A Spanning Tree-Based Multi-Resolution Approach for Pose-Graph Optimization

A Spanning Tree-Based Multi-resolution Approach for Pose-graph Optimization

Yuichi Tazaki[1]

***Abstract*—This paper proposes a computationally efficient method for pose-graph optimization that makes use of a multi-resolution representation of pose-graph transformation constructed on a spanning tree. It is shown that the proposed spanning tree-based hierarchy has a number of advantages over the previously known serial chain-based hierarchy in terms of preservation of sparsity and compatibility with parallel computation. It is demonstrated in numerical experiments using several public datasets that the proposed method outperforms a state-of-the-art solver for large-scale datasets.**

***Index Terms*—Mapping, SLAM**

I. Introduction

THE problem of estimating the robot's trajectory and generating a map of the environment simultaneously from a set of measurement data is called the Simultaneous Localization and Mapping (SLAM) problem. SLAM problems have two major categories, one is incremental SLAM, which considers partial and incremental update of a map, and the other is full SLAM, which considers generation of an entire map from a batch of observation data. A full SLAM problem of a posegraph is called a posegraph SLAM problem or posegraph optimization (PGO), and it has been a subject of active research in past decades. A posegraph SLAM is formulated as a nonlinear least squares problem, and it is often solved by using iterative methods such as Gauss-Newton and Levenberg-Marquardt methods. Each iteration minimizes a quadratic function whose coefficient matrix is sparse and positive symmetric reflecting the topology of the posegraph. Historically, Lu et al. was one of the earliest to propose a least squares formulation of posegraph SLAM [1]. Dellaert et al. proposed to utilize sparse linear algebra to solve large-scale posegraph SLAM problems efficiently [2]. Later, extension to Incremental SLAM has been proposed in [3][4], and a distributed version for multi-robot SLAM has been studied in [5][6]. Kümmerle et al. proposed a posegraph SLAM solver called g2o, which is considered to be one of the fastest implementation of posegraph SLAM solver to date [8]. Internally, g2o uses CHOLMOD, a state-of-the-art sparse direct solver for symmetric linear system of equations. Rosen et al. proposed SE-Sync, a posegraph optimization solver based on an alternative formulation of posegraph optimization and its efficient solution algorithm based on convex relaxation [15]. Its extension to distributed SLAM has been proposed in [16]. Bai et al. proposed posegraph optimization in cycle space, motivated by the intuition that most practical posegraphs with a large number of nodes may have much lower cycle space dimensions [17]. Apart from computational efficiency, improving the robustness against spurious links included in the posegraph is also considered to be an important research topic [9][10][11]. Some other studies focus on removing redundant nodes and links for reducing the size of the posegraph without losing essential information [12][13][14].

Although direct sparse solvers proved to be quite computationally efficient for posegraphs with moderate size, the theoretical complexity of posegraph optimization has higher order than linear complexity. The scale of environmental maps used for mobile robots, on the other hand, is expected to grow steadily in the near future. This means that further improvement of computational cost is required. One promising approach for reducing the size of posegraph optimization is hierarchization. Generally speaking, by transforming a single large-scale posegraph optimization problem into a hierarchy of smaller problems, the overall computational cost can be reduced dramatically. Frese et al. was one of the earliest to apply multi-grid methods to posegraph optimization [18]. Grisetti et al. proposed a method that constructs a hierarchy of posegraphs by grouping neighboring nodes [19]. One shortcoming of this method is that the grouping criterion and the setting of thresholds largely depend on heuristics. Anderson et al. proposed a multi-resolution method based on the representation of posegraphs by wavelet transform [20]. Guadagnino et al. proposed hierarchical initialization of posegraphs, in which an initial pose estimate is obtained from a coarse-grained posegraph.

Our study is strongly inspired by the earlier multi-resolution (MR) approaches [18][20]. In [18], coarser posegraphs are constructed by downsampling of nodes, while coarse-to-fine transformation was defined based on interpolation of poses. In [20], wavelet function was used, but the essential idea is similar to [18]. One drawback of this interpolation-based hierarchy is that the Hessians of higher (coarser) resolution levels tend to become dense, and consequently this type of MR does not contribute to the reduction of computation cost as much as one might expect. Moreover, existing studies on MR methods are nearly (or more than) a decade old, and quantitative comparison with sparsity-based methods which appeared later has not been reported.

