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 Limited, Cambridge Research Laboratory, UK
[3] SRI International, Division of Information and Computing Sciences, USA
[4] Georgia Tech Research Institute, ATAS Laboratory, USA

In this document we show extra-results from our tests on the Bundle Adjustment in the Large benchmarking datasets [2]. We use acronyms for the datasets, e.g., 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). We compare 3 approaches: the approach proposed in this paper (`gCG`), an implementation in we use implicit Schur representation for all factors (`iCG`), and the *Iterative Schur Solver* (`cCG`), available in `Ceres` [1]. In the tests comparing `gCG` and `cCG` we use a *truncated Newton* method [2] as in `Ceres`.

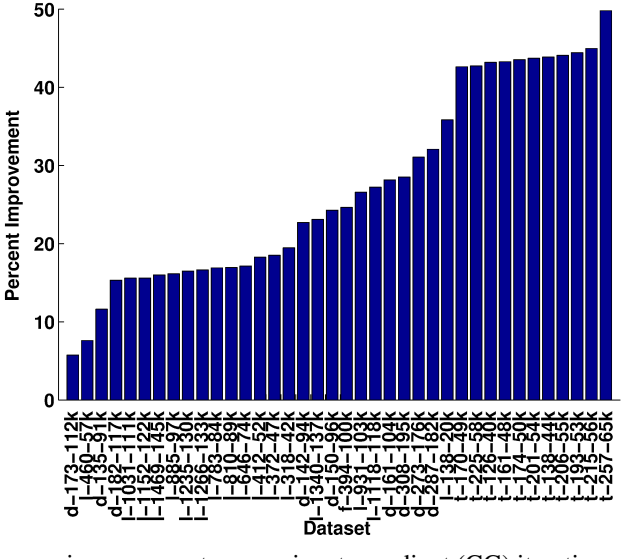

Figure 1: Average improvement per conjugate gradient (CG) iteration of `gCG` w.r.t. `iCG`. The improvement is computed as $\left(t_{\texttt{iCG}} - t_{\texttt{gCG}}\right)/t_{\texttt{iCG}}$ (%), where t_i is the average time for a CG iteration in technique i.

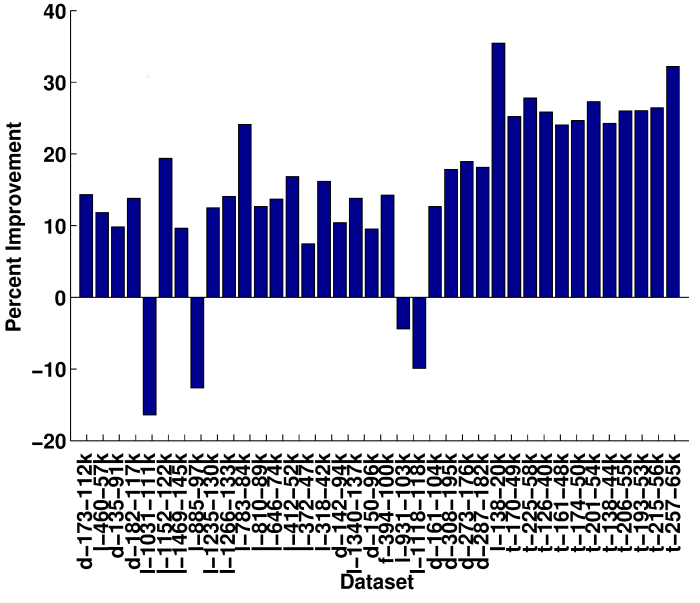

Figure 2: Total time reduction of `gCG` w.r.t. `iCG`. Time reduction is computed as $\left(T_{\texttt{iCG}} - T_{\texttt{gCG}}\right)/T_{\texttt{iCG}}$ (%), where T_i is the time to reduce the objective by 90%.

