Improving the Performance of Local Bundle Adjustment for Visual-Inertial SLAM with Efficient Use of GPU Resources

Shishir Gopinath [1] Karthik Dantu [2] Steven Y. Ko [1]

[1]Simon Fraser University [2]University at Buffalo

Introduction

Bundle adjustment is a process used in visual-inertial SLAM to correct the inconsistencies in the relationships between keyframes, map points, and inertial variables. Reprojection and inertial residuals can be represented with **non-linear constraints**.

- Bundle adjustment performs **iterative non-linear least squares optimization** to refine parameters and reduce inconsistency by minimizing the total error across these constraints.
- The normal equations $H\Delta x = -b$ (or damped variants) are constructed for the error minimization. Each iteration tries to find a new **parameter update** Δx to reduce the error.
- **Local bundle adjustment** enforces consistency frequently for a subset of the map, but may still be expensive for resource-constrained systems.

The Schur Complement Method

To compute Δx, optimization libraries such as g2o [3] partition the system in terms of **pose** and **landmark** variables.

$$\begin{bmatrix} H_{pp} & H_{pl} \\ H_{pl}^T & H_{ll} \end{bmatrix} \begin{bmatrix} \Delta x_p \\ \Delta x_l \end{bmatrix} = - \begin{bmatrix} b_p \\ b_l \end{bmatrix} \tag{1}$$

The equations can be solved more efficiently using the **Schur complement method**, which computes a **reduced system** [3].

$$H_{Schur} = H_{pp} - H_{pl} H_{ll}^{-1} H_{pl}^T \tag{2}$$

$$b_{Schur} = -b_p + H_{pl} H_{ll}^{-1} b_l \tag{3}$$

$$H_{Schur} \Delta x_p = b_{Schur} \tag{4}$$

$$\Delta x_l = H_{ll}^{-1}(-b_l - H_{pl}^T \Delta x_p) \tag{5}$$

This is performed by a software component called a **block solver**.

Local Bundle Adjustment Workloads

We develop techniques based on two observations about the workload generated by local visual-inertial bundle adjustment.

1. Computing the **Schur complement** has the largest overhead when solving for the parameter update.
2. The calculations involve **small to medium-sized sparse matrices**.

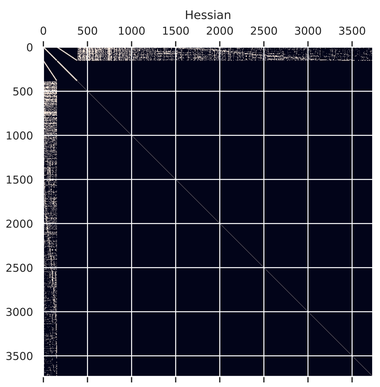

Figure 1. The sparse block structure of the Hessian matrix (H) for visual-inertial bundle adjustment.

Our Approach

We replace the block solver in g2o [3] with our own. It computes the reduced system (Eq. 2 and 3) and the landmark update (Eq. 5) on a GPU using Vulkan **compute shaders**.

For this, we use the following strategies described below.

1. Sparse Block Matrix Representation

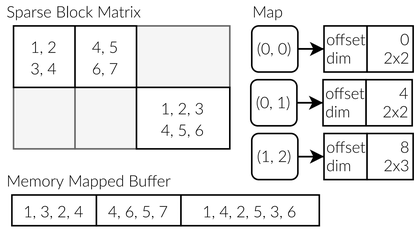

Figure 2. Sparse block matrix representation. Block dimensions vary by the sizes of the parameters. The submatrices of H are constructed on the CPU as before, directly in the buffer.

2. Work Queues

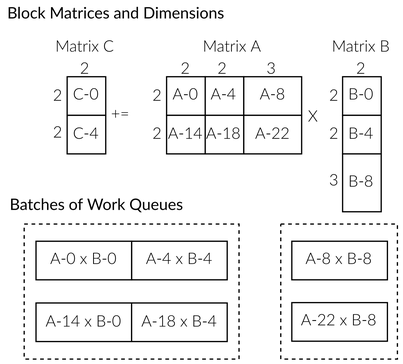

Figure 3. Work queues coordinate block-by-block multiplications on a GPU. They are batched together if they multiply blocks of the same dimensions, allowing the operation to complete more quickly.

3. **Setup Pipelining**: Shaders are set up in parallel while building the full linear system during the first optimization iteration, allowing the Schur complement to be computed as soon as it is ready.
4. **Memory Suballocation**: Memory for each GPU buffer is allocated from larger reserved blocks of memory, avoiding performance degradation for longer sequences [1].

