Document for Linear SLAM MATLAB and C/C++ Source Code

Liang Zhao, Shoudong Huang and Gamini Dissanayake
Centre for Autonomous Systems
Faculty of Engineering and Information Technology
University of Technology Sydney, Australia
Liang.Zhao-1@uts.edu.au
March 22, 2013

Abstract

This document provides some details about how to use the MATLAB and C/C++ source code of Linear SLAM algorithm for both pose feature and pose graph SLAM datasets, 2D and 3D, based on submap joining. The code is written by Liang Zhao (Liang.Zhao-1@uts.edu.au), Shoudong Huang (Shoudong.Huang@uts.edu.au) and Gamini.Dissanayake (Gamini.Dissanayake@uts.edu.au). The code with some simulation and experimental datasets are available on the OpenSLAM website. Please contact Liang Zhao if you have any questions/comments about the code.

I. Introduction

Linear SLAM is a linear approach to the pose feature and pose graph SLAM problems based on submap joining. The solution to a large scale SLAM problem that requires joining a number of local submaps either sequentially or in a more efficient Divide and Conquer manner, can be obtained through solving a sequence of linear least squares problems. The proposed Linear SLAM technique is applicable to both pose feature and pose graph SLAM, in two and three dimensions, and does not require any assumption on the character of the covariance matrices or an initial guess of the state vector.

Same as other submap joining algorithm such as Sparse Local Submap Joining Filter (SLSJF) or I-SLSJF, the input of Linear SLAM is a sequence of submaps, with the end pose of the i^{th} local map is the start pose of the $(i+1)^{th}$ local map, then all the local maps can be linked together. For more details about submap joining algorithm, please refer to project "2D I-SLSJF" in OpenSLAM.

Two submap joining methods are used in Linear SLAM algorithm

- Sequential
- Divide and Conquer

and we suggest to use "Divide and Conquer" instead of "Sequential" because of its low computational cost.

The output of Linear SLAM algorithm is a global map estimates as well as the corresponding information matrix. For convenience, the result of the global map is transformed into the coordinate frame of the first robot pose.

II. Start to Run

A. Datasets

The code are accompanied by some simulation and experimental datasets, for both pose feature and pose graph SLAM, 2D and 3D. Thus, user can run the code directly by using these demo datasets (see Table I).

The following datasets are ready to use:

1) 2D Pose Feature Datasets:

- VicPark 200 local maps
- VicPark 6898 local maps
- DLR 200 local maps
- DLR 3298 local maps
- 8240 data 50 local maps
- 35188 data 700 local maps

2) 3D Pose Feature Datasets:

- Simu 3D 870 Loop

3) 2D Pose Graph Datasets:

