Vision-Aided Navigation (086761)
Homework #4

Submission in pairs. Please email your submission (a single file in the format ID1-ID2.zip) to moranbar@campus.technion.ac.il by 22 December 2021, 23:59. Electronic submission is preferred.

In this homework we will use the GTSAM library to estimate the trajectory of a moving robot *(a.k.a. pose SLAM).* Data for the exercise can be found in `hw4_data.mat`, under 'Resources', on Piazza.

Instructions:

- *Download "Precompiled MATLAB toolbox", to avoid building the library. GTSAM also has (and is based on) a C++ interface / implementation, as well as python bindings. Working in a Unix environment is advised if not using the MATLAB toolbox.*
- *The MATLAB toolbox contains code examples which typically run out-of-the-box, and can be quite useful. Background and additional explanations can be found in the hands-on introduction.*

Pose SLAM

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 as a product of odometry (*i.e. terms of the form* $p(z_k \mid x_k, x_{k+1})$) and prior ($p(x_0)$) probabilistic terms *(No need to specify the exact PDF). We shall call the probabilistic terms comprising the joint posterior* factors.
2. Formulate the smoothing (pose SLAM) optimization problem (note that the objective is *a trajectory*). Describe an iterative process to obtain the MAP estimate.

Hands-on Exercises - Please print and submit your code

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, and `poses3_gt` - the actual ground truth poses we will be comparing against in the following sections.

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.
   (a) `Convert transformations to gtsam.Pose3` objects.
   This class allows to represent 3D poses and transformations, with useful methods like getters for the rotation and translation elements, and some others that will be covered below. gtsam.Pose3 object can also be constructed from a pair of gtsam.Rot3 and a gtsam.Point3 denoting rotation and translation respectively.
   (b) Store the above poses in a `gtsam.Values` object representing the initial estimate for robot trajectory
   Each variable to be optimized (node) is identified with a unique key. Key-value pairs can be stored in gtsam.Values objects as follows:

```
trajectory = gtsam.Values
trajectory.insert(key, value-to-insert);
```

Here, key is associated with the inserted value, and can be an explicit number (e.g. time index) or generated by `gtsam.symbol` function, which can be used to create keys differentiating between different kinds of variables, e.g. `gtsam.symbol('x', 4)` can be used to identify/refer to a variable x_4 as opposed to `gtsam.symbol('l', 4)` for l_4. A particular value can be retrieved using trajectory.at(key).

(c) Use gtsam.plot3DTrajectory to display the list of poses. Include the plot in your report. *For your convenience, the interface is:*

```
gtsam.plot3DTrajectory(values-list , linespec , [] , scale)
```

`values-list` is a `gtsam.Values` object. `scale` *determines the size of axes denoting the pose.* ***`linespec`*** *is in the regular MATLAB format. You can use*

```
view([0, -1, 0]); axis equal tight;
```

to obtain a top view of the trajectory.

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

Note: For save/load gtsam objects to work correctly generally need to serialize (convert to char array using saveobj/loadobj methods of the relevant object), and store / load as char array (rather than directly the object itself).

2. Construct factor graph. Variable `dpose` in the provided mat file holds (noisy) relative poses at consecutive times, starting with time 0. These correspond to measurements z_k from the first clause, corrupted with measurement noise Σ_v.

(a) `graph = gtsam.NonlinearFactorGraph` creates a general factor graph. Factors can be added using `graph.add(factor)`.

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

```
gtsam.BetweenFactorPose3(key1 , key2 , relative-pose , noise-model)
```

Use gtsam.noiseModel.Diagonal.Sigmas($[S_r, S_t]$), where S_r, S_t are 3-component row vectors specifying the standard deviation along each dimension in the rotation (S_r) and translation (S_t). Use standard deviation of $1e-3$ radians for rotation and 0.1 metres for translation (along each axis), writing explicitly:

$$\Sigma_v = \begin{pmatrix} diag(S_r^2) & 0 \\ 0 & diag(S_t^2) \end{pmatrix}$$

(here $(\cdot)^2$ is applied element-wise).

(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. *Interface:*

```
gtsam.PriorFactorPose3(key , pose , noise-model)
```

Use the same noise model as before (i.e. $\Sigma_0 = \Sigma_v$).

3. Calculate and display the MAP trajectory esimate.

(a) Create and run an optimizer. *As follows:*

```
optimizer = gtsam.LevenbergMarquardtOptimizer(graph , initial-estimate);
result = optimizer.optimizeSafely();
```

*Use initial trajecory from 1st clause to provide initial estimates for all optimization targets. Obtained **`result`** holds the MAP estimate.*

(b) Display updated trajectory estimate. Plot marginal covariances using:

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

Then passing `marginals` as additional (after `scale`) parameter to `gtsam.plot3DTrajectory (alternatively, can use gtsam.plotPose3)`. Include the plot in your report.

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

4. Loop closure. Suppose the robot observed the same scene at times $t_1 = 3$ and $t_2 = 42$, and as a result calculated the following pose difference:

$$R_2^1 = \begin{pmatrix} 0.330571768 & 0.0494690228 & -0.942483486 \\ 0.0138000518 & 0.998265226 & 0.0572371968 \\ 0.943679959 & -0.0319273223 & 0.329315626 \end{pmatrix} \quad t_{2\to 1}^1 = \begin{pmatrix} -24.1616858 \\ -0.0747429903 \\ 275.434963 \end{pmatrix}$$

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

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