# An Efficient Solution to Structured Optimization Problems using Recursive Matrices 

Darius Rückert ${ }^{1}$ and Marc Stamminger ${ }^{1}$
${ }^{1}$ University of Erlangen-Nuremberg, Germany

## Motivation



## > 50 \% of the code is for <br> - Sparse Block Matrices <br> - Linear Solvers

## Can we get rid of that?

https://github.com/ceres-solver/ceres-solver

## Overview



# Eigen Extension 


https://eigen.tuxfamily.org

## What is a Recursive Matrix?

## Definition

A recursive matrix is a rectangular array of numbers or recursive matrices.

$$
\left.\left[\begin{array}{ccc}
0 & -1 & 1 \\
1 & 0 & -1 \\
-1 & 1 & 0
\end{array}\right] \quad\left[\left(\begin{array}{ll}
1 & 0 \\
0 & 1
\end{array}\right) \quad\left(\begin{array}{ll}
0 & 0 \\
0 & 0
\end{array}\right)\right]\left[\begin{array}{lllll}
\left(\begin{array}{ll}
1 & 2 \\
3 & 4
\end{array}\right) & & & \left(\begin{array}{ll}
5 & 6 \\
7 & 8
\end{array}\right) & \\
\\
& & \left(\begin{array}{ll}
1 & 1 \\
1 & 1
\end{array}\right) & & \\
0 & 0 \\
0 & 0
\end{array}\right) \quad\left(\begin{array}{ll}
1 & 0 \\
0 & 1
\end{array}\right)\right]\left[\begin{array}{llll} 
\\
\left(\begin{array}{lll}
5 & 0 \\
0 & 5
\end{array}\right) & & & \\
& \ddots & & \\
& & & \\
& & & \\
& & & \left(\begin{array}{ll}
4 & 1 \\
2 & 3
\end{array}\right)
\end{array}\right]
$$

## Recursive Matrix Operations

- Multiplication
- Addition
- Transposition

$\left[\begin{array}{ll}\left(\begin{array}{ll}1 & 2 \\ 5 & 6\end{array}\right) & \left(\begin{array}{ll}3 & 4 \\ 7 & 8\end{array}\right) \\ \left(\begin{array}{cc}9 & 10 \\ 13 & 14\end{array}\right) & \left(\begin{array}{ll}11 & 12 \\ 15 & 16\end{array}\right)\end{array}\right]^{T}=\left[\begin{array}{ll}\left(\begin{array}{ll}1 & 2 \\ 5 & 6\end{array}\right)^{T} & \left(\begin{array}{cc}9 & 10 \\ 13 & 14\end{array}\right)^{T} \\ \left(\begin{array}{ll}3 & 4 \\ 7 & 8\end{array}\right)^{T} & \left(\begin{array}{ll}11 & 12 \\ 15 & 16\end{array}\right)^{T}\end{array}\right]=\left[\begin{array}{ll}\left(\begin{array}{ll}1 & 5 \\ 2 & 6\end{array}\right) & \left(\begin{array}{ll}9 & 13 \\ 10 & 14\end{array}\right) \\ \left(\begin{array}{ll}3 & 7 \\ 4 & 8\end{array}\right) & \left(\begin{array}{ll}11 & 15 \\ 12 & 16\end{array}\right)\end{array}\right]$


## Recursive Matrices in Eigen

- Scalar Matrix

$$
\left[\begin{array}{ccc}
0 & -1 & 1 \\
1 & 0 & -1 \\
-1 & 1 & 0
\end{array}\right]
$$

Matrix<float,3,3>

- Dense Block Matrix
Matrix<Matrix<float, 2,2>2,2>

$$
\left[\begin{array}{ll}
\left(\begin{array}{ll}
1 & 0 \\
0 & 1
\end{array}\right) & \left(\begin{array}{ll}
0 & 0 \\
0 & 0
\end{array}\right) \\
\left(\begin{array}{ll}
0 & 0 \\
0 & 0
\end{array}\right) & \left(\begin{array}{ll}
1 & 0 \\
0 & 1
\end{array}\right)
\end{array}\right]
$$

- Sparse Block Matrix
SparseMatrix<Matrix<float,2,2>>
- Recursive Operations in Eigen
> Not working without our extension (see Paper)



## Least-Squares Optimization

 $\underset{p, q, \ldots}{\arg \min } \sum_{\Downarrow} R(p, q, \ldots)^{2}$
## Adjacency/Hessian Matrix



## As Rigid as Possible (ARAP)

 $\underset{x}{\arg \min } \sum R\left(x_{i}, x_{j}\right)^{2}$


Recursive Matrix Type:

## Sparsematrix< Matrix<float,3,3> > H;

## Bundle Adjustment

## $\underset{x, p}{\arg \min } \sum R\left(x_{i}, p_{j}\right)^{2}$





Po
$3 \times 6$ Block

Recursive Matrix Type:
sparsematrix<
Matrix<float,-1,-1> > H;

## Slow! D:

## Mixed Matrix

## Definition

A mixed matrix is a rectangular array, where each element can be of a different type.

$$
\left[\begin{array}{ccc}
{\left[\begin{array}{ccc}
0 & -1 & 1 \\
1 & 0 & -1 \\
-1 & 1 & 0
\end{array}\right]} & 42 \\
31+5 i & \text { "Hello" }
\end{array}\right] \begin{aligned}
& \text { MixedMatrix22< } \\
& \text { Matrix<float, 3, 3> } \\
& \text { float, } \\
& \text { complex<float> } \\
& \text { string } \\
& >A ;
\end{aligned}
$$

## Recursive Bundle Adjustment



MixedMatrix22<
Diagonalmatrix<Matrix<f1oat,3,3>>, SparseMatrix<Matrix<float,3,6>>, Sparsematrix<Matrix<float,6,3>>, DiagonalMatrix<Matrix<float,6,6>> > H;

## How do we solve

$$
H \Delta x=b \text { ? }
$$

## Recursive Linear Solvers

"Normal" recursive matrices

- Recursive CG, Recursive LDLT,...
> Straight forward (see paper)
Mixed recursive matrices
- General solvers (CG)
- Partially specialized solvers


## Partially Specialized Solvers



Diagonal
Arbitrary
Arbitrary
Arbitrary
Implementation

1. Invert Diagonal Matrix
2. Compute Schur Complement
3. Solve Reduced System (Recursive Call!)
4. Compute Solution for Initial System

Mixed 2x2 Matrix
$>$ Better than general solver

## Template Matching



## Results

## Structured Optimization

## ARAP - Speedup

## Bundle Adjustment - Speedup



## Results

## Sparse Block Matrix Multiplication

Sparse Block Matrix-Vector

## Sparse Block Matrix-Matrix

Speedup
2.5


Speedup

$>$ Clang 8.0-1 Thread on i7-7700K, SSE+AVX

## Application

Camera Tracking (SLAM)

- Local/Global BA
- Pose Refinement
- Pose Graph Optimization

$>1.17 \mathrm{~ms} /$ frame ( $\sim 850 \mathrm{FPS})^{*}$
*4 Threads on i7-8850H


## Any Questions?


(? https://github.com/darglein/EigenRecursive
$\Delta$ darius.rueckert@fau.de

