Document for ParallaxBA C/C++ Source Code

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

Abstract

This document provides some details about how to use the C/C++ source code of ParallaxBA algorithm, bundle adjustment using parallax angle parametrization for monocular SLAM or structure from motion problem. The code is written by Liang Zhao (Liang.Zhao-1@uts.edu.au), Shoudong Huang (Shoudong.Huang@uts.edu.au), Yanbiao Sun (syb51@pku.edu.cn) and Gamini.Dissanayake (Gamini.Dissanayake@uts.edu.au). The code with some experimental datasets are available on the OpenSLAM website. Please contact Liang Zhao or Yanbiao Sun if you have any questions/comments about the code.

I. Introduction

ParallaxBA is a new bundle adjustment algorithm based on parallax angle feature parametrization. It is demonstrated that the new parametrization can reliably represent both nearby and distant features under different camera motions. Because good information of the proposed feature parameters is always available, the singularity of the information matrix due to near zero parallax can be avoided. Moreover, using the new parameters results in better properties of the objective function as compared with those using traditional Euclidean XYZ and inverse depth parametrization. Thus ParallaxBA has better convergence properties as compared with existing BA algorithms.

The input of ParallaxBA is the matched image points, camera calibration matrix, as well as the initial values of camera poses and features (if provided). The format will be described later. The output of ParallaxBA is the optimized 3D feature locations as well as the camera poses.

II. Start to Run

A. Datasets

The code are accompanied by some experimental datasets. Thus, user can run the code directly by using these demo datasets (see Table I).

B. Interface: class IParallaxBA

Class IParallaxBA is provided as the interface of this library. On windows platform, we can set different parameters to run ParallaxBA as following.

pba_run(bool bRobust, bool bLM, int nMaxIter, char szCam, char* szFea, char* szXYZ, char* szCalib, char* szReport, char* szPose, char* sz3D, double Tau);*

- bRobust when it is true, BA with robust kernel will be run which default kernel function is Cauchy.

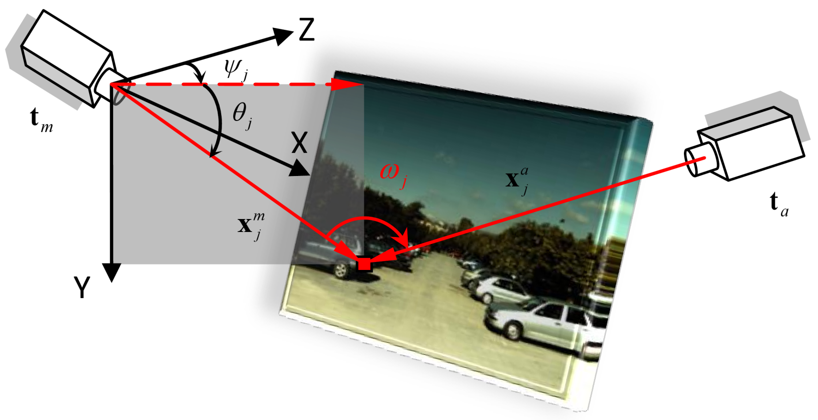


Fig. 1. Parallax angle parametrization for feature.

TABLE I
EXPERIMENTAL DATASETS FOR PARALLAXBA

Dataset	Number of cameras	Number of points	Number of projections
Malaga	170	58404	167285
Village	90	305719	779268
College	468	1236502	3107524
NewCollege	3500	449096	2124449
Venice	871	530304	2838740
Toronto	13	139648	297097
Vaihingen	20	554914	1204888
DunHuan	63	237934	565269
Jinan	76	1228959	2864740

TABLE II
CONVERGENCE OF PARALLAXBA

	Malaga	Village	College	New College	Venice
Initial MSE	337.457518	28170.979450	202329.447636	23.354933	560.817183
Final MSE	0.109208	0.083716	0.734738	0.979580	7.307519
Method	LM	GN	GN	GN	LM
Iteration	62	6	12	13	43
Solve	81	6	12	13	74
Time/Iter (sec)	0.25	0.67	2.71	5.35	9.69
Windows/Linux	0.34	0.96	3.85	7.60	12.45
Total Time (sec)	15.29	5.07	37.14	77.52	666.83
Windows/Linux	24.91	6.98	51.55	108.39	783.23

*Run on an Intel i5-3210M@2.5GHz CPU.

