Abstract
This document provides some details about the MATLAB code of 2D Iterated Sparse Local Submap Joining Filter (I-SLSJF).
If you provide a sequence of local maps (point feature maps), I-SLSJF will give you a global map. The code is written by Shoudong Huang and Zhan Wang (zhwang@eng.uts.edu.au), the code together with some simulation and experimental data sets are available on the website: http://services.eng.uts.edu.au/~sdhuang/research.htm. Please contact Shoudong Huang if you have any questions/comments about the code.

I. IS THIS MATLAB CODE USEFUL FOR YOU?
If you are a researcher working on SLAM related problems, you may find this code useful for you (or your students). Below is a quick check on this.
After downloading the code, just run MainLoop.m and you will get the map joining result using the 200 local maps built from the Victoria Park data set as shown in Figure 1(b).
Now open the file DoSetupParam.m and change the parameter Params.Simulation = 5 into Params.Simulation = 4, then run MainLoop.m again. Now you will get the map joining result using the 200 local map built from the DLR-Spatial-Cognition data set as shown in Figure 2(b).
If you think the results looks good to you, then you may want to understand more about the algorithm and the code.

II. BACKGROUND
Recently we have developed the I-SLSJF algorithm [1] – an efficient and consistent algorithm for point feature based SLAM. I-SLSJF is a local submaps joining algorithm. The local maps can be built by any reliable SLAM algorithm. The format of the local map is the same as that obtained by EKF SLAM — a state vector and the corresponding covariance matrix.
The details of the I-SLSJF can be found in [1] and [2]. The advantages of I-SLSJF include:
• Efficiency — since the information matrix is exactly sparse, the algorithm is very efficient.
(a) Structure of the environment and the robot trajectory
(b) The map obtained by I-SLSJF

Fig. 2. The map joining results using DLR-Spatial-Cognition data set.

- Consistency — since a least squares smoothing is added on top of the EIF map joining, the map estimate is more consistent than filter based SLAM algorithms.
- Data association — since the local maps are joined in sequence, the data association between local map and the global map can be performed efficiently.

The code for Sparse Local Submap Joining Filter (SLSJF) [2] is also included as a special case of I-SLSJF. As compared with I-SLSJF, SLSJF is more efficient (fast) but less accurate (consistent).

### III. Input and Output of I-SLSJF

**A. Input: A sequence of local maps**

Suppose there are \( p \) local maps. The local map \( k \) (\( 1 \leq k \leq p \)) is expressed by

\[
(\hat{X}^L_k, P^L_k)
\]

where \( \hat{X}^L_k \) (here the superscript ‘L’ stands for the local map) is an estimate of the state vector

\[
X^L_k = (X^L_k r, X^L_k 1, \ldots, X^L_k n_k)
\]

and \( P^L_k \) is the associated covariance matrix. The state vector \( X^L_k \) contains the final robot location \( X^L_k r \) and the \( n_k \) local features \( X^L_k 1, \ldots, X^L_k n_k \) (assuming the robot location is (0,0,0) when the local map is started).

**The ONLY Requirement:** the final robot location in local map 1 is the same as the robot starting point in local map 2, the final robot location in local map 2 is the same as the robot starting point in local map 3, and so on.

**B. Output: A global map**

The coordinate system of the global map is the same as that of local map 1. The state vector of the global map contain all the feature positions and the final robot poses in each local map.

\[
X^G = (X^G 1, X^G n_1, X^G k r, X^G 21, \ldots, X^G 2m_2, \ldots, X^G p, X^G pm_p, X^G pr)
\]

where \( X^G 1 = (X^G 1 r, y^G 1 r, \phi^G 1 r) \) is the final robot location in submap \( k \) (\( 1 \leq k \leq p \)). Note that the total number of features is smaller than the sum of the number of features in each local maps because of the overlap of the local maps. For example, \( X^G 21, \ldots, X^G 2m_2 \) are the features in local map 2 but not in local map 1 (\( m_2 \leq n_2 \)).

