Technion – Israel Institute of Technology

HW4

Vision Aided Navigation

086761

Alon Spinner	305184335	alonspinner@gmail.com
Sher Hazan	308026467	sherhazan@campus.technion.ac.il

December 18, 2021

Pose SLAM

Question 1: Assume you are given a prior on initial pose $p(x_0) = N(\hat{x}_0, \Sigma_0)$. Additionally, suppose the robot obtains noisy odometry (relative pose) measurements $z_k = x_{k+1} \ominus x_k + v_k$, with $v_k \sim N(0, \Sigma_v)$. Express the joint posterior $p(x_{0:k+1}|z_{0:k})$ as a product of the form $p(z_k|x_k, x_{k+1})$ and the prior $p(x_0)$ probabilistic terms. We shall call the probabilistic terms comprising the joint posterior factors.

$$p(x_{0:k+1}|z_{0:k}) \underbrace{=}_{Bayes} \frac{p(z_k|x_{0:k+1}, z_{0:k-1}) \cdot p(x_{0:k+1}|z_{0:k-1})}{p(z_k|z_{0:k-1})} \underbrace{=}_{CR} \frac{p(z_k|x_{0:k+1}, z_{0:k-1}) \cdot p(x_{k+1}|x_{0:k}, z_{0:k-1}) \cdot p(x_{0:k}|z_{0:k-1})}{p(z_k|z_{0:k-1})}$$

$Markov\ assumption$:

$$p(x_{k+1}|x_{0:k}, z_{0:k-1}) = p(x_{k+1}|x_k)$$

$$p(z_k|x_{0:k}, z_{0:k}) = p(z_k|x_{k+1}, x_k)$$

$Define\ Normalizer$:

$$p(z_k|z_{0:k-1}) = \frac{1}{\eta_k}$$

$$p(x_{0:k+1}|z_{0:k-1}) = \eta_k \cdot p(z_k|x_k, x_{k+1}) \cdot p(x_{k+1}|x_k) \cdot p(x_{0:k}|z_{0:k-1})$$

$Given\ no\ motion\ model\ we\ will\ assume\ p(x_{i+1}|x_i)\ is\ distributed\ uniformaly\ across\ the\ space$:

$$p(x_{i+1}|x_i) = const$$

$define\ \ \eta'_k = \eta_k \cdot const:$

$$p(x_{0:k+1}|z_{0:k-1}) = \eta'_k \cdot const \cdot p(x_{0:k}|z_{0:k-1}) \cdot p(z_k|x_k, x_{k+1})$$

$Similarly\ we\ will\ develop\ the\ expression\ for\ p(x_{0:k}|z_{0:k-1})$:

$$p(x_{0:k}|z_{0:k-1}) = \eta'_{k-1} \cdot p(x_{0:k-1}|z_{0:k-2}) \cdot p(z_{k-1}|x_{k-1}, x_k)$$

$$p(x_{0:k+1}|z_{0:k}) = \eta'_k \eta'_{k-1} \cdot p(x_{0:k-1}|z_{0:k-2}) \cdot p(z_k|x_k, x_{k+1}) \cdot p(z_{k-1}|x_{k-1}, x_k)$$

$And\ after\ k\ iterations\ we\ get\ the\ following\ expression$:

$$p(x_{0:k+1}|z_{0:k}) = p(x_0) \prod_{i=0}^{k} \eta'_i \cdot p(z_i|x_i, x_{i+1})$$

$define \prod_{i=0}^{k} \eta'_i = \eta:$

$$\boxed{p(x_{0:k+1}|z_{0:k}) = \eta p(x_0) \prod_{i=0}^{k} p(z_i|x_i, x_{i+1})}$$

Question 2: Formulate the smoothing optimization problem. Describe an iterative process to obtain the MAP estimate.

$$z_k = x_{k+1} \ominus x_k + v_k = h(x_{k+1}, x_k) + v_k$$

$$x^*_{0:k+1} = \arg\max\big(p(x_{0:k+1}|z_{0:k})\big) = \arg\max\left(\eta p(x_0)\prod_{i=0}^{k} p(x_{i+1}|x_i)\cdot p(z_i|x_i, x_{i+1}))\right)$$

$$= \arg\min\left(-\log\left(p(x_0)\prod_{i=0}^{k} p(x_{i+1}|x_i)\cdot p(z_i|x_i, x_{i+1})\right)\right)$$

$$= \arg\min\left(||x_0 - \hat{x}_0||^2_{\Sigma_0} + \sum_{i=0}^{k}||z_i - h(x_{i+1}, x_i)||^2_{\Sigma_v}\right)$$