- Intel
- manhattanOlson3500
- city10000

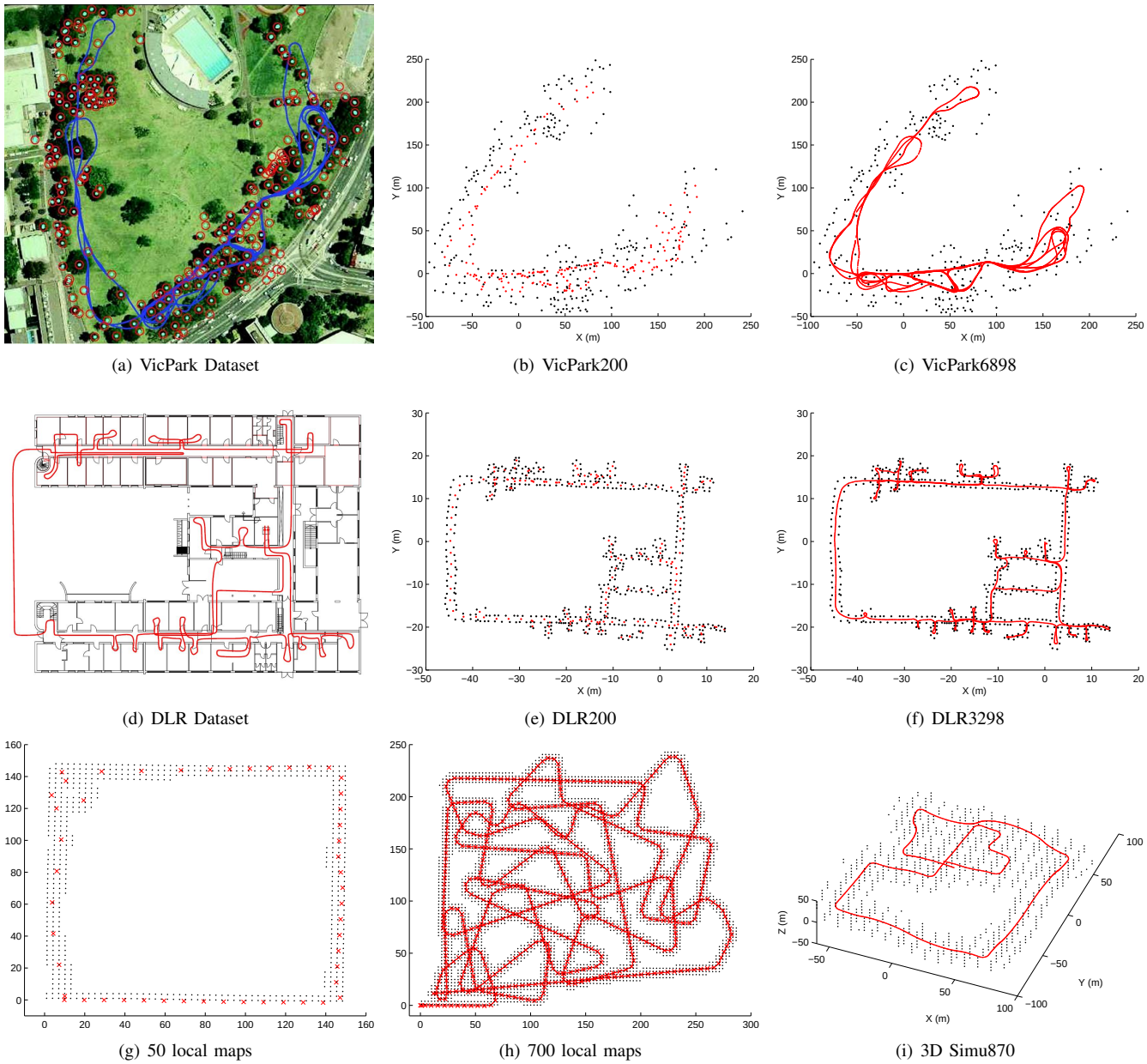
4) 3D Pose Graph Datasets:

- parking garage
- sphere2500

TABLE I

2D AND 3D POSE FEATURE AND POSE GRAPH SLAM DATASETS

Dataset	Number of Local Map	Dataset Type	Dimensions
VicPark200	200	Pose Feature	2D
DLR200	200	Pose Feature	2D
VicPark6898	6898	Pose Feature	2D
DLR3298	3298	Pose Feature	2D
8240 data 50 local maps	50	Pose Feature	2D
35188 data 700 local maps	700	Pose Feature	2D
3D Simu870	870	Pose Feature	3D
Intel	942	Pose Graph	2D
manhattanOlson3500	3499	Pose Graph	2D
city10000	9999	Pose Graph	2D
parking garage	1660	Pose Graph	3D
sphere2500	2499	Pose Graph	3D

