Bundle Adjustment in the Large

Sameer Agarwal
Google

Noah Snavely
Cornell University

Steven M. Seitz
Google & University of Washington

Richard Szeliski
Microsoft Research

Current Bundle adjustment algorithms do not scale beyond 1-2K images.

Our Algorithm

1. 14k images, 4.5M points in less than an hour.
2. Inexact step Levenberg Marquardt algorithm.
3. Predictable and minimal memory usage.
4. No need for high performance BLAS/LAPACK.
4. Easily parallelizable (shared and distributed memory)
5. Simple preconditioners give state of the art performance.

Exact Step Levenberg Marquardt

Until convergence
$\min_{\Delta x_k} \|J(x_k)\Delta x_k + f(x_k)\|^2 + \mu\|D(x_k)\Delta x_k\|^2$
if $\|f(x_k + \Delta x_k)\|^2 < \|J(x_k)\Delta x_k + f(x_k)\|^2$
$x_{k+1} \leftarrow x_k + \Delta x_k$
$\mu \leftarrow \mu/2$
else
$x_{k+1} \leftarrow x_k$
$\mu \leftarrow 2 * \mu$
$k \leftarrow k + 1$

Calculating the LM Step

Normal Equations

$$\left[J^\top(x)J(x) + \mu D(x)^\top D(x)\right]\Delta x = -J^\top(x)f(x)$$

Hessian Approximation $H_\mu \Delta x = -g$

Cameras / Points
$$\begin{bmatrix} B & E \\ E^\top & C \end{bmatrix}\begin{bmatrix}\Delta y \\ \Delta z\end{bmatrix} = \begin{bmatrix} v \\ w\end{bmatrix}$$

Schur Complement $\left[B - EC^{-1}E^\top\right]\Delta y = v - EC^{-1}w$
$\Delta z = C^{-1}(w - E^\top \Delta y)$

$S = B - EC^{-1}E^\top$

1. Expensive to compute and store
2. Extremely expensive to factorize

Inexact Step Levenberg Marquardt

Replace the exact solution to

$$\min_{\Delta x_k} \|J(x_k)\Delta x_k + f(x_k)\|^2 + \mu\|D(x_k)\Delta x_k\|^2$$

with an approximate solution satisfying

$$\|H_\mu(x_k)\Delta x_k + g(x_k)\| \le \eta_k \|g(x_k)\|$$

↑ Forcing sequence, controls the quality of LM step

Calculating the Inexact step

Use Preconditioned Conjugate Gradients.

But which linear system ?

$$H_\mu \Delta x = -g$$

Large linear system. Easy to evaluate matrix-vector products.

Or,

$$\left[B - EC^{-1}E^\top\right]\Delta y = v - EC^{-1}w$$

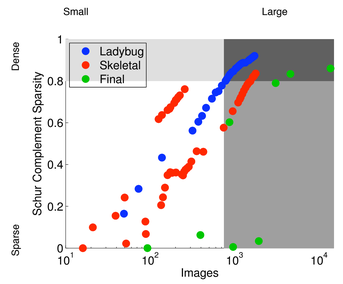
Much smaller linear system. Expensive to compute and store, but

$$x_2 = E^\top \Delta y,\ x_3 = C^{-1}x_2,\ x_4 = Ex_3$$
$$x_5 = B\Delta y$$
$$[B - EC^{-1}E^\top]\Delta y = x_5 - x_4$$

i.e., We can run PCG *implicitly* on the reduced camera matrix at the same cost as the Normal equations!

Lemma (Saad 2003): CG with on the reduced camera matrix with a preconditioner P is the same as CG on the normal equations with the SSOR preconditioner:

$$M(P) = \begin{bmatrix} P & E \\ 0 & C\end{bmatrix}\begin{bmatrix} P^{-1} & 0 \\ 0 & C^{-1}\end{bmatrix}\begin{bmatrix} P & \\ E^\top & C\end{bmatrix}$$

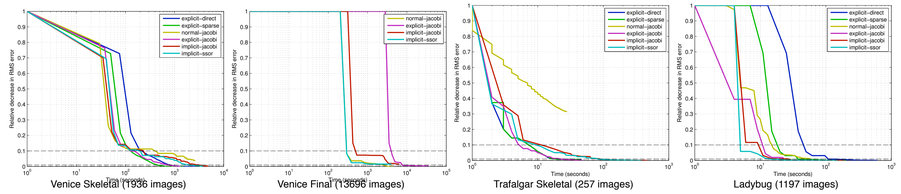


Dataset

1. Ladybug: Images captured from a vehicle driving down a street.
2. Skeletal: Sparse incremental reconstruction of Flickr images.
3. Final: Reconstructions from the final stage of skeletal sets algorithm on Flickr images.

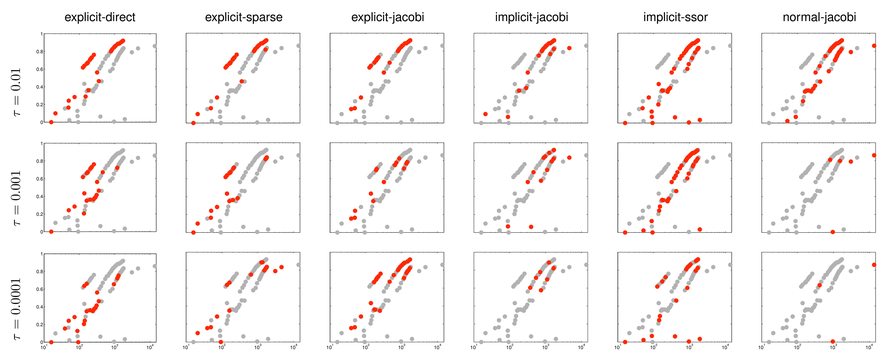
Experiments

Run time



1. The first few steps don't need to be very accurate, initial decrease can be quite fast.
2. The quality of preconditioner decides performance in later iterations.
3. Simpler preconditioners give crude solutions fast and then stall.
4. Small problems: SBA/direct-factorization.
5. Large problems: PCG with SSOR preconditioning.

Solution Accuracy



Code and data on our website `http://grail.cs.washington.edu/projects/bal`