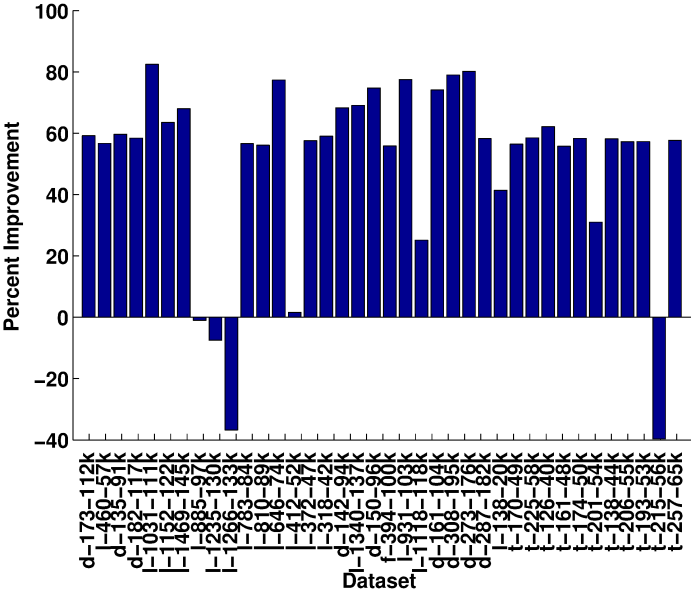

Figure 3: Total time reduction of `gCG` w.r.t. `cCG`. Both use block-Jacobi preconditioning and inexact Newton steps. Time reduction is computed as $\left(T_{\texttt{cCG}} - T_{\texttt{gCG}}\right)/T_{\texttt{cCG}}$ (%), where T_i is the time to reduce the objective by 90%, for technique i. The few cases in which `Ceres` has better performance are connected to slight differences in the LM policy, that allow `cCG` to apply more effective steps. If we look at the time per LM iteration (Fig. 4), the advantage of grouping is even clearer.

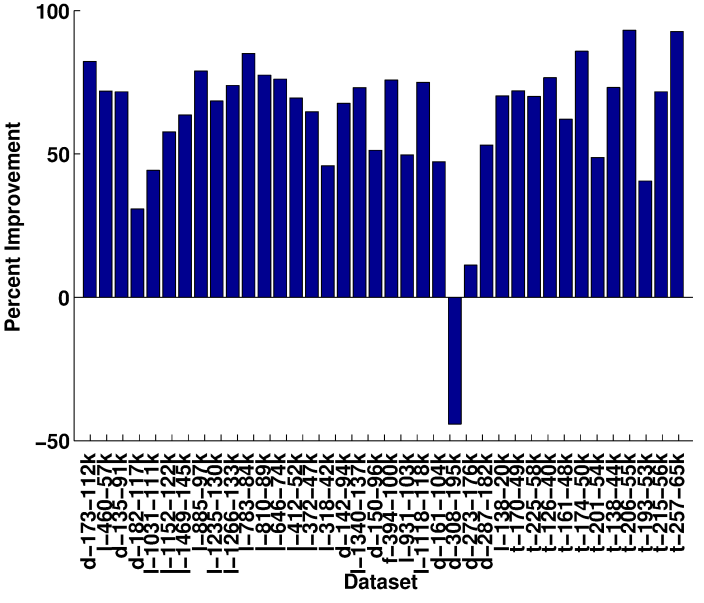

Figure 4: Reduction in the average LM iteration time of gCG w.r.t. cCG. Time reduction is computed as $\left(T^{\text{LM}}_{\text{cCG}} - T^{LM}_{\text{gCG}}\right)/T^{LM}_{\text{cCG}}$ (%), where T_i^{LM} is the average time for a single LM iteration, for technique i. The only negative peak, for which gCG is slower than cCG corresponds to the dataset d-308-195k, in which `Ceres` terminates earlier (details in Fig. 5).

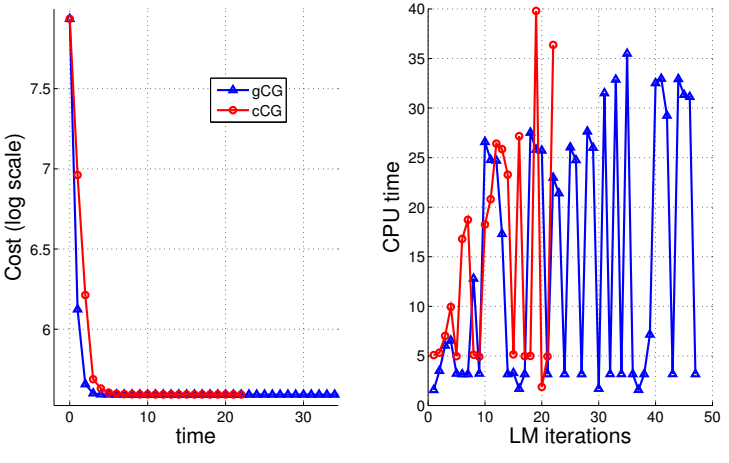

Figure 5: (a) Cost (in log scale) versus time (in seconds) for gCG (solid blue line) and cCG (solid red line). The longer tail in gCG depends on the termination condition for LM, which can be slightly different for the two techniques. (b) CPU time required to complete each LM iteration. In this experiment, although gCG has better performance (see faster convergence in (a)), it performs more iterations, with few expensive iterations at the end. Note that cCG terminates around iteration 22 and before that point, in figure (b), one can observe that the cost per iteration is smaller for gCG, as shown in all other datasets of Fig. 4.

References

[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.