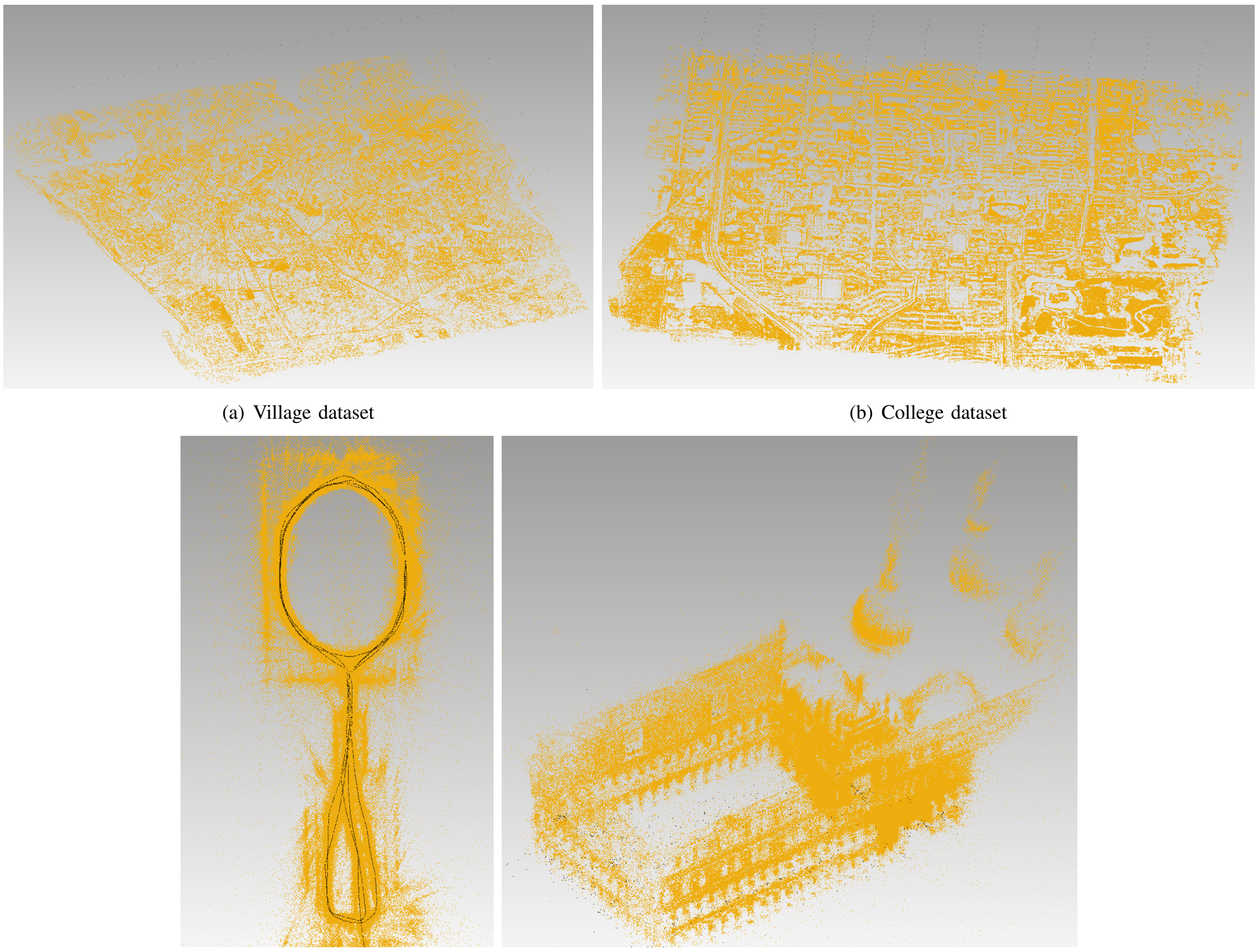

(a) Village dataset
(b) College dataset
(c) New College dataset
(d) Venice dataset

Fig. 2. The results of ParallaxBA.

TABLE III

PARALLAXBA COMMAND FOR LINUX

command	content
-help	list all commands and corresponding contents.
-cam	input initial camera pose.
-fea	input features serving as BA observation.
-calib	input calibration matrix.
-xyz	input initial 3D XYZ.
-pose	output optimal camera poses.
-3D	output optimal 3D point cloud.
-report	output BA report.
-solve	set Levenberg-Marquart (LM) and Gauss-Newton(GN), its value must be LM or GN.
-i	input maximum number of iterations.
-t	set Levenberg-Marquart parameters: Tau.
-robustKernel	use Cauchy or Huber robust kernel estimation, its value must be Cauchy or Huber.
-robustKernelWidth	set width for the robust kernel function.