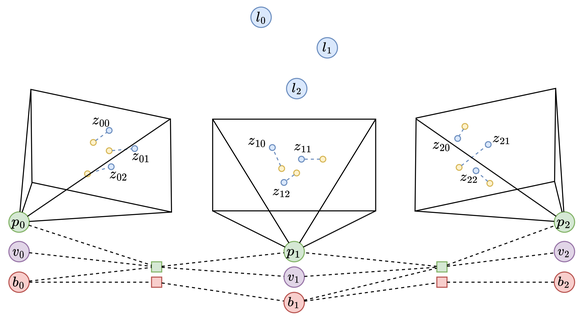

Figure 4. A visual-inertial bundle adjustment problem (simplified).

Evaluation

We process EuRoC and TUM-VI datasets with stereo-inertial ORB-SLAM3 [2] on two machines. The **local-inertial bundle adjustment** execution time and GPU buffer memory usage are measured.

Specs	Desktop	Jetson Xavier NX
CPU	8-core AMD 5800X @ 3.8 GHz	6-core ARM Carmel @ 1.4 GHz 15W
GPU	8704-core NVIDIA RTX 3080	384-core NVIDIA Volta
RAM	32 GB	8 GB

Table 1. Evaluation machine specifications.

Overall Performance

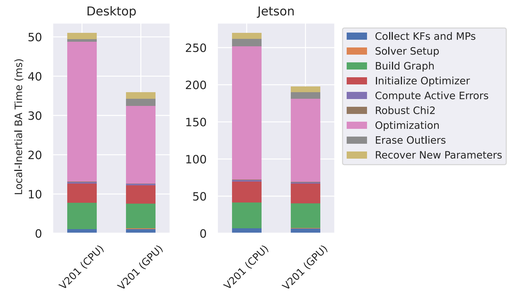

Figure 5. Performance improvement on EuRoC V201.

Block Solver Performance

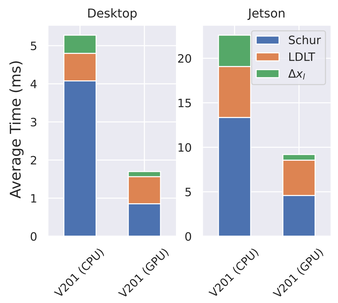

Figure 6. For the reduced system calculation (Schur), we see speedups of $5.08\times$ on the desktop and $2.87\times$ on the Jetson for EuRoC V201.

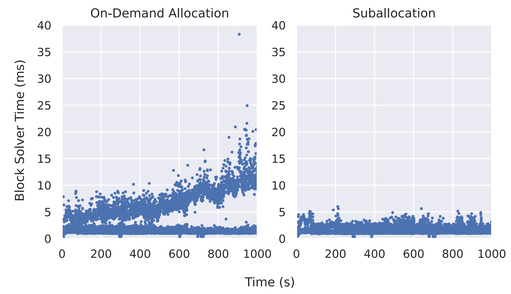

Figure 7. Suballocation avoids performance degradation over time. This is especially noticeable for longer sequences such as outdoors6.

GPU Buffer Memory Usage

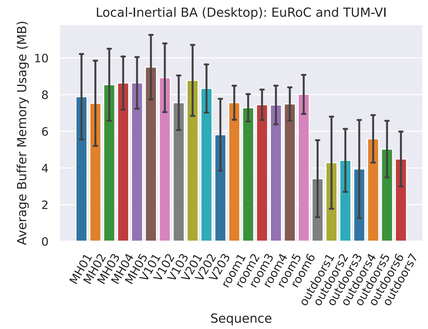

Figure 8. Total GPU buffer memory usage on the desktop. *Host-cached* and *device-local* allocations are made for each matrix.

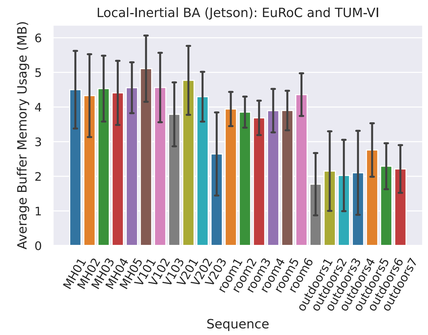

Figure 9. Total GPU buffer memory usage on the Jetson. The usage is lower because it uses a single *host-coherent* allocation for each matrix.

Conclusion

We can improve the performance of bundle adjustment for visual-inertial SLAM using a GPU even for small local workloads. Our methods decrease the average execution time of local-inertial bundle adjustment by up to **33.79%** ($1.51\times$ **speedup**) on a desktop machine and up to **26.68%** ($1.36\times$ **speedup**) on a Jetson Xavier NX board. Our implementation is open-source and available at the code and link below.

`https://github.com/sfu-rsl/gpu-block-solver`

References

[1] AMD. Vulkan memory allocator, 2022.

[2] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37(6):1874–1890, 2021.

[3] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. g2o: A general framework for graph optimization. In *2011 IEEE International Conference on Robotics and Automation*, pages 3607–3613, 2011.