The global map is given in the form of (i) a state estimate \( Est.StGlobal \), (ii) an information vector \( Est.InfoVectorGlobal \), and (iii) a sparse information matrix \( Est.InfoMatrixGlobal \).
IV. Usage of the Code

A. Data sets

There are five data sets that are provided together with the code:

- 535 steps simulation data;
- 8240 steps simulation data;
- 35188 steps simulation data;
- DLR-Spatial-Cognition data set;
- Victoria Park data set.

Each data set is given in the format of the following:

- A sequence of local maps – the local map \( i \) is given by \( \text{localmap}_i.mat \)
- For simulation data sets, \( \text{parameters.txt} \) explains the parameters used, the robot motion model and the control and observation data. Ground truths of features and robot poses are also available for simulation data sets – \( \text{store_beaconsTrue.mat} \) and \( \text{store_robotTrue.mat} \).
- For DLR-Spatial-Cognition data set, the original data is in the file \( \text{map.circles} \), more information about the data can be found at http://www.sfbtr8.spatial-cognition.de/insidedataassociation/data.html
  However, we have found some small errors in the data as compared with the video provided. The corrected data is given by \( \text{corrected_data.circles.mat} \).
- For Victoria Park data set, the control data and the preprocessed sensor data are provided as \( \text{ControlData.mat} \) and \( \text{data_sensor_clean_clean.dat} \), the parameters and the vehicle motion model and the observation model are given as MATLAB files \( \text{Parameters.m} \), \( \text{DoPredictionOptimized.m} \) and \( \text{DoUpdate.m} \). More information about the raw sensor data can be found at http://www-personal.acfr.usyd.edu.au/nebot/victoria_park.htm
  For Victoria Park data set, we also provide another data which has the same format as that of \( \text{map.circles} \) in the DLR-Spatial-Cognition data set. The data is given by \( \text{VicPark_clean_data.mat} \). With this data, the user does not need to worry about the details of the vehicle model and the raw sensor noise etc. since the covariance matrices of odometry and observations are all given.

B. Use of the code

1) Set up parameters: All the parameters needed to be set up are in \( \text{DoSetupParam.m} \)

The parameters include:
- \( \text{Params.Simulation} \) — select which data set to use
- \( \text{Params.NumOfSubmap} \) — how many local maps to fuse
- \( \text{Params.IndexSubmapStart} \) — the index of the first local map to fuse
- \( \text{Params.SmoothAfterUpdate} \) — select whether to use SLSJF or I-SLSJF
- \( \text{Params.AssumeDataAssoc} \) — select whether to perform data association or assume data association
- \( \text{Params.ReorderSubmaps} \) — select whether to reorder the state vector or not
- \( \text{Params.ReorderAMD} \) — select whether to perform AMD reordering or AMD plus distance reordering [2]

There are some other parameters for data association and reordering of the state vector.

2) Format of the result: Once the parameters are selected, just run \( \text{MainLoop.m} \)

The output of the algorithm include the global map, the consistency check result (for simulation data with ground truth) and the wrong data association results (if data association is performed).

The computation time for fusing each local map, the sparse information matrix, and the final global map are shown in figures.

C. Try your own local maps

If you want to try to fuse your own local maps using I-SLSJF, simply build the local maps (note the requirement listed in Section III-A) and save them in the correct format. In each \( \text{localmap}_j \), \( \text{localmap}_j \) is the state vector (same format as EKF SLAM state) and \( \text{localmap}_j, \) is the corresponding covariance matrix. The first column of \( \text{localmap}_j \) is used to keep the ground truth feature index of the feature (if it is available), the index of robot pose is 0.