Above is the formulation of the optimization problem.
We provide an optimal solution via LMS, pseudo-inverse below:

We define:

$$J(x_{0:k+1}) = ||x_0 - \hat{x}_0||^2_{\Sigma_0} + \sum_{i=0}^{k}||z_i - h(x_{i+1}, x_i)||^2_{\Sigma_v}$$

Linearizing $h(x_i, x_{i+1})$ around $(\bar{x}_i, \bar{x}_{i+1})$:

We Define $\Delta x_i = x_i - \bar{x}_i$

$$h(x_i, x_{i+1}) \approx h(\bar{x}_i, \bar{x}_{i+1}) + \frac{\partial h}{x_i}\bigg|_{\bar{x}_i, \bar{x}_{i+1}} \Delta x_i + \frac{\partial h}{x_{i+1}}\bigg|_{\bar{x}_i, \bar{x}_{i+1}} \Delta x_{i+1} = h(\bar{x}_i, \bar{x}_{i+1}) + H_i^1 \Delta x_i + H_i^2 \Delta x_{i+1}$$

$$J(x_{0:k+1}) \approx ||x_0 - \hat{x}_0||^2_{\Sigma_0} + \sum_{i=0}^{k}||z_i - h(\bar{x}_i, \bar{x}_{i+1}) + H_i^1 \Delta x_i + H_i^2 \Delta x_{i+1}||^2_{\Sigma_v} =$$

$$= \left|\left|\Sigma_0^{-\frac{1}{2}}(x - \hat{x}_0)\right|\right|^2 + \sum_{i=0}^{k}\left|\left|\Sigma_v^{-\frac{1}{2}}(z_i - h(\bar{x}_i, \bar{x}_{i+1}) + H_i^1 \Delta x_i + H_i^2 \Delta x_{i+1})\right|\right|^2 =$$

$$= ||A\theta - b||^2$$

$$\theta = \begin{bmatrix} x_0 - \bar{x}_0 \\ x_1 - \bar{x}_1 \\ \vdots \\ \vdots \\ x_{k+1} - \bar{x}_{k+1} \end{bmatrix} = \begin{bmatrix} x_0 - \hat{x}_0 \\ x_1 - \bar{x}_1 \\ \vdots \\ \vdots \\ x_{k+1} - \bar{x}_{k+1} \end{bmatrix}$$

$$A = \begin{bmatrix} \Sigma_0^{-\frac{1}{2}} - \Sigma_v^{-\frac{1}{2}} H_0^1 & -\Sigma_v^{-\frac{1}{2}} H_0^2 & 0 & \cdots & 0 \\ 0 & -\Sigma_v^{-\frac{1}{2}} H_1^1 & -\Sigma_v^{-\frac{1}{2}} H_1^2 & & \\ \vdots & & -\Sigma_v^{-\frac{1}{2}} H_2^1 & -\Sigma_v^{-\frac{1}{2}} H_2^2 & \\ \vdots & & & \ddots & \\ 0 & & & -\Sigma_v^{-\frac{1}{2}} H_k^1 & -\Sigma_v^{-\frac{1}{2}} H_k^2 \end{bmatrix}$$

$$b = \begin{bmatrix} \Sigma_v^{-\frac{1}{2}} \left(-z_0 + h(\hat{x}_0, \bar{x}_1) \right) \\ \Sigma_v^{-\frac{1}{2}} \left(-z_1 + h(\bar{x}_1, \bar{x}_2) \right) \\ \vdots \\ \vdots \\ \Sigma_v^{-\frac{1}{2}} \left(-z_k + h(\bar{x}_k, \bar{x}_{k+1}) \right) \end{bmatrix}$$

The vector θ^* which minimizes the squared residuals $\left|\left|A\theta - b\right|\right|^2$ is given by minimizing $\left|\left|A\theta - b\right|\right|$ with pseudo-inverse:

$$\theta^* = (A^T A)^{-1} A^T b$$

As such, the solution for our states will be given by:

$$x_{0:k+1} = \bar{x}_{0:k+1} + \theta^*$$

The combined covariance on all states:

$$\Sigma = I^{-1} = (A^T A)^{-1} = \left(\Sigma_0^{-1} + \sum_{i=0}^{k} \left({H_i^1}^T \Sigma_v^{-1} {H_i^1}^T + {H_i^2}^T \Sigma_v^{-1} {H_i^2}^T \right) \right)^{-1}$$