- bLM when it is true, Levenberg-Marquardt (LM) will be used. On the contrary, Gauss-Newton will be used.
- nMaxIter denotes the maximum number of iterations.
- szCam denotes the initial poses path.
- szFea denotes the features path.
- szXYZ denotes the initial 3D points path.
- szCalib denotes the calibration matrix path.
- szReport denotes the saved report path.
- szPose denotes the final convergent poses path.
- sz3D denotes the final convergent 3D features path.
- Tau denotes the important parameter of LM.

On Linux platform, many commands are provide in Table III.

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

1) Install:

ParallaxBA C/C++ in Linux requires cmake, Eigen, SuiteSparse. On Ubuntu/Debian these dependencies are resolved by installing the following packages. If one want to improve the efficiency of solving sparse linear systems, we recommend to download and install GotoBLAS.

```
- sudo apt-get install cmake
- sudo apt-get install libeigen3-dev
- 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 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. First, one needs to enter the dataset folder. Then choose the corresponding command lines as follows.

```
ParallaxBA -cam Cam170.txt -fea Feature170.txt -calib cal170.txt -solve LM -report report.txt
ParallaxBA -cam Cam90.txt -fea Feature90.txt -calib cal90.txt -solve GN -report report.txt
ParallaxBA -cam Cam468.txt -fea Feature468.txt -calib cal468.txt -solve GN -report report.txt
ParallaxBA -cam Cam3500.txt -fea Feature3500.txt -calib cal3500.txt -xyz XYZ3500.txt -solve GN -report report.txt
ParallaxBA -cam Cam871.txt -fea Feature871.txt -xyz XYZ871.txt -solve LM -report report.txt
ParallaxBA -cam Cam13.txt -fea Feature13.txt -calib cal13.txt -solve GN -report report.txt
ParallaxBA -cam Cam20.txt -fea Feature20.txt -calib cal20.txt -solve GN -report report.txt
ParallaxBA -cam Cam63.txt -fea Feature63.txt -calib cal63.txt -solve GN -report report.txt
ParallaxBA -cam Cam76.txt -fea Feature76.txt -calib cal76.txt -solve LM -report report.txt -t 0.001
```

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

The Eigen, 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 “demo.sln” in folder ”windows/msvc/demo” to get the results of the demonstrated datasets. We provide a virtual class ”IParallaxBA” to accomplish the whole ParallaxBA functions.

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

1) Open the “demo.sln” Solution:

2) Create IParallaxBA instance:

IParallaxBA* pBA = newParallaxBA();

3) Select the dataset to run:

Malaga dataset

```
char* szCam = "../../../data/Malaga/Cam170.txt";
char* szFea = "../../../data/Malaga/Feature170.txt";
char* szCalib = "../../../data/Malaga/cal170.txt";
char* szReport = "../../../data/Malaga/report.txt";
bool bLM = true;
pBA→ pba_run( false, bLM, 100, szCam, szFea, NULL, szCalib, szReport, NULL, NULL );
```

Village dataset

```
char* szCam = "../../../data/Village/Cam90.txt";
char* szFea = "../../../data/Village/Feature90.txt";
char* szCalib = "../../../data/Village/cal90.txt";
char* szReport = "../../../data/Village/report.txt";
bool bLM = false;
pBA→ pba_run( false, bLM, 100, szCam, szFea, NULL, szCalib, szReport, NULL, NULL );
```

College dataset

```
char* szCam = "../../../data/College/Cam468.txt";
char* szFea = "../../../data/College/Feature468.txt";
char* szCalib = "../../../data/College/cal468.txt";
char* szReport = "../../../data/College/report.txt";
bool bLM = false;
pBA→ pba_run( false, bLM, 100, szCam, szFea, NULL, szCalib, szReport, NULL, NULL );
```

NewCollege dataset

```
char* szCam = "../../../data/NewCollege/Cam3500.txt";
char* szFea = "../../../data/NewCollege/Feature3500.txt";
char* szCalib = "../../../data/NewCollege/cal3500.txt";
char* szXYZ = "../../../data/NewCollege/XYZ3500.txt";
char* szReport = "../../../data/NewCollege/report.txt";
bool bLM = false;
pBA→ pba_run( false, bLM, 100, szCam, szFea, szXYZ, szCalib, szReport, NULL, NULL );
```

venice dataset