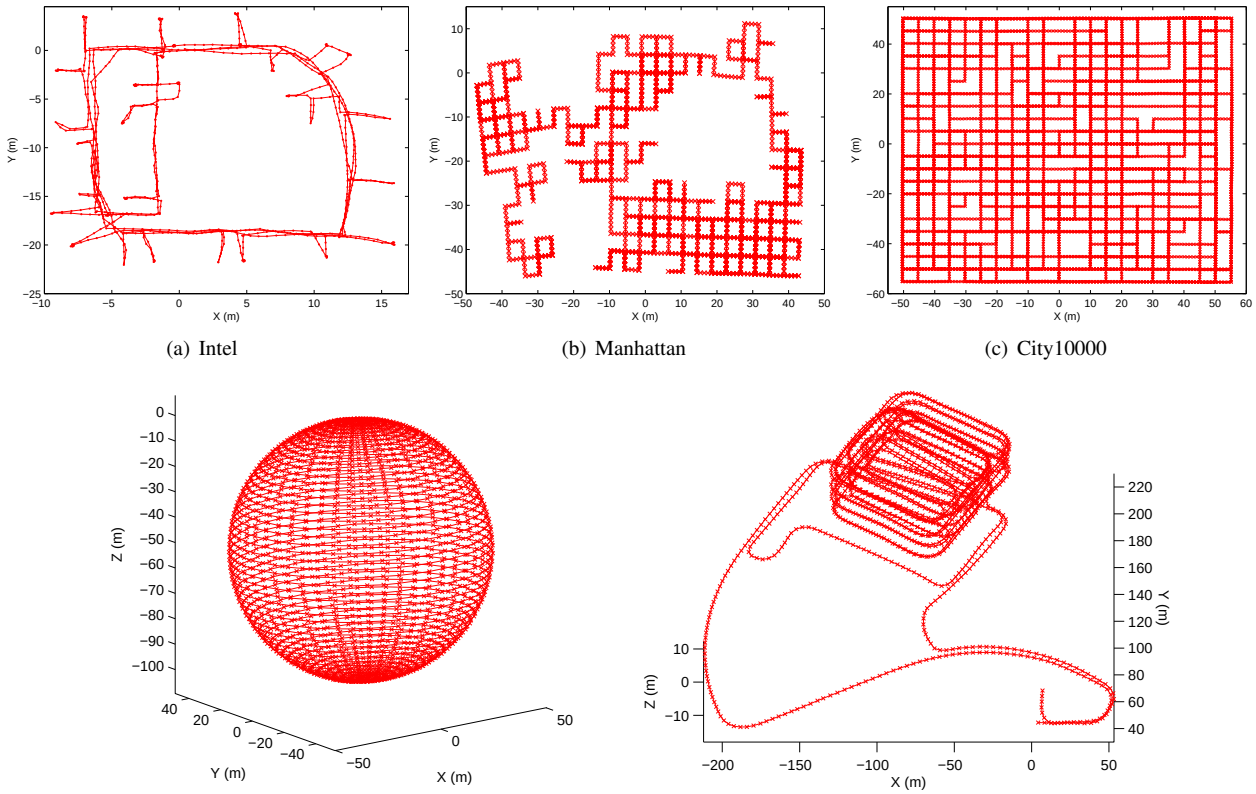


(a) VicPark Dataset (b) VicPark200 (c) VicPark6898

(d) DLR Dataset (e) DLR200 (f) DLR3298

(g) 50 local maps (h) 700 local maps (i) 3D Simu870

Fig. 1. Linear SLAM results of 2D and 3D pose feature SLAM datasets.



(a) Intel (b) Manhattan (c) City10000

(d) Sphere (e) Parking garage

Fig. 2. Linear SLAM results of 2D and 3D pose graph SLAM datasets.

B. Start to Run the MATLAB Code

Suppose the user is going to run the code using "VicPark 200 local maps" 2D Pose Feature SLAM dataset, by "Divide and Conquer" map joining, the code can be run by the following steps:

1) Open the "Main.m" Function:

2) Select the Map Joining Method:

Uncomment "Method = 'Divide&Conquer"' and comment "Method = 'Sequential"' as follows

```
% Method = 'Sequential';
Method = 'Divide&Conquer';
```

3) Select the Dataset Type:

Uncomment "DataType = '2D Pose Feature"' and comment all the other dataset types as follows

```
DataType = '2D Pose Feature';
% DataType = '3D Pose Feature';
% DataType = '2D Pose Graph';
% DataType = '3D Pose Graph';
```

4) Select the Dataset:

Uncomment "Dataset = 'VicPark_200_local_maps"' and comment all the other datasets as follows

```
Dataset = 'VicPark_200_local_maps';
% Dataset = 'VicPark_6898_local_maps';
% Dataset = 'DLR_200_local_maps';
% Dataset = 'DLR_3298_local_maps';
% Dataset = '8240_data_50_local_maps';
% Dataset = '35188_data_700_local_maps';
```

5) Run the "Main.m" Function:

Notice: **Please select the "Dataset" corresponding to the right "DataType".**

C. Start to Run the C/C++ Code in Linux

1) Install:

Linear SLAM C/C++ in Linux requires cmake, SuiteSparse. On Ubuntu/Debian these dependencies are resolved by installing the following packages.