This paper proposes a method for posegraph optimization that is based on a novel spanning tree-based multi-resolution hierarchy. The proposed MR formulation defines a coarse-to-fine transformation in such a way that the pose of a node that belongs to the lower (finer) level is defined by extrapolating the pose of a single supernode that belongs

Manuscript received: February, 20, 2022; Revised May, 6, 2022; Accepted June, 6, 2022.

This paper was recommended for publication by Editor Javier Civera upon evaluation of the Associate Editor and Reviewers ' comments. This work was not supported by any organization.

[1]Yuichi Tazaki is with Kobe University tazaki@mech.kobe-u.ac.jp

Digital Object Identifier (DOI): see top of this page.

to a higher (coarser) level. Compared to the interpolation-based transformation, this new formulation can reduce the densification of Hessians in higher levels, and as a result, the linear equation of each level can be solved more efficiently by sparse direct methods. Moreover, the spanning tree-based hierarchy possesses a computationally favorable characteristic that enables parallel computation of disjoint sub-equations in the same resolution level. Using open datasets, the proposed method is compared with g2o and SE-Sync in terms of convergence speed and computation time. It has been shown that the proposed method outperforms other methods when applied to large-scale datasets. An implementation of the proposed method is made publicly available for evaluation[1].

The rest of this paper is organized as follows. In Section II, the basic formulation of posegraph optimization is briefly reviewed. In Section III, a posegraph optimization method based on a novel spanning tree-based multi-resolution parametrization is presented. In Section IV, comparison results of the proposed method and the g2o solver are reported. A brief summary and concluding remarks are given in Section V.

II. Formulation of Pose-graph optimization

A posegraph consists of nodes and directed links where each node represents a pose in the configuration space (either 2D or 3D) and each link possesses desired relative pose and error covariance between two nodes connected by that link. The set of nodes is denoted by $\mathcal{N}$ and the set of links is denoted by $\mathcal{L}$. The source and destination nodes of a link l are denoted by n_l and n'_l, respectively. The pose of each node is expressed by an element of either SE(2) or SE(3), depending on whether we consider 2D or 3D posegraphs, respectively. In the following discussion, we assume a 3D posegraph, but the proposed method is applicable to 2D posegraphs with minor modification.

Let us denote the pose of a node n by $x_n = (p_n, q_n)$, where $p_n \in \mathbb{R}^3$ is the translational component and $q_n \in \mathcal{Q}$ is the rotational component ($\mathcal{Q}$ is the set of unit quaternions). Moreover, let us denote the desired relative pose associated with a link l by (p_l, q_l), and the covariance matrix by I_l. The error function of l is defined as follows.

$$e_l(x_{n_l}, x_{n'_l}) = \begin{bmatrix} q_{n_l}^{-1}(p_{n'_l} - p_{n_l}) - p_l \\ v(q_l^{-1}(q_{n_l}^{-1} q_{n'_l})) \end{bmatrix} \tag{1}$$

Here, for $p \in \mathbb{R}^3$ and $q \in \mathcal{Q}$, qp applies rotation expressed by q to p, and q^{-1} denotes the conjugate of q. This function expresses the error between the relative pose of the two nodes calculated from their absolute poses and the desired relative pose associated with the link. The function $v(q)$ returns a vector in $\mathbb{R}^3$ that consists of the x, y, and z components of a quaternion q. Moreover, the quadratic cost function of l is defined as

$$J_l(x_{n_l}, x_{n'_l}) = e_l^{\mathsf{T}} I_l e_l. \tag{2}$$

[1]https://github.com/ytazz/mrpgo

A posegraph optimization problem is formulated as a minimization problem of the sum of cost of all links.

$$\begin{aligned} &\min_{x_n, n \in \mathcal{N}} J \\ &J = \sum_{l \in \mathcal{L}} J_l(x_{n_l}, x_{n'_l}) \end{aligned} \tag{3}$$

A posegraph optimization problem has nonlinearity because of rotation transformation of SE(3) (or SE(2)). A widely accepted strategy is to solve this problem by iteration. Namely, instead of optimizing the poses directly, we define correction of poses as new decision variables and formulate a new problem which is to reduce the cost as much as possible. The change of pose of $n \in \mathcal{N}$ is $\delta x_n = [\delta p_n^{\mathsf{T}}\, \delta q_n^{\mathsf{T}}]^{\mathsf{T}} \in \mathbb{R}^6$, where δq_n is an angle-axis vector that expresses a change of orientation. The difference of the error function e_l is derived as follows.

