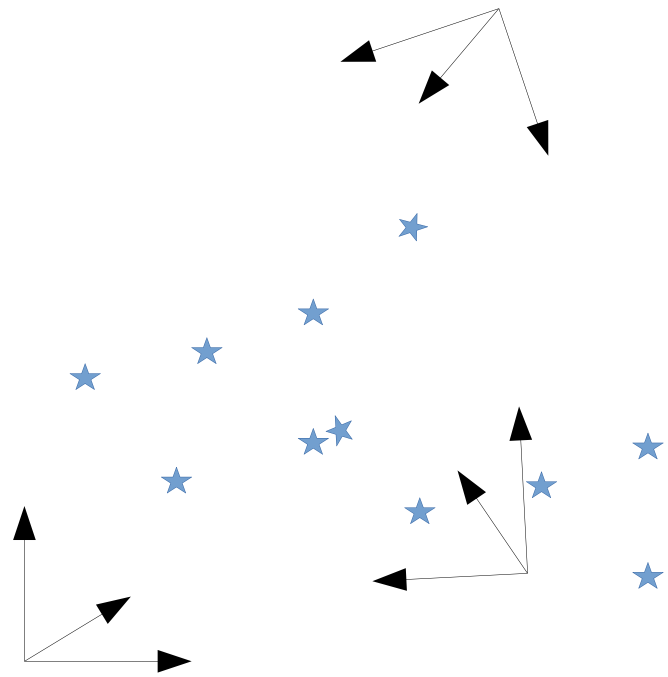
Problem Definition

The environment is populated with identifiable landmarks whose position is unknown

The landmarks are observed by a 3D sensor from multiple positions

We want to determine

- The landmark locations
- The robot positions



State

The state is a collection of transforms and landmark positions

$$\mathbf{X} \quad : \quad \mathbf{X} = \{\mathbf{X}_r^{[1]}, \ldots, \mathbf{X}_r^{[N]}, \mathbf{X}_l^{1}, \ldots, \mathbf{X}_l^{[M]}\}$$

robot $\mathbf{X}_r^{[n]} \in SE(3) \quad : \quad \mathbf{X}^{[n]} = \left(\mathbf{R}^{[n]} | \mathbf{t}^{[n]}\right)$

landmark $\mathbf{X}_l^{[m]} \in \Re^3 \quad : \quad \mathbf{X}_l^{[m]} = \left(x^{[m]}\ y^{[m]}\ z^{[m]}\right)^T$

The increments are represented by a large vector containing the minimal perturbation for each state variable

$$\mathbf{\Delta x} \in \Re^{6N+3M} \quad : \quad \mathbf{\Delta x} = \left(\mathbf{\Delta x}_r^{[1]T}, \ldots, \mathbf{\Delta x}_r^{[N]T}, \mathbf{\Delta x}_l^{[1]T}, \ldots, \mathbf{\Delta x}_l^{[M]T}\right)^T$$

robot $\mathbf{\Delta x}_r^{[n]T} \in \Re^6 \quad : \quad \mathbf{\Delta x}_r^{[n]T} = (\underbrace{\Delta x^{[n]}\ \Delta y^{[n]}\ \Delta z^{[n]}}_{\mathbf{\Delta t}^{[n]}}\ \underbrace{\Delta \alpha_x^{[n]}\ \Delta \alpha_y^{[n]}\ \Delta \alpha_z^{[n]}}_{\Delta \alpha^{[n]}})^T$

landmark $\mathbf{\Delta x}_l^{[m]T} \in \Re^3 \quad : \quad \mathbf{\Delta x}_l^{[l]T} = (\underbrace{\Delta x^{[m]}\ \Delta y^{[m]}\ \Delta z^{[m]}}_{\mathbf{\Delta t}^{[m]}})^T$

State (implementation)

In an Octave Implementation, we will represent the states as

- an array of **4 x 4** transformation matrices, for the robot poses
- a matrix of **3 x num_landmarks** elements, for the landmarks. Each column is a landmark