Hands-on Exercises

We will now use the GTSAM library to solve the smoothing problem as above. The file hw4_data.mat contains three variables:

dposes - holding noisy odometry measurements,

traj3- a rough initial estimate for robot trajectory

poses3_gt - the actual ground truth poses

```
addpath(fullfile(pwd,'gtsam_toolbox'));
disp('added toolbox to path')
```

```
added toolbox to path
```

1. Load and display the initial trajectory.

Cell array traj3 contains robot 6dof poses given as 4x4 transformation matrices (T C G in course notations). Follow the steps below to display the trajectory using GTSAM. Add your plot to your homework pdf / report.

```
data=importdata('hw4_data.mat');
```

(a) Convert transformations to gtsam.Pose3 objects.

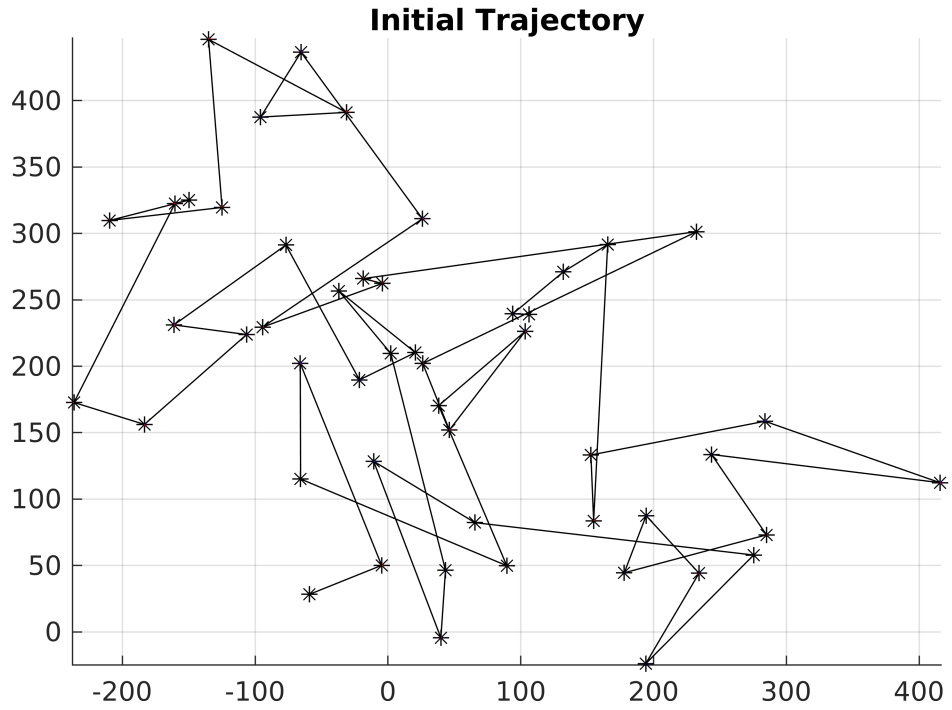
```
traj3Poses = cellfun(@(R) gtsam.Pose3(R), data.traj3);
```

(b) Store the above poses in a gtsam.Values object representing the initial estimate for robot trajectory.

```
trajectory = gtsam.Values;
for ii = 1:length(traj3Poses)
    trajectory.insert(gtsam.symbol('x',ii),traj3Poses(ii));
end
```