$$\begin{aligned} \delta e_l &= \frac{\partial e_l}{\partial x_{n_l}} \delta x_{n_l} + \frac{\partial e_l}{\partial x_{n'_l}} \delta x_{n'_l}, \\ \frac{\partial e_l}{\partial x_{n_l}} &= \begin{bmatrix} -R_{n_l}^{\mathsf{T}} & R_{n_l}^{\mathsf{T}}(p_{n'_l} - p_{n_l})^{\times} \\ O & -\frac{1}{2}(R_{n_l} R_l)^{\mathsf{T}} \end{bmatrix}, \\ \frac{\partial e_l}{\partial x_{n'_l}} &= \begin{bmatrix} R_{n_l}^{\mathsf{T}} & O \\ O & \frac{1}{2}(R_{n_l} R_l)^{\mathsf{T}} \end{bmatrix} \end{aligned} \tag{4}$$

where R_{n_l} and R_l is a rotation matrix equivalent to q_{n_l} and q_l, respectively, and $r^{\times}$ is a skew-symmetric matrix equivalent to a cross product with r. Further more, the difference of the cost function J_l is derived as follows.

$$\begin{aligned} \delta J_l &= 2(I_l e_l)^{\mathsf{T}} \delta e_l + \delta e_l^{\mathsf{T}} I_l \delta e_l \\ &= 2\left(\begin{bmatrix} \frac{\partial e_l}{\partial x_{n_l}}^{\mathsf{T}} \\ \frac{\partial e_l}{\partial x_{n'_l}}^{\mathsf{T}} \end{bmatrix} I_l e_l \right)^{\mathsf{T}} \begin{bmatrix} \delta x_{n_l} \\ \delta x_{n'_l} \end{bmatrix} \\ &+ \begin{bmatrix} \delta x_{n_l} \\ \delta x_{n'_l} \end{bmatrix}^{\mathsf{T}} \left(\begin{bmatrix} \frac{\partial e_l}{\partial x_{n_l}}^{\mathsf{T}} \\ \frac{\partial e_l}{\partial x_{n'_l}}^{\mathsf{T}} \end{bmatrix} I_l \begin{bmatrix} \frac{\partial e_l}{\partial x_{n_l}} & \frac{\partial e_l}{\partial x_{n'_l}} \end{bmatrix} \right) \begin{bmatrix} \delta x_{n_l} \\ \delta x_{n'_l} \end{bmatrix} \\ &=: 2 d_l^{\mathsf{T}} \begin{bmatrix} \delta x_{n_l} \\ \delta x_{n'_l} \end{bmatrix} + \begin{bmatrix} \delta x_{n_l} \\ \delta x_{n'_l} \end{bmatrix}^{\mathsf{T}} H_l \begin{bmatrix} \delta x_{n_l} \\ \delta x_{n'_l} \end{bmatrix} \end{aligned} \tag{5}$$

Here, d_l and H_l are the gradient and the pseudo Hessian of δJ_l. We call it the pseudo Hessian because the second derivative of e_l is ignored. In the remaining discussion, H_l is referred to as the Hessian for simplicity. Now, the differential version of the original cost minimization problem can be written as follows:

$$\begin{aligned} &\min_{\delta x} \delta J, \\ &\delta J = \sum_{l \in \mathcal{L}} \delta J_l = 2 d^{\mathsf{T}} \delta x + \delta x^{\mathsf{T}} H \delta x \end{aligned} \tag{6}$$

Here, d and H are the gradient and the Hessian of the overall problem. Since this is a simple minimization problem of a quadratic function, its minimizer can be obtained by solving the following linear system of equations.

$$H \delta x + d = 0 \tag{7}$$

Once δx is obtained, the pose of each node is update as follows:

$$\begin{aligned} p_n &\leftarrow p_n + \delta p_n, \\ q_n &\leftarrow q\left(\frac{\delta q_n}{\|\delta q_n\|}, \|\delta q_n\|\right) q_n \end{aligned} \tag{8}$$

where $q(\eta, \theta)$ is a quaternion expressing a rotation around η with an angle θ. The above procedure is repeated for a specified number of times or until the cost reduction δJ becomes smaller than a threshold. The most computationally demanding part is computation of the linear equation (7). The Hessian H generally possesses a sparse structure reflecting the topology of the posegraph. An effective solution known to date is to use a sparse direct solver such as CHOLMOD for computing (7). In order to achieve further improvement of computational efficiency, in the next section we propose a multi-resolution approach which decomposes the original minimization problem into a hierarchy of smaller problems.