State (implementation)

With such an articulated representation of the state it might come in handy to have functions that translate

- a landmark or
- pose index

in the corresponding position in the perturbation vector

For a pose:

```
v_idx=1+(pose_index-1)*6;
```

For a landmark:

```
v_idx=1+(num_poses)*6 + (landmark_index-1)*3;
```

State (implementation)

```
function v_idx=poseMatrixIndex(pose_index, num_poses, num_landmarks)
  global pose_dim; # 6 in our case
  global landmark_dim; # 3 in our case
  if (pose_index>num_poses)
    v_idx=-1;
    return;
  endif;
  v_idx=1+(pose_index-1)*pose_dim;
endfunction;

function v_idx=landmarkMatrixIndex(landmark_index, num_poses, num_landmarks)
  global pose_dim;
  global landmark_dim;
  if (landmark_index>num_landmarks)
    v_idx=-1;
    return;
  endif;
  v_idx=1 + (num_poses)*pose_dim + (landmark_index-1) * landmark_dim;
endfunction;
```

Boxplus

The boxplus has to be adapted to apply the individual perturbations for each variable block

$$
\begin{aligned}
\mathbf{X}' &= \mathbf{X} \boxplus \mathbf{\Delta x} \\
{\mathbf{X}_r^{[n]}}' &= \mathbf{X}_r^{[n]} \boxplus \mathbf{\Delta x}_r^{[n]} \\
&= v2t(\mathbf{\Delta x}_r^{[n]})\mathbf{X}_r^{[n]} \\
{\mathbf{X}_l^{[m]}}' &= \mathbf{X}_l^{[m]} + \mathbf{\Delta x}_l^{[n]}
\end{aligned}
$$

Boxplus (implementation)

```
function [XR, XL]=boxPlus(XR, XL, num_poses, num_landmarks, dx)
  global pose_dim;
  global landmark_dim;
  for(pose_index=1:num_poses)
    pose_matrix_index=poseMatrixIndex(pose_index,
                                      num_poses,
                                      num_landmarks);
    dxr=dx(pose_matrix_index:pose_matrix_index+pose_dim-1);
    XR(:,:,pose_index)=v2t(dxr)*XR(:,:,pose_index);
  endfor;
  for(landmark_index=1:num_landmarks)
    landmark_matrix_index=landmarkMatrixIndex(landmark_index,
                                              num_poses,
                                              num_landmarks);
    dxl=dx(landmark_matrix_index:landmark_matrix_index+landmark_dim-1,:);
    XL(:,landmark_index)+=dxl;
  endfor;
endfunction;
```

Measurements and Predictions

A measurement of the landmark *m*, performed by robot pose *n* is as follows

$$\mathbf{z}^{[n,m]} \in \Re^3 \quad : \quad \mathbf{z}^{[n,m]} = \left(x^{[n,m]} \ y^{[n,m]} \ z^{[n,m]}\right)^T$$

The prediction and the error of this measurements are very similar to the ICP ones

$$\begin{aligned}
\mathbf{h}^{[n,m]}(\mathbf{X}) &= \mathbf{X}_r^{[n]}\mathbf{X}_l^{[m]} \\
\mathbf{e}^{[n,m]}(\mathbf{X}) &= \mathbf{X}_r^{[n]}\mathbf{X}_l^{[m]} - \mathbf{z}^{[n,m]} \\
\mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \mathbf{\Delta x}) &= v2t(\mathbf{\Delta x}_r^{[n]})\mathbf{X}_r^{[n]}(\mathbf{X}_l^{[m]} + \mathbf{\Delta x}_l^{[m]}) - \mathbf{z}^{[n,m]}
\end{aligned}$$

Jacobians

The prediction depends only on the robot pose m and the landmark n, so it will be mostly 0