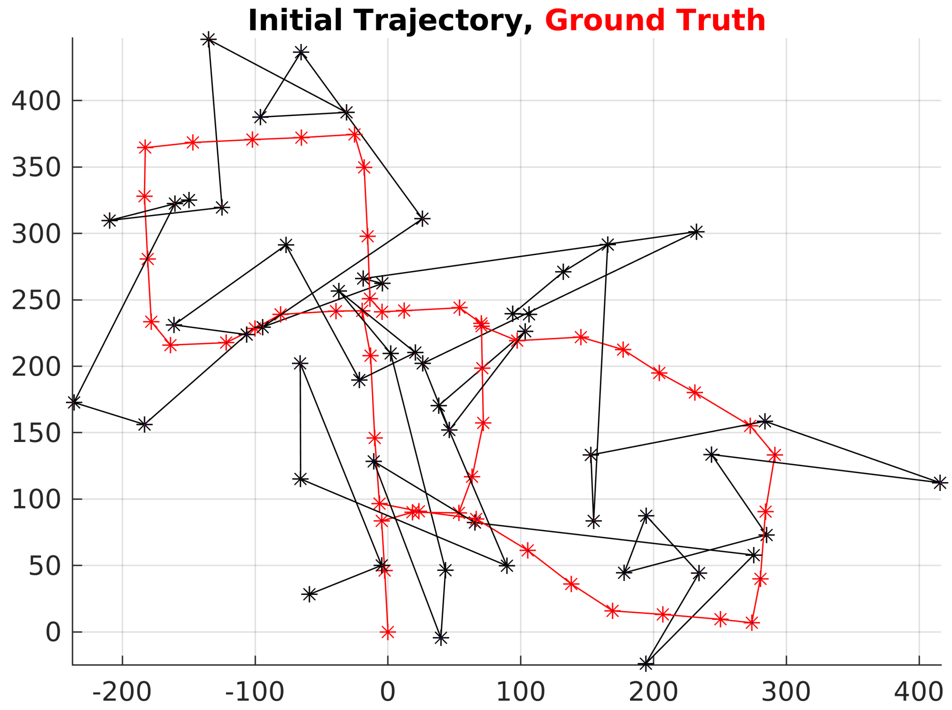
(c) Use gtsam.plot3DTrajectory to display the list of poses. Include the plot in your report.

```
figure()
gtsam.plot3DTrajectory(trajectory, '*-k' ,200)
view([0,-1,0]); axis equal; axis tight; grid on;
title('Initial Trajectory')
```



(d) Overlay in your plot the ground truth trajectory given in the variable poses3_gt (use a different color).

```
traj3Poses = cellfun(@(R) gtsam.Pose3(R), data.traj3);

gt3Poses = cellfun(@(R) gtsam.Pose3(R), data.poses3_gt);
gt = gtsam.Values;
for ii = 1:length(traj3Poses)
    gt.insert(gtsam.symbol('x',ii),gt3Poses(ii));
end

figure();
gtsam.plot3DTrajectory(trajectory, '*-k' ,200);
view([0,-1,0]); axis equal; axis tight; grid on; hold on;
gtsam.plot3DTrajectory(gt, '*-r' ,200);
title('\color{black}Initial Trajectory, \color{red}Ground Truth')
```



2. Construct factor graph.

(a) graph = gtsam.NonlinearFactorGraph creates a general factor graph.

```
graph = gtsam.NonlinearFactorGraph;
```

(b) Add gtsam.BetweenFactorPose3 factors for given "measured" relative poses.

```
Sr = 1e-3 * ones(3,1);
St = 0.1 * ones(3,1);
noiseModel = gtsam.noiseModel.Diagonal.Sigmas([Sr ; St ]);
zd3Poses = cellfun(@(R) gtsam.Pose3(R), data.dpose);

for ii = 1:length(zd3Poses)
    graph.add(gtsam.BetweenFactorPose3(gtsam.symbol('x',ii),...
        gtsam.symbol('x',ii+1),...
        zd3Poses(ii),...
        noiseModel));
end
```

(c) Assume robot is initially located at the origin, its axes aligned with the global reference frame. Use gtsam.PriorFactorPose3 to incorporate this information into the factor graph.

```
Origin = gtsam.Pose3(eye(4));
```

```
graph.add(gtsam.PriorFactorPose3(gtsam.symbol('x',1),Origin,noiseModel));
```

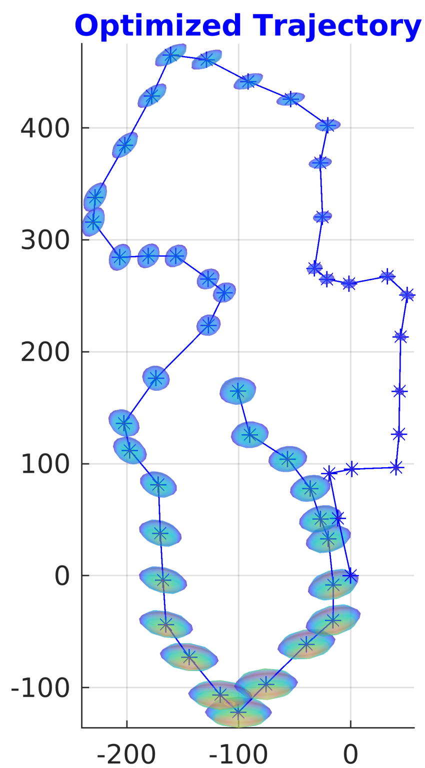
3. Calculate and display the MAP trajectory esimate.