```
char* szCam = "../../../data/Venice/Cam871.txt";
```

char szFea = "../../../data/Venice/Feature871.txt";*
char szXYZ = "../../../data/Venice/XYZ871.txt";*
char szReport = "../../../data/Venice/report.txt";*
bool bLM = ture;
pBA→ pba_run(false, bLM, 100, szCam, szFea, szXYZ, NULL, szReport, NULL, NULL);

---

other datasets

For other datsets such as Toronto, Vaihingen, DunHuan, Jinan, uncomment the corresponding code.

---

4) delete IParallaxBA instance:
freeParallaxBA(pBA);

III. Use Your Own Datasets

To use your own datasets in the ParallaxBA algorithm, the input and the formats are described as follows.

A. Camera Calibration Matrix

1) Single Camera Calibration Matrix: If the calibration matrices for all the cameras are the same, a camera calibration file needs to be provide which contains the focal length and principle points in pixels.

$$K = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

2) Multiple Camera Calibration Matrices: If the calibration matrices for the cameras are different (like Venice dataset), one can provide the camera calibration parameters of each camera just after the initial value of the camera in the initial camera poses file (see the following section).

B. The Initial Camera Poses

The i^{th} camera pose is presented as a six-vector in the i^{th} row of the initial camera pose file. The first three are the Euler angles $[\alpha, \beta, \gamma]^T$, and the last three is the camera centre **t**.

The Euler angles $[\alpha, \beta, \gamma]^T$, the centre of the camera pose and the rotation matrix R are defined as follows:

$$\begin{bmatrix} u \\ v \end{bmatrix} \propto KR(\mathbf{F} - \mathbf{t}) \tag{2}$$

$$R = R_X(\gamma)R_Y(\beta)R_Z(\alpha) \tag{3}$$

where

$$R_Z(\alpha) = \begin{bmatrix} \cos(\alpha) & \sin(\alpha) & 0 \\ -\sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad R_Y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix}, \quad R_X(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & \sin(\gamma) \\ 0 & -\sin(\gamma) & \cos(\gamma) \end{bmatrix}. \tag{4}$$

For multiple camera calibration matrices, there are eight values in each row for a camera pose, where the last two values are focal length. And in this condition, the origin of uv in each image must be the principle point.

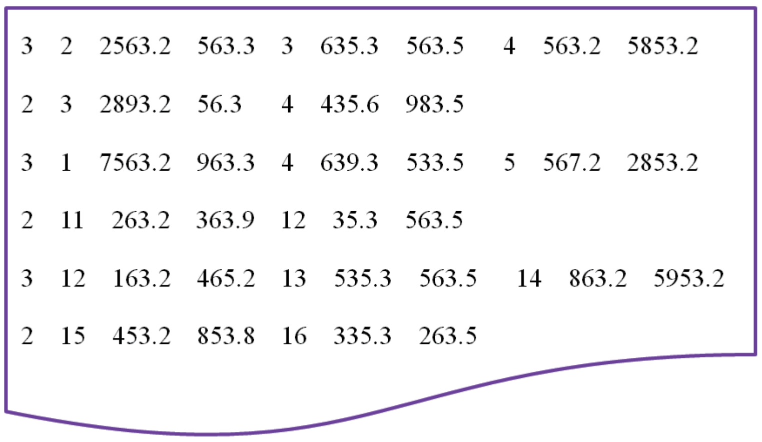
C. The Image Point Features

The format of the matched image point features is the same as the one of SBA. As show in Fig. 3, the first value represents the total number of projection corresponding to a same 3D features, afterward three ones are one projection, including image index and image points. Note that the first image index is 0.

D. Initial Feature XYZ Location (optional)

ParallaxBA can initialize the parallax angle features parameters in the code, so a initial value of features are not required.

However, one can still use the initial XYZ locations of features to initialize the parallax angle features parameters. The j^{th} features **F** is presented as a 3-vector of X, Y and Z in the j^{th} row of the initial feature XYZ location file.



```
3   2    2563.2   563.3   3    635.3    563.5     4    563.2    5853.2

2   3    2893.2   56.3     4    435.6    983.5

3   1    7563.2   963.3   4    639.3    533.5     5    567.2    2853.2

2   11    263.2   363.9   12    35.3    563.5

3   12    163.2   465.2   13    535.3    563.5     14    863.2    5953.2

2   15    453.2   853.8   16    335.3    263.5
```

Fig. 3. Feature file format

IV. Academic Usage and Paper Reference

The papers details the ParallaxBA algorithm are

- L. Zhao, S. Huang, L. Yan, and G. Dissanayake, “Parallax angle parametrization for monocular SLAM,” In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3117-3124, 2011.
- L. Zhao, S. Huang, Y. Sun, L. Yan and G. Dissanayake, “ParallaxBA: Bundle Adjustment using Parallax Angle Feature Parametrization,” Submitted to *International Journal of Robotics Research (IJRR)*, August, 2013.

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