Non 0 only for

- pose block n
- landmark block m

$$
\begin{aligned}
\frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \boldsymbol{\Delta}\mathbf{x})}{\partial \boldsymbol{\Delta}\mathbf{x}} &= \left(\cdots \mathbf{0}_{3\times 6} \cdots \frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \boldsymbol{\Delta}\mathbf{x})}{\partial \boldsymbol{\Delta}\mathbf{x}_r^n} \cdots \mathbf{0}_{3\times 6} \mathbf{0}_{3\times 3} \cdots \frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \boldsymbol{\Delta}\mathbf{x})}{\partial \boldsymbol{\Delta}\mathbf{x}_l^m} \cdots \mathbf{0}_{3\times 3} \cdots \right) \\
\hat{\mathbf{z}}^{[m,n]} &= \mathbf{X}_r^{[n]} \mathbf{X}_l^{[m]} \\
\left. \frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \boldsymbol{\Delta}\mathbf{x})}{\partial \boldsymbol{\Delta}\mathbf{x}_r^n} \right|_{\boldsymbol{\Delta}\mathbf{x}_r^{[n]} = \mathbf{0}} &= \underbrace{\left(\mathbf{I}_{3\times 3} | \lfloor -\hat{\mathbf{z}}^{[m,n]} \rfloor_\times \right)}_{\mathbf{J}_r^{n,m}} \\
\left. \frac{\partial \mathbf{e}^{[n,m]}(\mathbf{X} \boxplus \boldsymbol{\Delta}\mathbf{x})}{\partial \boldsymbol{\Delta}\mathbf{x}_l^m} \right|_{\boldsymbol{\Delta}\mathbf{x}_l^{[m]} = \mathbf{0}} &= \underbrace{\mathbf{R}^{[n]}}_{\mathbf{J}_l^{n,m}} \\
\mathbf{J}^{[n,m]} &= \left(\cdots \mathbf{0}_{3\times 6} \cdots \mathbf{J}_r^{[n,m]} \cdots \mathbf{0}_{3\times 6} \mathbf{0}_{3\times 3} \cdots \mathbf{J}_l^{[n,m]} \cdots \mathbf{0}_{3\times 3} \cdots \right)
\end{aligned}
$$

Jacobians (implementation)

```
function [e,Jr,Jl]=errorAndJacobian(Xr,Xl,z)
    R=Xr(1:3,1:3);
    t=Xr(1:3,4);
    z_hat=R*Xl+t; #prediction
    e=z_hat-z;
    Jr=zeros(3,6);
    Jr(1:3,1:3)=eye(3);
    Jr(1:3,4:6)=-skew(z_hat);
    Jl=R;
endfunction;
```

H Matrix and B vector

H and b for a measurement have only few non zero blocks

$$
\begin{aligned}
\mathbf{H}^{[n,m]} &= \mathbf{J}^{[n,m]T}\boldsymbol{\Omega}^{[n,m]}\mathbf{J}^{[n,m]} \\
&= \begin{pmatrix}
\cdots & \mathbf{J}_r^{[n,m]T}\boldsymbol{\Omega}^{[n,m]}\mathbf{J}_r^{[n,m]} & \cdots & \mathbf{J}_r^{[n,m]T}\boldsymbol{\Omega}^{[n,m]}\mathbf{J}_l^{[n,m]} & \cdots \\
 & \vdots & & \vdots & \\
\cdots & \mathbf{J}_l^{[n,m]T}\boldsymbol{\Omega}^{[n,m]}\mathbf{J}_r^{[n,m]} & \cdots & \mathbf{J}_l^{[n,m]T}\boldsymbol{\Omega}^{[n,m]}\mathbf{J}_l^{[n,m]} & \cdots
\end{pmatrix} \\
\mathbf{b}^{[n,m]} &= \mathbf{J}^{[n,m]T}\boldsymbol{\Omega}^{[n,m]}\mathbf{e}^{[n,m]} \\
&= \begin{pmatrix}
\vdots \\
\mathbf{J}_r^{[n,m]T}\boldsymbol{\Omega}^{[n,m]}\mathbf{e}^{[n,m]} \\
\vdots \\
\mathbf{J}_l^{[n,m]T}\boldsymbol{\Omega}^{[n,m]}\mathbf{e}^{[n,m]} \\
\vdots
\end{pmatrix}
\end{aligned}
$$