(a) Create and run an optimizer.

```
optimizer = gtsam.LevenbergMarquardtOptimizer(graph,trajectory);
result =optimizer.optimizeSafely();
```

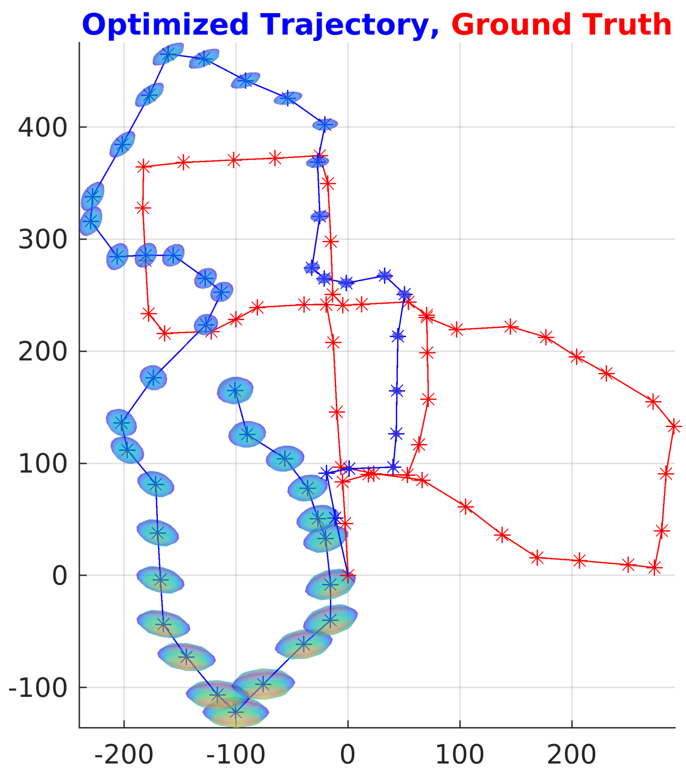
(b) Display updated trajectory estimate.

```
marginals = gtsam.Marginals(graph, result);

figure();
gtsam.plot3DTrajectory(result, '*-b' ,[],[],marginals)
view([0,-1,0]); axis equal; axis tight; grid on;
title('\color{blue}Optimized Trajectory');
```



(c) Overlay with the ground truth as in the 1st clause.

```
figure();
gtsam.plot3DTrajectory(result, '*-b' ,[],[],marginals)
gtsam.plot3DTrajectory(gt, '*-r' ,200);
view([0,-1,0]); axis equal; axis tight; grid on;
title('\color{blue}Optimized Trajectory, \color{red}Ground Truth')
```



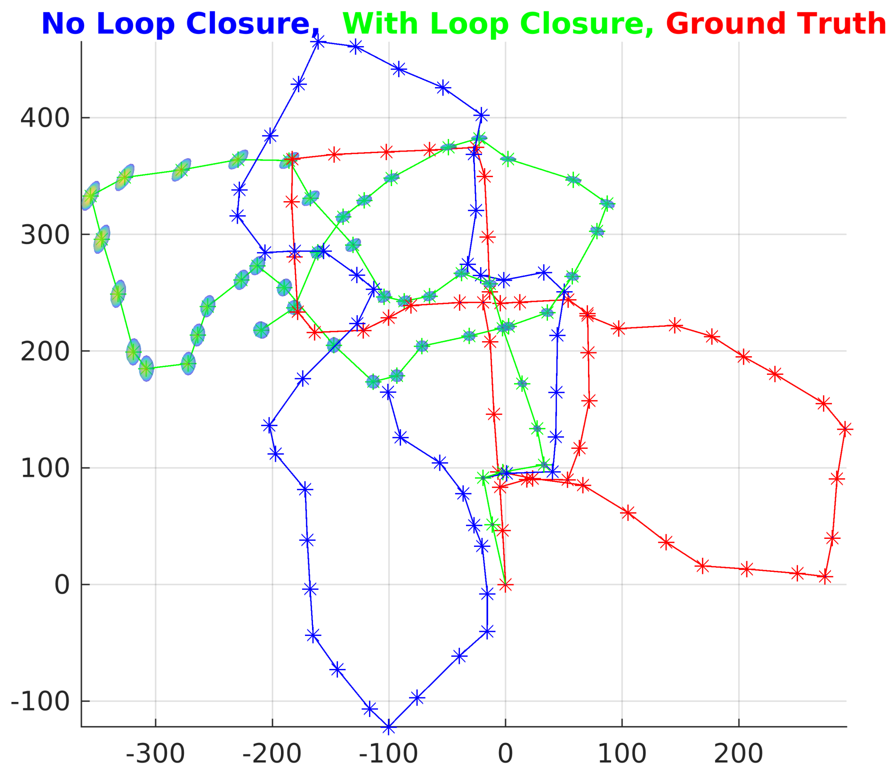
4. Loop closure

```
R2t1=[0.330571768  0.0494690228    -0.942483486;
         0.0138000518 0.998265226     0.0572371968;
          0.943679959  -0.0319273223   0.329315626];
t_2t1_in1=[-24.1616858;
               -0.0747429903;
               275.434963];
t1 = 3;
t2 = 42;
```

(a) Assuming the same noise model as before, show the estimated trajectory when incorporating this new information (as before, overlay it with ground truth).