If you do have the ground truth of your robot poses and feature positions, save them as \( \text{store_beaconsTrue.mat} \) and \( \text{store_robotTrue.mat} \) and put then and all the local map in the folder \( \text{Test_your_local_maps_with_ground_truth} \). Choose \( \text{Params.Simulation} = 0 \), then run \( \text{MainLoop.m} \)

If you do not have the ground truth but do have the true feature index in the local maps, then put all the local map data in the folder \( \text{Test_your_local_maps} \). Choose \( \text{Params.Simulation} = 6 \), then run \( \text{MainLoop.m} \).
• If you do not have the true index of features in the local maps, you can given any index (e.g. the local index) and put the local map data in the folder Test_your_local_maps_without_true_index. Choose Params.Simulation = −1, then run MainLoop.m. In this case, it is impossible to check the data association result, so you have to choose Params.AssumeDataAssoc = 0.

D. How to build your local maps

If you know how to use EKF SLAM to build small maps, then just use it as a good starting point. If you want to improve the quality of the local maps (such that the quality of the global map can also be improved), then we suggest you to use Maximal Likelihood (ML) approach. If you do not know how to build map using ML, there is a secret here – you can use I-SLSJF to do it.

In fact, when the number of local maps is equal to the number of time steps, that is, each local map is built using observations made from one robot pose together with one step odometry data, then the map fusion result from I-SLSJF is equivalent to the ML result (although I-SLSJF may be not as efficient in this case).

The data sets DLR_3297_local_maps and VicPark_6898_local_maps are included to demonstrate this. You can select Params.Simulation = 7 and Params.Simulation = 8, respectively to get the ML results for VicPark data and DLR data. However, it will take a long time if you want to get the ML result for the whole data.

If you do not know EKF SLAM and do not know how to build a local map using a single observation, then you need a first lesson on SLAM.

V. LIMITATIONS AND FURTHER IMPROVEMENTS OF THE CODE

A. Data association

In the current version of the code, the selection of potential matched features and the recovery of the corresponding covariance submatrix [2] is implemented, but the actual matching is only implemented using a simple nearest neighbor approach. A more robust matching algorithm such as JCBB [3] will help improving the quality of the data association. If you have developed your own matching algorithm, just replace the file joint_match_beacons_NN.m with your code.

• For 535 step simulation data set, the data association is trivial since the data set is very small.
• For 35188 step simulation data set, the correct data association result can be achieved using I-SLSJF, but a slightly wrong result will occur if SLSJF is used. This is because the global map estimate of I-SLSJF is more accurate than that of SLSJF.
• For 8240 step simulation data sets, the data association result is wrong when fusing the last local map. Sure this is due to the large loop involved. We deliberately leave the data like it is to show that data association is more challenging when closing large loop (JCBB is not able to handle this scenario, you need some kind of global localization technique to resolve this).
• For Victoria Park data set, although the exact data association result is hard to achieve, the final map result is very similar to that when data association is assumed.
• For the DLR-Spatial-Cognition data set with 200 local maps, wrong data association starts to occur as early as fusing the 17-th local map. For the DLR-Spatial-Cognition data set with 3927 local maps, wrong data association starts to occur when fusing the 280-th local map. Most of these problems can be fixed by using JCBB, but some scenarios can not be handled by JCBB. For example, when fusing local map 1148 for the DLR-Spatial-Cognition data set with 3927 local maps, global localization is needed since a large loop is closed.

Also, the parameters used in data association is problem dependent, you need to know some information about the feature density etc. to choose the parameters.

B. Smoothing using least squares

The current implementation of the smoothing step is not very efficient, there is still a lot of rooms to improve the speed of the algorithm.

VI. MATLAB CODE OF SOME OTHER SLAM ALGORITHMS

Recently, we have extended the I-SLSJF to 3D [4] and also developed a new map joining algorithm – Iterated D-SLAM Map Joining (I-DMJ) algorithm [5]. In I-DMJ, no robot poses is included in the global map, so the state dimension only depends on the number of features and is not related to the length of the robot trajectory at all.

If we get enough encouragement from the SLAM community, we will publish the source code of these two algorithms shortly.
REFERENCES


