GPGPU solutions of the Linear Least Squares Problem for Simultaneous Localization and Mapping

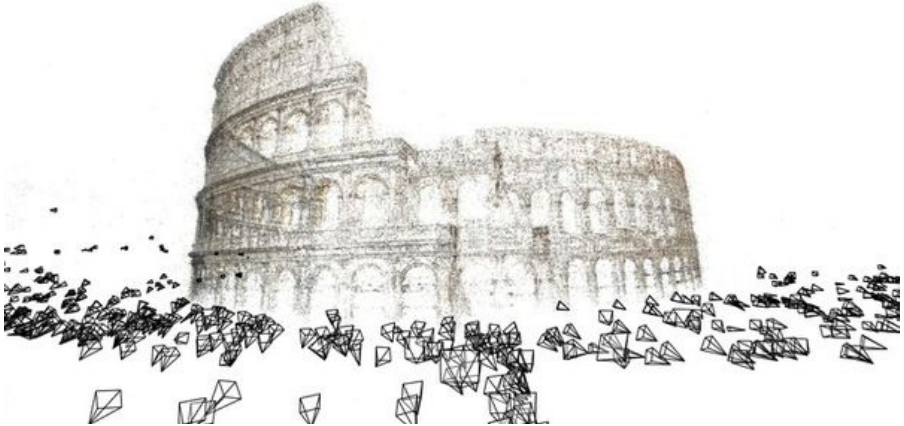
Zhaoyang Lv, Samarth Mishra, Stefan Stojanov

Introduction

Solving nonlinear least-square problems are widely used in Robotics and Computer Vision:

- Simultaneous Localization and Mapping
- Structure from Motion
- Object Tracking
- Etc.

All of the least-square solvers heavily rely on efficient and accurate matrix factorization of the linearized problem.



[Building Rome in a Day, ICCV, 2009]

Nonlinear Least-square Problem Revisited

Consider a SLAM problem with only pairwise constraint:

$$X = \underset{X}{\operatorname{argmin}} \sum_{i=1}^{N} \|f(x_{i-1}, x_i)\|_2$$

Using iterative method, e.g. Newton's method, we update

$$X^{k+1} = X^k + \delta X$$

$$\mathbf{J}\delta X = \mathbf{r}$$

Jacobian Matrix of the pairwise constraint

Residual at each iteration

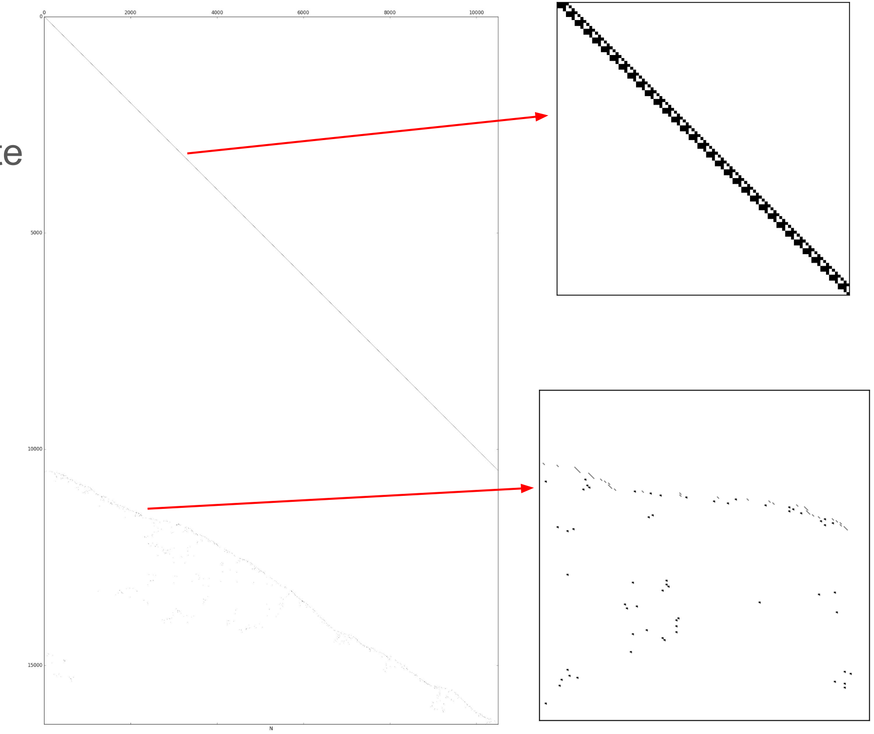
We can solve the following the least square problem:

$$\delta X = \operatorname{argmin} \|A\delta X - b\|_2$$

1. QR Factorization $A = \mathbf{J}, b = \mathbf{r},$
2. Cholesky Factorization $A = \mathbf{J}^\top \mathbf{J}$ and $b = \mathbf{J}\mathbf{r}$

- The LHS matrix is highly sparse.
- In general, M > N. More measurements than unknowns.

An example of Jacobian matrix J (16362 x 10500)



Our Implementation

We focus on solving $\delta X = \operatorname{argmin} \|A\delta X - b\|_2$

We implement and evaluate 6 different (CPU and GPU) benchmark algorithms that solve the LS problem on sparse matrices. These use the Compressed Sparse Row/Column (CSR/CSC) format to represent the matrix.