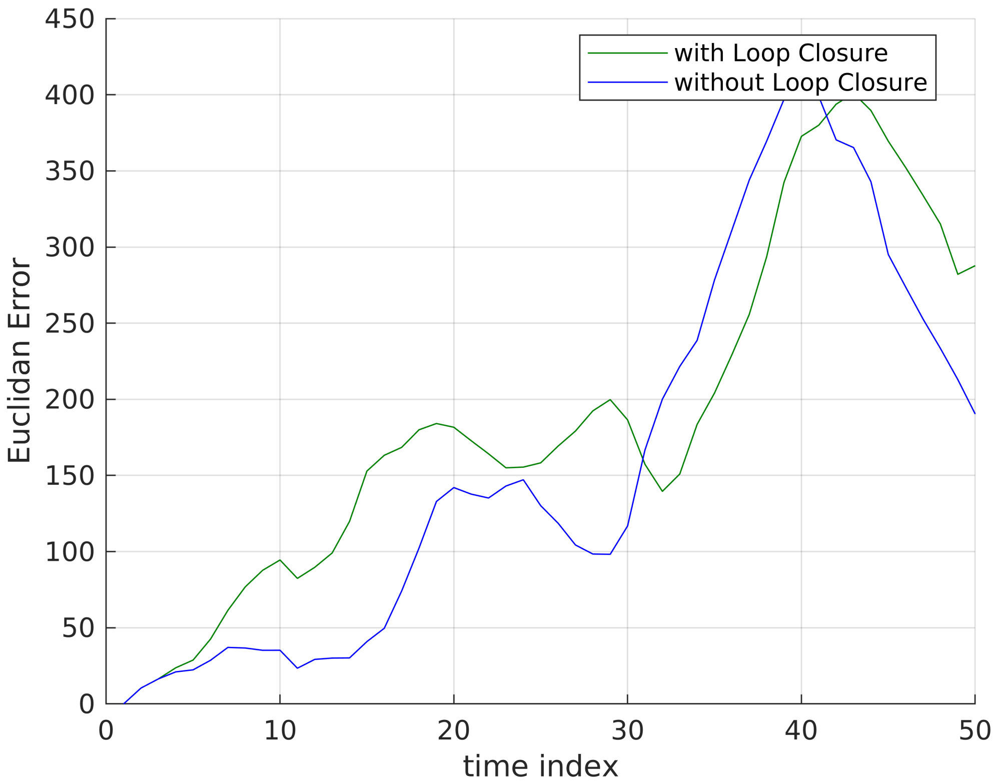
```
dPose = gtsam.Pose3([R2t1,t_2t1_in1; [0 0 0 1]]);
graph.add(gtsam.BetweenFactorPose3(gtsam.symbol('x',t1),...
        gtsam.symbol('x',t2),...
        dPose,...
        noiseModel));

optimizer = gtsam.LevenbergMarquardtOptimizer(graph,trajectory);
result_LC =optimizer.optimizeSafely();
marginals = gtsam.Marginals(graph, result_LC);

figure();
gtsam.plot3DTrajectory(result, '*-b');
gtsam.plot3DTrajectory(result_LC, '*-g' ,[],[],marginals);
```

```
gtsam.plot3DTrajectory(gt, '*-r' ,200);
view([0,-1,0]); axis equal; axis tight; grid on;
title('\color{blue}No Loop Closure, \color{green} With Loop Closure, \color{red}Ground
```