H and B Implementation

```
H=zeros(system_size, system_size);
b=zeros(system_size,1);
chi_stats(iteration)=0;
for (measurement_num=1:size(Z,2))
  pose_index=associations(1,measurement_num);
  landmark_index=associations(2,measurement_num);
  z=Z(:,measurement_num);
  Xr=XR(:,:,pose_index);
  Xl=XL(:,landmark_index);
  [e,Jr,Jl] = errorAndJacobian(Xr, Xl, z);
  Hrr=Jr'*Jr;
  Hrl=Jr'*Jl;
  Hll=Jl'*Jl;
  br=Jr'*e;
  bl=Jl'*e;

  pose_matrix_index=poseMatrixIndex(pose_index, num_poses, num_landmarks);
  landmark_matrix_index=landmarkMatrixIndex(landmark_index, num_poses, num_landmarks);

  H(pose_matrix_index:pose_matrix_index+pose_dim-1,
    pose_matrix_index:pose_matrix_index+pose_dim-1)+=Hrr;
  H(pose_matrix_index:pose_matrix_index+pose_dim-1,
    landmark_matrix_index:landmark_matrix_index+landmark_dim-1)+=Hrl;
  H(landmark_matrix_index:landmark_matrix_index+landmark_dim-1,
    landmark_matrix_index:landmark_matrix_index+landmark_dim-1)+=Hll;
  H(landmark_matrix_index:landmark_matrix_index+landmark_dim-1,
    pose_matrix_index:pose_matrix_index+pose_dim-1)+=Hrl';

  b(pose_matrix_index:pose_matrix_index+pose_dim-1)+=br;
  b(landmark_matrix_index:landmark_matrix_index+landmark_dim-1)+=bl;
endfor
```

Degrees of Freedom

A state configuration X, has the same error of all states obtained by applying a rigid transform to all variables.

The system will be under-determined

- 6 degrees of freedom (dof of a rigid transform)

To solve the system we need to either

- eliminate the redundant variables
- adding a prior

Eliminating Variables

We can impose that one robot pose stays fixed

To this extent we can

- reduce the linear system by suppressing the rows and the columns of the variables that stay fixed.
- solve the reduced system
- propagate the computed perturbation to all variables that have not been eliminated

Eliminating Variables (impl)

```
dx=zeros(system_size,1);

% we solve the linear system, blocking the first pose
% this corresponds to "remove" from H and b the locks
% of the 1st pose, while solving the system

dx(pose_dim+1:end)=
   -(H(pose_dim+1:end,pose_dim+1:end)\
    b(pose_dim+1:end,1));
[XR, XL]=boxPlus(XR,XL,num_poses, num_landmarks, dx);
```

Adding a Prior

Alternatively we can add a "flexible" constraint of the type $\mathbf{\Delta x}_r^{[1]} = \mathbf{0}$ to the least squares problem.

This is done by adding an error term

$$\begin{array}{ccc} \mathbf{e}_{\mathrm{prior}} & = & \mathbf{\Delta x}_r^{[1]} \\ \mathbf{J}_{\mathrm{prior}} & = & \mathbf{I} \\ \mathbf{H}_{\mathrm{prior}} & = & \mathbf{\Omega}_{\mathrm{prior}} \\ \mathbf{b}_{\mathrm{prior}} & = & \mathbf{\Omega}_{\mathbf{\Delta x}} \\ \mathbf{\Delta x} \rightarrow \mathbf{0} & \Rightarrow & \mathbf{b}_{\mathrm{prior}} \rightarrow \mathbf{0} \end{array}$$

