Multicore Bundle Adjustment Manual

Changchang Wu

University of Washington at Seattle

Contents

1. Introduction

Multicore bundle adjustment is a parallel-accelerated implementation of bundle adjustment for multicore CPU and GPU. By restructuring the non-linear optimization problem, the overall computation becomes dominated by series of simple matrix-vector operations. The matrix-vector operations are then parallelized with a combination of multi-threading and SIMD (Single Instruction Multiple Data). Additionally, the problem restructuring enables tremendous memory saving by computing Jacobians on-the-fly during matrix-vector multiplication. I prefer to call it **PBA** (Parallel Bundle Adjustment), which corresponds to the class interface **ParallelBA**.

2. Library Interface

To use the library for your projects, you can directly work with the ParallelBA class object, or use the two c-like functions I developed for the users of bundler and SBA.

The usage of ***ParallelBA*** is demonstrated in src/driver/driver.cpp. Given the special alignment required by SIMD on both CPU and GPU, the input data must be first converted to special internal formats. Cameras, 3D points and 2D measurements must be stored as class CameraT, Point3D and Point2D respectively, which are defined in src/pba/DataInterface.h. The functions for converting to/from other common data formats are provided.

The ***run_sfm_pba*** function is designed for easy integration with the popular bundler software, and you can replace the ***run_sfm*** function in bundler. Be careful that not all parameters of run_sfm are supported, and the function will simply do nothing if unsupported parameters are specified. When integrating with bundler, you need to **–lpba** to your makefile.

3. Suggestion on SfM

You should use more LM iterations for the first few cameras if the two-view initialization is bad (decomposed from Fundamental matrix rather than Essential matrix). An alternative is to switch from regular BA to PBA only after a few cameras (e.g. 5).

Since the library relies on single-precision math, it is recommended to add a filtering step in the reconstruction to **remove 3D points that are close to (or behind) camera planes** before BA. With unlucky conversion errors from double precision to single precision, it is possible that near- degenerate points are incorrectly moved to the wrong sides of the camera planes.

4. Parameter System

Our bundle adjustment provides two parameter control schemes. You can either specify command line options to ***ParallelBA::ParseParam***, or directly modify the members of the internal configuration objet ***ParallelBA::GetInternalConfig().***

Command line options	ConfigBA member variable	default	comments
Controling the number of LM and CG iterations			
-lmi <int>	__lm_max_iteration	50	Maximum LM iteration
-cgi <int>	__cg_min_iteration	10	Minimum CG iteration per LM
-cgim<int>	__cg_max_iteration	100	Maximum CG iteration per LM
-budget <int>	__bundle_time_budget	INF	Set a one-time time budget for LM
Stopping criteria on quality			
-lmd <float>	__lm_delta_threshold	1e-6	Quit LM on small absolute change
-lmg <float>	__lm_gradient_threshold	1e-10	Use only if (__lm_check_gradient)
-chkg	__lm_check_gradient	false	Quit LM if gradient is small enough
-lme<float>	__lm_mse_threshold	0.25	Quit LM if MSE is small enough
-cgn <float>	__cg_norm_threshold	0.1	Quit CG if norm is small enough
-cgg <float>	__cg_norm_guard	1.0	Quit CG if norm incorrectly gets larger
LM behavior			
-damp <float>	__lm_initial_damp	0.001	Initial damping factor
-dmin <float>	__lm_minimum_damp	1e-10	Minimum damping factor
-dmax <float>	__lm_maximum_damp	1e+5	Maximum damping factor
-id (false)	__lm_use_diagonal_damp	true	Use diag(Jt*J) or I as damping vector
-schur	__cg_schur_complement	false	Use implicit Schur complement
Camera model			
-calibrated	__fixed_focallength	false	Keep focal lengths unmodified
-pd (1) -md (-1)	__use_radial_distortion	0	1 for projection distortion, -1 for measurement distortion
-r00	__reset_initial_distortion	false	Ignore the input radial distortion
Verbosity control			
-v <int>	__verbose_level	2	How detailed are the messages printed
-vcgi	__verbose_cg_iteration	false	Show details of PCG?
-vall	__verbose_allocation	false	Show details of memory allocation?

Note on command line options :

1. <> , the parameters are set to the user-specified following value (you must specify one).
2. () , the parameters will be set to the value in () when the option is used.
3. For the other options, the parameters will be set to true when the option is used.
4. The command line options are used by ParallelBA::ParseParam

5. Camera Model & Radial Distortion

By default, PBA will use a 7 parameter camera model: 1 for focal length, 3 for rotation, and 3 for translation. We also implemented **TWO** types of single-parameter **radial distortion** as follows:

PBA_PROJECTION_DISTORTION:

Single value parameter; applies to projections
Set __use_radial_distortion = 1 or use commandline option -pd

----

Given camera K[R T], K = [f, 0 0; 0 f 0; 0 0 1], radial distortion r, and a 3D point X.

The projection is [x, y, z]' = (RX + T) -> (xn, yn)' = (x/z, y/z)'

Let r2 = r * (xn * xn + zn * zn),

The undistorted projection is (1 + r2) * f * (xn, yn)'

Let the measurement be [mx, my]

The reprojection error is **[(1 + r2)* f * xn - mx, (1 + r2) * f * yn - my]**

You can use the second order parameter from the Matlab Camera Calibration Toolbox.

PBA_MEASUREMENT_DISTORTION : (used by VisualSFM)

Single value parameter; applies to measurements
Set __use_radial_distortion = -1 or use commandline option -md

----

Given camera K[R T], K = [f, 0 0; 0 f 0; 0 0 1], radial distortion r, and a 3D point X.

The reprojection in the image is [x, y, z]' = K (RX + T) -> (x/z, y/z)'

Let the distorted measurement be [mx, my],

The distortion factor is r2 = r * (mx * mx + my * my)

The undistorted measurement is (1 + r2) * [mx, my]

Then, the reprojection error is **[x/z - (1 + r2) mx, y /z - (1 + r2) my]**

This measurement distortion is easy for computing reprojection/Jacobians of feature points, but slightly harder for generating the undistorted images (need to solve cubic equations)

If you have the second order radial distortion r' from the Matlab Camera calibration Toolbox, the approximate value for the radial distortion here can be -r'/f/f