How optimization is implemented in gs-rs

Samuel Valenzuela[1], **Daniel Pape**[1]

[1] *TNG Technology Consulting GmbH, Unterföhring, Germany*

May 19, 2020

gs-rs is a Rust framework for the optimization of non-linear least squares problems embeddable as a (hyper)graph. It is suitable for the optimization of an error function with respect to a set of parameters as in SLAM (simultaneous localization and mapping) or BA (bundle adjustment).

1 Optimization Algorithm

1.1 Background Literature

The optimization algorithm used by **gs-rs** is based on that which parts of **g2o** uses. This paper should suffice to understand the optimization's implementation in **gs-rs**. The following papers by the developers of **g2o** are recommended if a deeper understanding of the theory behind the algorithm is of interest.

- *g2o: A General Framework for Graph Optimization, Kümmerle et al.* [1]: This paper documents the derivation of the algorithm's structure.
- *A Tutorial on Graph-Based SLAM, Grisetti et al.* [2]: This paper contains additional comments on the calculations in 2D and 3D. Here it is presented how the least squares optimization works on a manifold.

1.2 Iteration Steps

Given a specific number of iterations n and the initial guess $x_i^{(0)}$ for each variable, the optimizer algorithm will repeat the following steps n times:

1. Calculate H and b by looping through all factors and updating its variables' entries in H and b.
2. Calculate Δx, the vector containing data about how much each current variable guess $x_i^{(k)}$ should be updated in this step, by solving the linear system

$$H\Delta x = -b. \tag{1}$$

3. Update the guesses for each variable x_i with

$$x_i^{(k+1)} = x_i^{(k)} + \Delta x_i. \tag{2}$$

How the parts of H and b are calculated depends on the exact factor type. In the following sections, the calculation is described for all 2D and 3D factors supported by **gs-rs**.

2 Optimization in 2D

Given the current guess $x_i^{(k)}$ and, if it is a factor involving two variables, $x_j^{(k)}$, as well as the measurement x_{ij}, the Jacobian matrix $J_{ij}^{(k)}$ and error vector $e_{ij}^{(k)}$ can be computed. With these as well as the information matrix Ω_{ij} corresponding to x_{ij}, the parts H_{factor_ij} and b_{factor_ij} can be calculated as follows:

$$E = mc^2 \tag{3}$$

If the factor involves only one variable, the submatrix H_{ii} and subvector b_i are incremented by H_{factor_ij} and b_{factor_ij}, respectively.

If the factor involves two variables, the submatrix H_ii will be incremented by the top-left submatrix of H_{ij} and H_j by the bottom-right submatrix of H_{ij}. The submatrices with bottom-left and top-right submatrices, respectively.

The calculation path for $J_{ij}^{(k)}$ and $e_{ij}^{(k)}$ depends on the specific factor. The individual calculation paths are presented in the following sections.

2.1 Position2D

2.2 Odometry2D

2.3 Observation2D

3 Optimization in 3D

3.1 Position3D

3.2 Odometry3D

3.3 Observation3D

References

[1] Rainer Kümmerle et al. “g2o: A general framework for graph optimization”. In: *2011 IEEE International Conference on Robotics and Automation*. IEEE. 2011, pp. 3607–3613.

[2] Giorgio Grisetti et al. “A tutorial on graph-based SLAM”. In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43.