(b) Plot the localization error in metres (location difference only - Euclidean distance) over time, for the result without and with the loop closure.

```
poses_gt=gtsam.utilities.extractPose3(gt);
poses_nLC=gtsam.utilities.extractPose3(result);
poses_yLC=gtsam.utilities.extractPose3(result_LC);

%extract translation, motion is almost planar on the x-z plane.
t_poses_gt = poses_gt(:,10:12);
t_poses_nLC = poses_nLC(:,10:12);
t_poses_yLC = poses_yLC(:,10:12);

err_nLC = vecnorm(t_poses_nLC - t_poses_gt,2,2);
err_yLC = vecnorm(t_poses_yLC - t_poses_gt,2,2);

fig = figure('color',[1,1,1]);
ax = axes(fig);
grid(ax,'on'); hold(ax,'on'); xlabel(ax,'time index'); ylabel(ax,'Euclidan Error');
plot(ax,err_yLC,'color',[0 0.5,0]);
plot(ax,err_nLC,'b');
legend('with Loop Closure','without Loop Closure','location','best');
```



Bonus: Compute a good loop closure from poses in t9 and t50 and repeat

reminder:

$$R_1^2 = R_G^2 R_1^G = \left(R_2^G\right)^T R_1^G$$

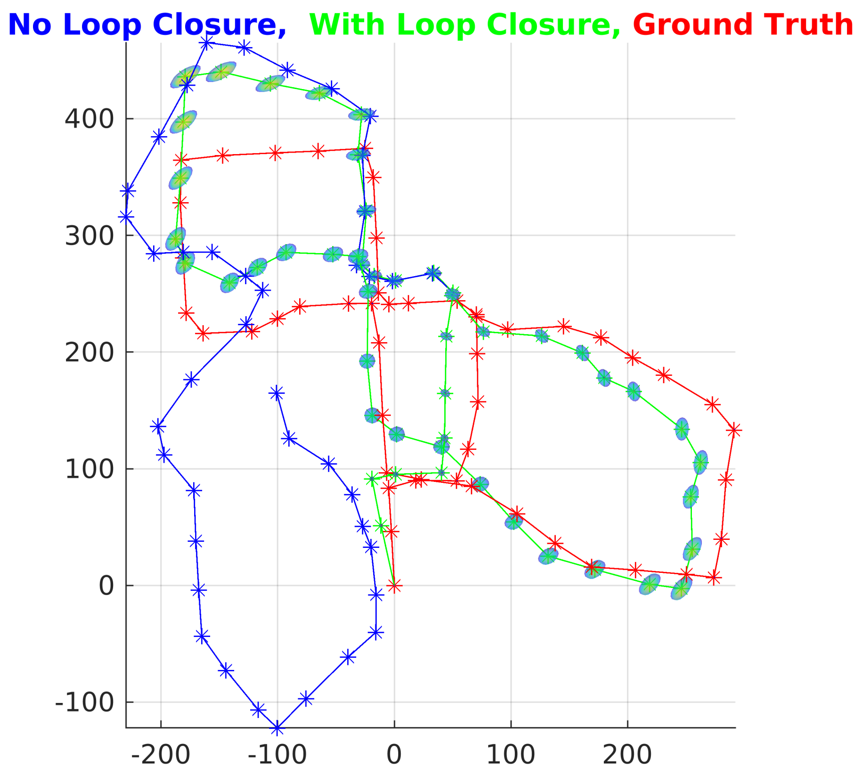
$$t_{2\to 1}^2 = R_G^2\left(t_{2\to G}^G - t_{1\to G}^G\right) = \left(R_2^G\right)^T\left(t_{2\to G}^G - t_{1\to G}^G\right)$$

Reconstruct Graph

```
graph = gtsam.NonlinearFactorGraph;
Sr = 1e-3 * ones(3,1);
St = 0.1 * ones(3,1);
noiseModel = gtsam.noiseModel.Diagonal.Sigmas([Sr ; St ]);
zd3Poses = cellfun(@(R) gtsam.Pose3(R), data.dpose);
for ii = 1:length(zd3Poses)
    graph.add(gtsam.BetweenFactorPose3(gtsam.symbol('x',ii),...
        gtsam.symbol('x',ii+1),...
        zd3Poses(ii),...
        noiseModel));
end
Origin = gtsam.Pose3(eye(4));
graph.add(gtsam.PriorFactorPose3(gtsam.symbol('x',1),Origin,noiseModel));
```