Can be implemented by adding to the final H a 6x6 (large) information matrix in correspondence of the first variable

The larger the Omega, the more stiff the constraint will be

Adding a Prior (impl)

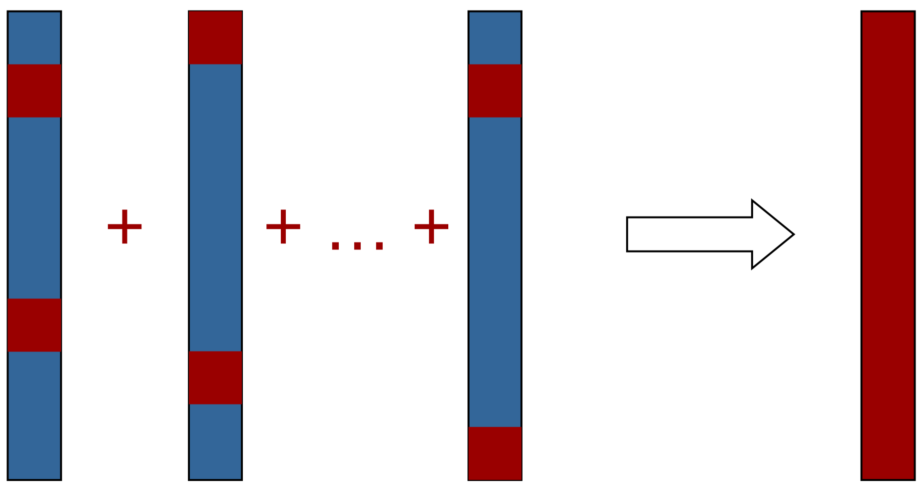
```
dx=zeros(system_size,1);
H(1:pose_dim,1:pose_dim)+=eye(6)*large_value;


dx=-(H\b);
[XR, XL]=boxPlus(XR,XL,num_poses,
                 num_landmarks, dx);
```

Structure

Structure of **H**: location of the non zero elements

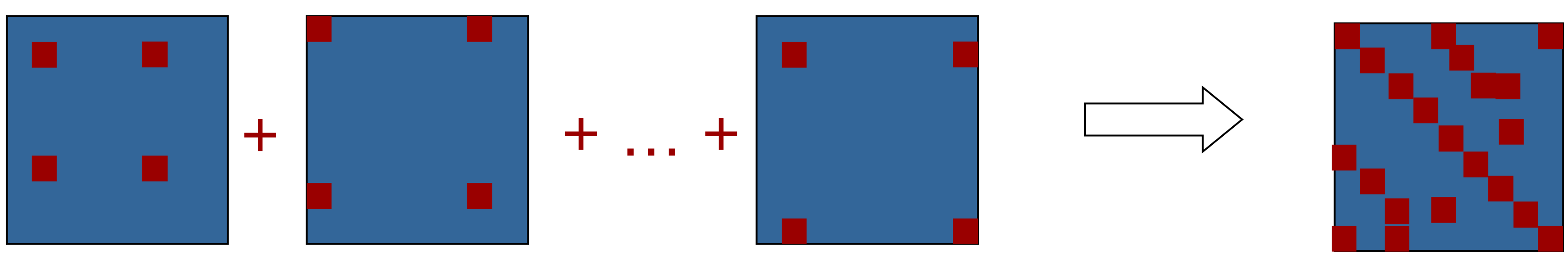
The structure of H depends only on the structure of the measurements

$$\mathbf{b} = \sum \mathbf{b}^{[n,m]}$$



full diagonal
block*[mn]* full if robot *m* observes landmark *n*

$$\mathbf{H} = \sum \mathbf{H}^{[n,m]}$$



Conclusions

We approached a complex multi-robot multi-landmark problem

- The measurement independence leads to a sparse structure of H
- The structure of H does not change during the iterations
- It can be efficiently solved by using sparse methods (we will see these later in this course)