- sudo apt-get install cmake
- sudo apt-get install libsuitesparse-dev

We recommend a so-called out of source build which can be achieved by the following command sequence.

- cd LinearSLAM_C/linux
- mkdir build
- cd build
- cmake ..
- make
- sudo make install

The binaries will be placed in bin and the libraries in lib which are both located in the top-level folder.

2) Run:

The demonstrated datasets can be run by one of the following command lines

- LinearSLAM -path VicPark_200_local_maps -num 200 -meth DC -type 2DPF -p pose.txt -f feature.txt
- LinearSLAM -path VicPark_6898_local_maps -num 6898 -meth DC -type 2DPF -p pose.txt -f feature.txt
- LinearSLAM -path DLR_200_local_maps -num 200 -meth DC -type 2DPF -p pose.txt -f feature.txt
- LinearSLAM -path DLR_3298_local_maps -num 3298 -meth DC -type 2DPF -p pose.txt -f feature.txt
- LinearSLAM -path 8240_data_50_local_maps -num 50-meth DC -type 2DPF -p pose.txt -f feature.txt
- LinearSLAM -path 5188_data_700_local_maps -num 700 -meth DC -type 2DPF -p pose.txt -f feature.txt
- LinearSLAM -path Simu_3D_870_Loop -num 870 -meth DC -type 3DPF -p pose.txt -f feature.txt
- LinearSLAM -path Intel -num 942 -meth DC -type 2DPG -p pose.txt
- LinearSLAM -path manhattanOlson3500 -num 3499 -meth DC -type 2DPG -p pose.txt
- LinearSLAM -path city10000 -num 9999 -meth DC -type 2DPG -p pose.txt
- LinearSLAM -path parking-garage -num 1660 -meth DC -type 3DPG -p pose.txt
- LinearSLAM -path sphere2500 -num 2499 -meth DC -type 3DPG -p pose.txt

If the user wants to know all the commands, one can type

- LinearSLAM -help

and the following help information will be on the command window

LinearSLAM General Options:

-path	Set Data Path.
-st	Set Path to Save Final State Vector.
-i	Set Path to Save Information Matrix.
-p	Set Path to Save Final Poses.
-f	Set Path to Save Final Features.
-num	Set Local Map Number.
-meth	Set Process Method, including SEQ (Sequential) and DC (Divide and Conquer).
-type	Set Data Type, including 2DPF (2D Pose Feature), 3DPF (3D Pose Feature), 2DPG (2D Pose Graph) and 3DPG (3D Pose Graph).

Notice: **When running pose graph datasets, please do not try to save the final features in the command line.**

D. Start to Run the C/C++ Code in Windows

The CHOLMOD and GotoBLAS2 software packages are used in the C/C++ code in Windows to solve linear equations.

We recommend the user to use Microsoft Visual Studio 2010 or higher version to compile the code. The user can open and run solution "LinearSLAM.sln" in folder "windows/LinearSLAM" to get the results of the demonstrated datasets. We provide class "CLinearSLAM" to accomplish the whole LinearSLAM functions.

The user can easily set all the inputs in "_tmain()" function to run a specific dataset by the following steps:

1) Open the "LinearSLAM.sln" Solution:

2) Select the Map Joining Method:

Uncomment "method = LinearSLAM_DivideConquer" and comment "method = LinearSLAM_Sequential" as follows

```
//LinearSLAM_Method method = LinearSLAM_Sequential;
LinearSLAM_Method method = LinearSLAM_DivideConquer;
```

3) Select the Dataset Type:

Uncomment "dataType = LinearSLAM_2DPoseFeature" and comment all the other dataset types as follows

```
LinearSLAM_DataType dataType = LinearSLAM_2DPoseFeature;
//LinearSLAM_DataType dataType = LinearSLAM_3DPoseFeature;
//LinearSLAM_DataType dataType = LinearSLAM_2DPoseGraph;
//LinearSLAM_DataType dataType = LinearSLAM_3DPoseGraph;
```

4) Select the Dataset:

Uncomment "szData = "../../../DatasetsForC/VicPark_200_local_maps"; nMapCount = 200" and comment all the other datasets as follows

```
char* szData = "../../../DatasetsForC/VicPark_200_local_maps"; nMapCount = 200;
//char* szData = "../../../DatasetsForC/VicPark_6898_local_maps"; nMapCount = 6898;
//char* szData = "../../../DatasetsForC/DLR_200_local_maps"; nMapCount = 200;
//char* szData = "../../../DatasetsForC/DLR_3298_local_maps"; nMapCount = 3298;
//char* szData = "../../../DatasetsForC/8240_data_50_local_maps"; nMapCount = 50;
//char* szData = "../../../DatasetsForC/35188_data_700_local_maps"; nMapCount = 700;
```

5) Select Results to Save:

Select the results one want to save, and set the path and name of the saved results. Suppose the user want to save the results of poses and features of a 2D pose feature dataset,

```
szPose = "../../../pose.txt";
szFeature = "../../../feature.txt";
//szSt = "../../../State.txt";
//szInfo = "../../../information.txt";
```

6) Compile and Run: ***Notice:*** **When running pose graph datasets, please comment the feature path.**

III. Use Your Own Datasets

A. Use Your Own Datasets in MATLAB Code

To use your own datasets in the Linear SLAM algorithm, the i^{th} local map should be saved with the format as follows:

1) : The i^{th} local map is saved in the ".mat" MATLAB data format with the file name as "localmap_i.mat".

2) : Three local map data are inside the ".mat" file:

- **Ref**: A number represents the ID of the robot pose which is the coordinate frame of this local map.
- **st**: A two columns matrix, in which the second column is the estimate of the state vector of this local map, with the order of first poses then features. The format is as $[\cdots x_{P_j}, y_{P_j}, \phi_{P_j} \cdots, \cdots x_{F_k}, y_{F_k} \cdots]^T$ for 2D, and $[\cdots x_{P_j}, y_{P_j}, z_{P_j}, \alpha_{P_j}, \beta_{P_j}, \gamma_{P_j} \cdots, \cdots x_{F_k}, y_{F_k}, z_{F_k} \cdots]^T$ for 3D. The first column of **st** are the global IDs of the poses and features in the state vector. we use positive IDs for features, zero and negative IDs for poses. For pose graph, the IDs of poses are zero or positive.
- **I**: The information matrix corresponding to the estimate of the state vector in **st**.

The Euler angles $[\alpha, \beta, \gamma]^T$ of a robot pose in 3D and the rotation matrix R are defined as follows:

$$R = R_X(\gamma)R_Y(\beta)R_Z(\alpha) \tag{1}$$

where

$$R_Z(\alpha) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_Y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix}, \quad R_X(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & \sin(\gamma) \\ 0 & -\sin(\gamma) & \cos(\gamma) \end{bmatrix}. \tag{2}$$

The 3D angle-to-matrix and matrix-to-angle functions are in the source code called "RMatrixYPR22.m" and "InvRotMatrixYPR22.m". One can use them to convert to the right format.

After all the local maps are prepared, one can put them in the folder named "test_your_own_dataset". Uncomment "Dataset = 'test_your_own_dataset"' and select the right "DataType" in the "Main.m" function. Then run it to get the Linear SLAM result using your own dataset.

B. Use Your Own Datasets in C/C++ Code

The i^{th} local map is defined the same way as in Section III-A. And the **Ref**, **st** and **I** of this local map are written in one text file as

Ref
N_st
st
I

with the name "localmap_i.txt", where **N_st** is the number of the variables in the state vector **st**.

1) In Linux: Put all the local maps in the folder named "test_your_own_dataset", and run the command

---

- LinearSLAM -path test_your_own_dataset -num 22 -meth DC -type 2DPF -p pose.txt -f feature.txt

---

with specified local map number, "DataType" and results to save.

2) In Windows: Uncomment "szData = "../../../DatasetsForC/test_your_own_dataset"; nMapCount = 22", select the right "DataType", input the local map number in the "_tmain()" function and set the results to save. Then compile and run to get the Linear SLAM result using your own dataset.

IV. Academic Usage and Paper Reference

The paper details the Linear SLAM algorithm is

- L. Zhao, S. Huang and G. Dissanayake, "Linear SLAM: A Linear Solution to the Pose Feature and Pose Graph SLAM based on Submap Joining," Submitted to *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

If this code is used for academic work, please reference this paper.

If you are interested in the paper for further understanding about the Linear SLAM algorithm, please send an email to the first author.