III. Multi-resolution Transformation of Pose-graphs

A. Multi-resolution Representation of Posegraph Transformation

In the following, a multi-resolution representation of posegraph transformation, which constitutes the foundation of the proposed method, is described. There are two basic ways for parametrizing the transformation of a posegraph: absolute and relative. The absolute parametrization considers the absolute pose of each node (i.e., pose with respect to the global coordinate frame) as an independent variable. The relative parametrization, on the other hand, considers a serial chain that connects the nodes from the first to the last one, and treats the relative pose between consecutive nodes as independent variables. Apart from these two basic parametrizations, a serial chain-based multi-resolution (MR) parametrizations has been proposed in [18] and in [20]. In the following, we propose an alternative spanning tree-based MR parametrization.

In the first step, consider a spanning tree of a posegraph. More specifically, we consider a breadth-first-search tree. The selection of the root node of the spanning tree is arbitrary. Unless there is some special reason, one can simply choose the first node in the posegraph as the root node. Let $\mathcal{N}_d$ be a subset of nodes whose depth in the spanning tree is d. Then $\mathcal{N}_0, \mathcal{N}_1, \ldots, \mathcal{N}_D$ defines a partition of the whole node set $\mathcal{N}$, where D is the maximum depth. Let us further partition the depths $0, 1, \ldots, D$ into multiple *levels* $\mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_L$. Here, I_0 consists of every second depth N_d, I_1 of every fourth, and so on in an interleaved way. This means every level encompasses every second of those N_d not included in previous levels. Finally, the highest level I_L includes all depths not included in the lower levels. Formally, each level is defined as follows:

$$\mathcal{I}_i := \begin{cases} \{d \,|\, d \equiv 0 \,(\mathrm{mod}\, 2^i), \;\; d \not\equiv 0 \,(\mathrm{mod}\, 2^{i+1})\} & (i < L) \\ \{d \,|\, d \equiv 0 \,(\mathrm{mod}\, 2^i)\} & (i = L) \end{cases} \tag{9}$$

For example, if $L = 2$, we have $\mathcal{I}_0 = \{1, 3, 5, \ldots\}$, $\mathcal{I}_1 = \{2, 6, 10, \ldots\}$, and $\mathcal{I}_2 = \{0, 4, 8, \ldots\}$. A simple example of partitioning is illustrated in Fig. 1(a). Moreover, actual partitioning of Intel dataset is visualized in Fig. 1(b). When $d \in \mathcal{I}_i$, we say that d (and also any $n \in \mathcal{N}_d$) belongs to the i-th level. For $i < L$, depths that belong to the same level are assigned serial indices starting from 0 and they are called blocks of that level. For example, $\mathcal{N}_{10}$ belongs to the 2nd block of level 1. All depths that belong to the maximum level are assigned the block index 0.

From the definition of breadth-first-search spanning tree, the depth of every node is equal to the length of the shortest path from the root node to that node. This indicates that every pair of nodes connected by a link either belong to the same depth or belong to adjacent depths. Moreover, from (9), no adjacent depths belong to the same level. From these two facts, we can conclude that nodes that belong to different blocks of the same level are disjoint. For example, in Fig. 1(a), depths 3 and 5 are two different blocks of level 0. Here, no link of the posegraph connects $\mathcal{N}_3$ and $\mathcal{N}_5$ directly. This characteristic is utilized for parallelization of the solution algorithm presented in Section III-C.

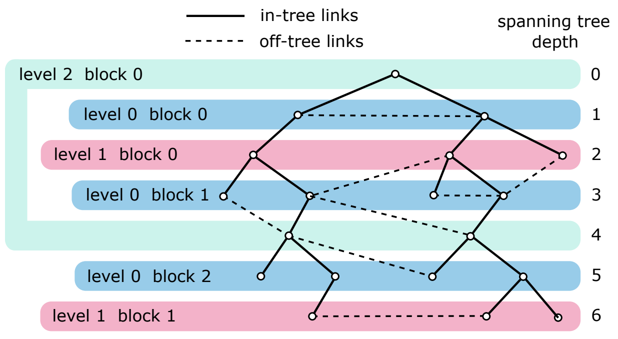
For a node $n \in \mathcal{N}_d$, its *supernode* is defined as its nearest ascendant in the spanning tree which belongs to a higher level than n. From (9), for any node n whose level is $i < L$, its supernode is uniquely defined. Nodes that belong to the L-th level do not have supernodes.

