Mining Structure Fragments for Smart Bundle Adjustment

Luca Carlone[1]
luca.carlone@gatech.edu

Pablo Fernandez Alcantarilla[2]
pablo.alcantarilla@crl.toshiba.co.uk

Han-Pang Chiu[3]
han-pang.chiu@sri.com

Zsolt Kira[4]
Zsolt.Kira@gtri.gatech.edu

Frank Dellaert[1]
dellaert@cc.gatech.edu

[1] Georgia Institute of Technology, College of Computing, USA

[2] Toshiba Research Europe, Cambridge Research Laboratory, UK

[3] SRI International, Center for Vision Technologies, USA

[4] Georgia Tech Research Institute, ATAS Laboratory, USA

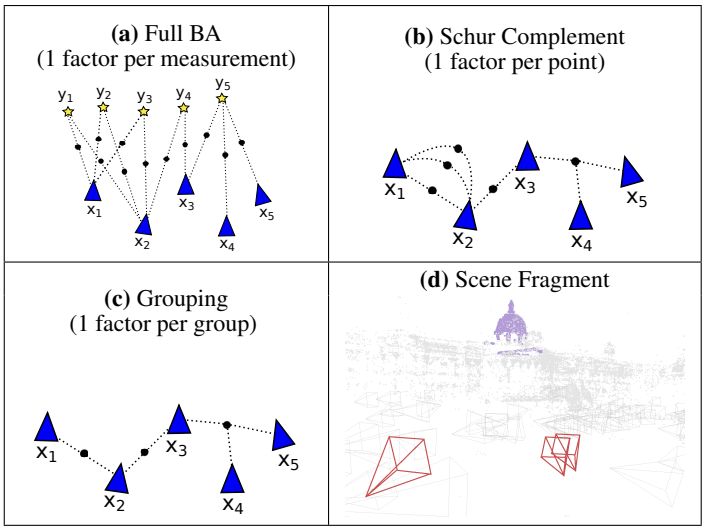

Table 1: (a) Factor graph $\mathcal{G}$ corresponding to standard BA. Points are shown as yellow stars, cameras are blue triangles, and factors are denoted with black dots. (b) Factor graph obtained after eliminating points from $\mathcal{G}$. (c) Factor graph obtained by grouping factors corresponding to points that are co-visible by the same patter of cameras. (d) A scene *fragment* is a group of N points that are visible in $M < N$ cameras. For those groups it is convenient to use an *explicit* representation for the Schur complement, as this reduces the computational cost of each conjugate gradient iteration.

Efficient bundle adjustment (BA) is an important prerequisite to a number of practical applications, ranging from 3D modeling and photo tourism, to hand-eye calibration, augmented reality, and autonomous navigation.

BA and Conjugate Gradient. BA estimates camera parameters and scene structure via nonlinear optimization. State-of-the-art approaches are based on successive linearizations: the nonlinear cost is linearized around the current estimate and a local update is computed by minimizing a quadratic approximation of the cost. Computing the local update requires solving a linear system, which is expensive for large problems.

Conjugate Gradient (CG) has been shown to be an effective linear solver for large-scale BA. The number of CG iterations can be reduced by *preconditioning*, or by using the *truncated Newton method*, which trades off accuracy of the solution for computational efficiency.

Recent work [2] provides the key insight that, in the CG method, the Schur complement trick can be applied without the explicit computation of the *reduced camera matrix*. This *implicit* representation is shown to be convenient (storage and computation-wise) with respect to *explicit* representations, in which a large (but sparse) square matrix has to be formed.

Contribution. In this paper, rather than proposing strategies to reduce the *number* of CG iterations, we propose an insight that reduces the complexity of *each* CG iteration. We adopt a factor graph perspective, and interpret BA in terms of inference over a factor graph (Table 1a). We show that the elimination of a single point induces a *factor* (i.e., a probabilistic constraint) over the cameras observing the point (Table 1b). The elimination of all points leads to the standard Schur complement, while in our approach we never build the reduced cameras system explicitly.

Reasoning in terms of factor graphs allows the solver to choose the best representation (i.e., implicit vs explicit) for each factor. A factor produced by the elimination of a single point provides a low-rank constraint on the cameras, and the use of an explicit representation is not efficient for those. However, we show that "grouping" factors corresponding to many points that are co-visible by the same set of cameras produces a single *grouped* factor, for which the explicit representation can be convenient (Table 1c): when a group of N points is visible in $M < N$ cameras, the *explicit* representation has a smaller computational cost in each conjugate gradient iteration. We call these groups of points "fragments" (Table 1d).

The grouping can be done in a grounded way: the problem is formally equivalent to well studied problems in data mining (e.g., *frequent items mining* [4]). The computational cost of grouping is negligible in BA.

In summary, our BA solver computes the fragments using data mining techniques and uses an explicit representation for the corresponding groups of factors, while the remaining factors are kept in implicit form.

Results. We tested our approach in the Bundle Adjustment in the Large benchmarking datasets [2]. Our method is implemented in C++ and released in [3]. We compare our technique against one using an implicit representation for all factors, and against a state-of-the-art solver (the *iterative Schur Solver*) available in the `Ceres` optimization suite [1].

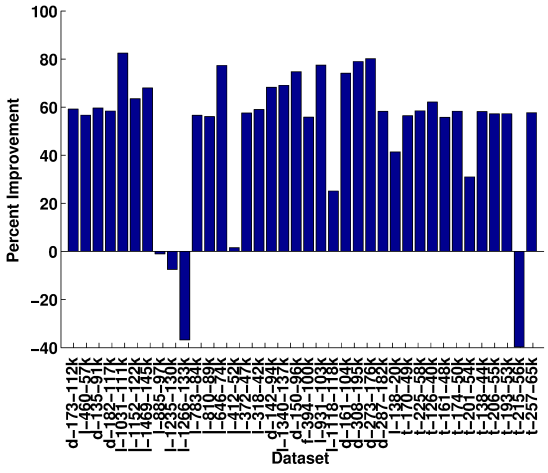

Figure 1: Total reduction in the optimization time, comparing the proposed approach against `Ceres`. The dataset *Dubrovnik* with 150 cameras and 95821 points is denoted with d-150-96k. Similar labels are used for *Ladybug*(l), *Trafalgar*(t), and *Final*(f).

Fig. 1 shows the reduction in the optimization time, comparing our approach against `Ceres`. Grouping leads to a time reduction of 50% (averaged across all datasets), with peaks reaching 80%; only in few cases `Ceres` was faster, due to early termination in CG iterations. The paper and the supplemental material include further comments and results, to show that this advantage is consistent across a large variety of tests.

[1] S. Agarwal, K. Mierle, and Others. Ceres solver, `https://code.google.com/p/ceres-solver/`.

[2] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, and R. Szeliski. Bundle adjustment in the large. In *European Conf. on Computer Vision (ECCV)*, pages 29–42, 2010.

[3] F. Dellaert et al. GTSAM: Georgia Tech Smoothing And Mapping, `https://borg.cc.gatech.edu`.

[4] J. Han, H. Cheng, D. Xin, and X. Yan. Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86, 2007.