Example :

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 0 \end{pmatrix}$$

CSR(A) = (M, IM, JM)

M = [5 8 3 6] — All non-zero elements

IM = [0 0 2 3 4] — Index into M, first non-zero in row

JM = [0 1 2 1] — Column indices for all elements in M

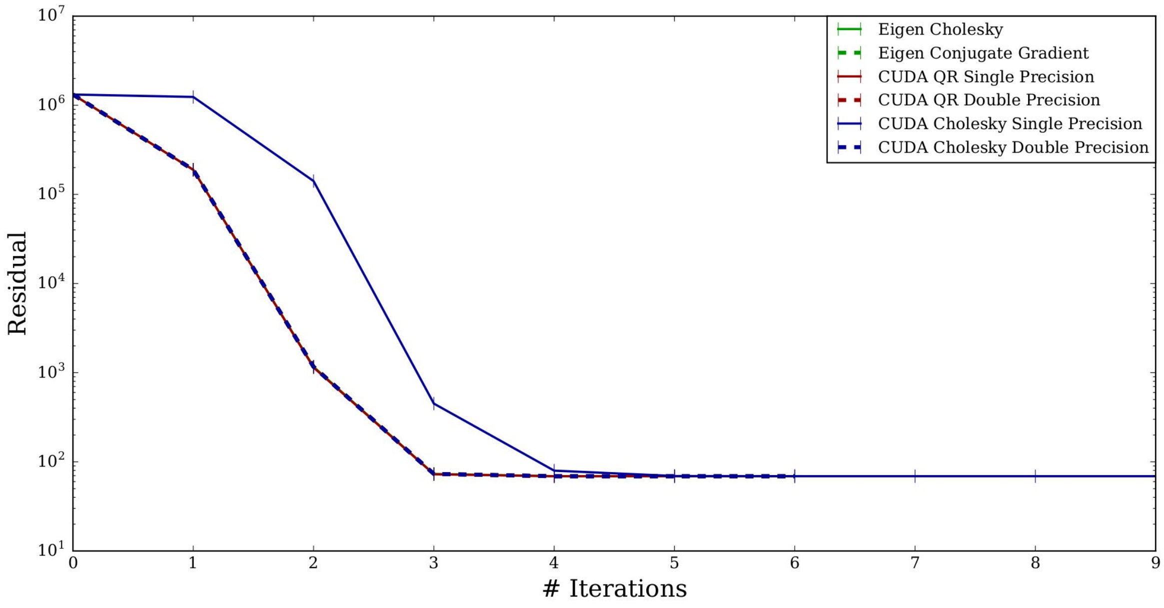
Benchmark Algorithms:

- **CPU** Sparse Householder QR decomposition [Eigen C++ library]
- **CPU** Sparse Cholesky Decomposition for solving normal equations [Eigen C++ library]
- **CPU** Sparse Conjugate Gradient Method with Incomplete Cholesky Preconditioning . [Eigen C++ library]
- **GPU** Sparse Householder QR decomposition [cuSolverSP library] (Both single and double precision)
- **GPU** Sparse Cholesky Decomposition [cuSolverSP library] (Both single and double precision)

CSR Example Source : https://en.wikipedia.org/wiki/Sparse_matrix

GPGPU v.s. CPU Least-square Solver Comparison

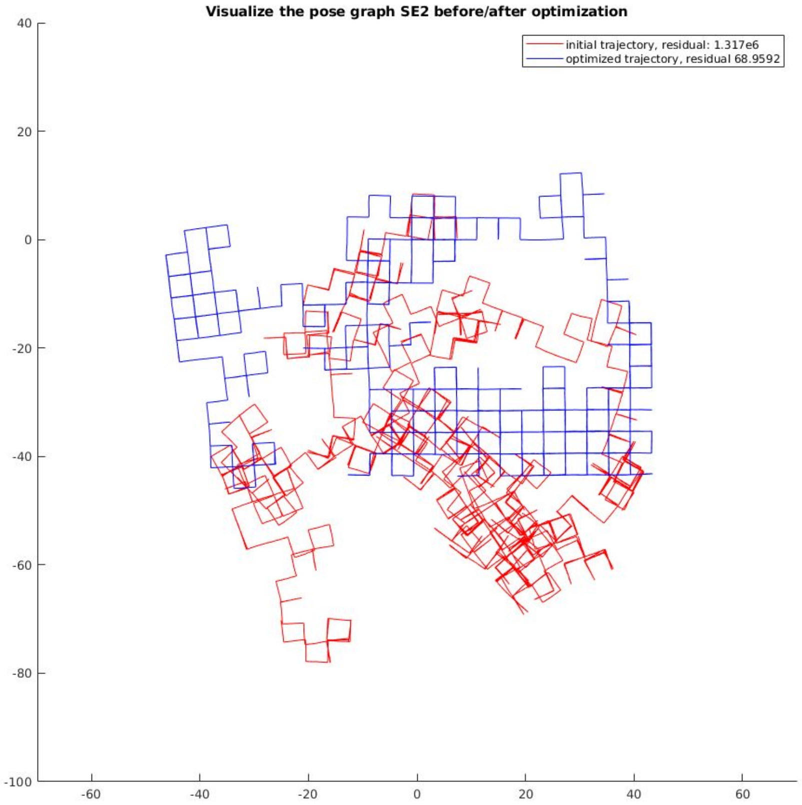
Method	Eigen CG	Eigen Cholesky	cuCholesky SP	cuCholesky DP	cuQR SP	cuQR DP
Time	6.6 min	49 s	3481 ms	2637 ms	40 s	46.4 s
# Iterations	6	6	22	6	6	6



- **SP:** Single Precision
- **DP:** Double Precision
- **Iterations:** number of iterations for Newton's method to converge.

Conclusion: Cuda Cholesky factorization in double precision delivers the fastest solution without any loss in accuracy.

Visualization of Optimization Results



Ground Truth