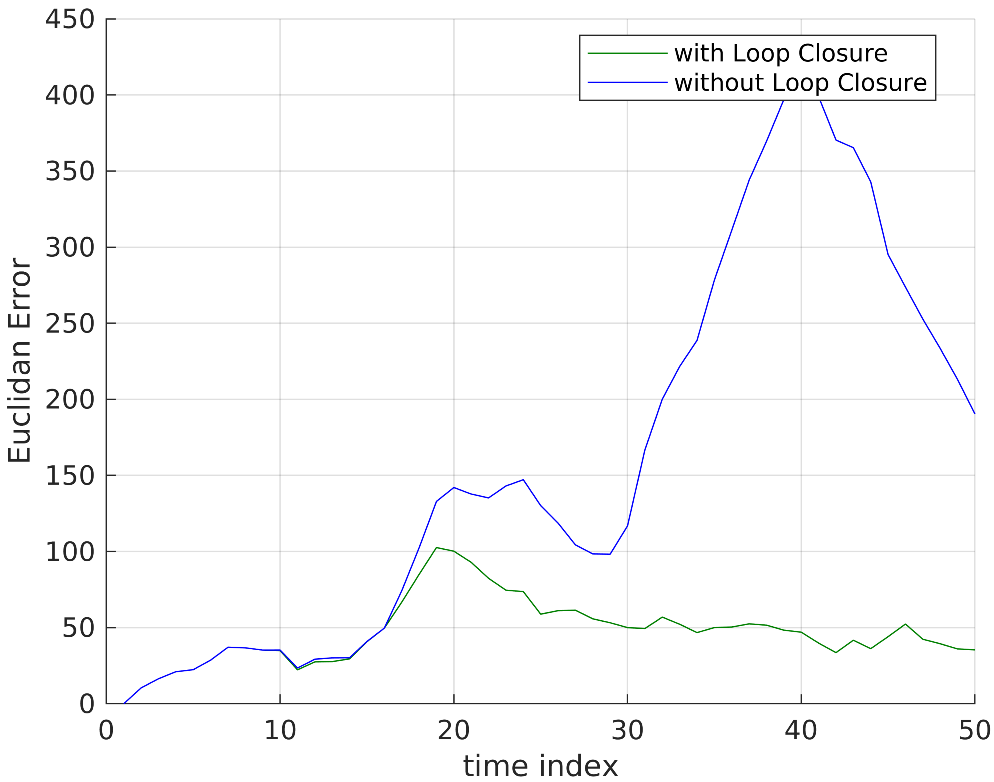
Connect new loop closure

```
t1 = 50;
t2 = 9;

pose1 = [reshape(poses_gt(t1,:),[3,4]);[0 0 0 1]];
pose2 = [reshape(poses_gt(t2,:),[3,4]);[0 0 0 1]];
R1tG = pose1(1:3,1:3);
R2tG = pose2(1:3,1:3);
t_1tG_inG = pose1(1:3,4);
t_2tG_inG = pose2(1:3,4);

R1t2 = R2tG'*R1tG;
t_2t1_in2 = R2tG'*(t_2tG_inG-t_1tG_inG);

dPose =  gtsam.Pose3([R1t2,t_2t1_in2;[0 0 0 1]]);

graph.add(gtsam.BetweenFactorPose3(gtsam.symbol('x',t1),...
        gtsam.symbol('x',t2),...
        dPose,...
        noiseModel));

optimizer = gtsam.LevenbergMarquardtOptimizer(graph,trajectory);
result_LC =optimizer.optimizeSafely();
marginals = gtsam.Marginals(graph, result_LC);

figure();
gtsam.plot3DTrajectory(result, '*-b');
gtsam.plot3DTrajectory(result_LC, '*-g' ,[],[],marginals);
gtsam.plot3DTrajectory(gt, '*-r' ,200);
view([0,-1,0]); axis equal; axis tight; grid on;
title('\color{blue}No Loop Closure, \color{green} With Loop Closure, \color{red}Ground
```



Error Graphs for our new loop closure

```
poses_gt=gtsam.utilities.extractPose3(gt);
poses_nLC=gtsam.utilities.extractPose3(result);
poses_yLC=gtsam.utilities.extractPose3(result_LC);

%extract translation, motion is almost planar on the x-z plane.
t_poses_gt = poses_gt(:,10:12);
t_poses_nLC = poses_nLC(:,10:12);
t_poses_yLC = poses_yLC(:,10:12);

err_nLC = vecnorm(t_poses_nLC - t_poses_gt,2,2);
err_yLC = vecnorm(t_poses_yLC - t_poses_gt,2,2);

fig = figure('color',[1,1,1]);
ax = axes(fig);
grid(ax,'on'); hold(ax,'on'); xlabel(ax,'time index'); ylabel(ax,'Euclidan Error');
plot(ax,err_yLC,'color',[0 0.5,0]);
plot(ax,err_nLC,'b');
legend('with Loop Closure','without Loop Closure','location','best');
```


with Loop Closure
without Loop Closure
Euclidan Error
time index
0
50
100
150
200
250
300
350
400
450
0
10
20
30
40
50