Now, let n be a node that belongs to the i-th level and n' be its supernode which belongs to the i'-th level. By definition, $i < i' \leq L$ holds. In the multi-resolution parametrization, the change of pose of n is expressed as follows:

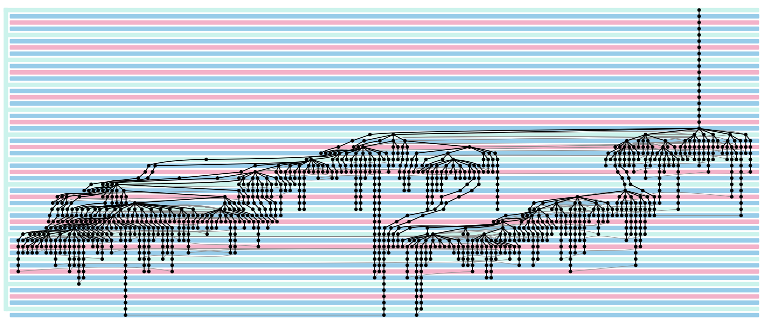
$$\begin{aligned} \delta x_n &= \begin{bmatrix} I & -(p_n - p_{n'})^\times \\ O & I \end{bmatrix} \delta x_{n'} + \delta \hat{x}_n \\ &=: A_{n,n'} \delta x_{n'} + \delta \hat{x}_n \end{aligned} \tag{10}$$

The first term in the right hand side of (10) expresses the extrapolation of the change of pose of the supernode n' to that of n; it expresses how much n moves when n' moves by $\delta x_{n'}$ while the relative pose between n and n' in the spanning tree is fixed. The second term is added to the first to make additional modification to the change of pose. Note that unless n' is in the highest level, $\delta x_{n'}$ is given by $\delta x_{n'} = A_{n'} \delta x_{n''} + \hat{x}_{n'}$, where n'' is the supernode of n'. In this manner, in the MR parametrization, the change of pose of each node is expressed as a superposition of $\hat{x}_n, \hat{x}_{n'}, \hat{x}_{n''}, \ldots$. In Fig. 2, the concept of the proposed parametrization is illustrated using an analogy of mechanical linkage. For ease of illustration, a single path from the root node to a certain leaf node in the spanning tree is shown, where indices indicate depths. In the figure, movement of different levels (level 0, 1, and 2) are expressed as mechanical links with different colors. The movement of Node 4 is transmitted to Nodes 5, 6, and 7 via a link representing the level-2 movement. Similarly, the movement of Node 6 is transmitted to Node 7 via a level-1 link.

Now, let us define δx_i as a concatenation of all δx_n of nodes that belong to the i-th level. Since the supernodes of these nodes all belong to the $i+1$-th level or higher, from the



(a) Illustrative example



(b) Partitioning of Intel dataset

Fig. 1. Partitioning of nodes into levels and blocks.

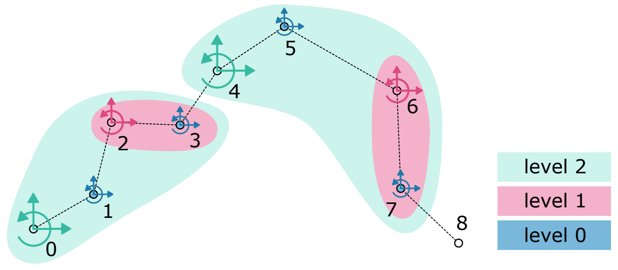


Fig. 2. Illustration of hierarchical pose transformation.

Algorithm 1 MR-PGO: outer loop

```
1: Initialize $x$
2: loop $n_{\mathrm{iter}}$ times
3:     Approximate $J$ at $x$ and compute $d$ and $H$
4:     Compute $\delta x$ by using Algorithm 2
5:     if $\delta J$ is small enough then
6:         break
7:     end if
8:     update $x$ by $\delta x$ using (8).
9: end loop
10: return $x$
```

transformation described above, we have

$$\delta x_i = A_i \delta x_{L:i+1} + \delta \hat{x}_i. \tag{11}$$

Here, $\delta x_{i_1:i_2} = [\delta x_{i_1}^\mathsf{T}, \dots, \delta x_{i_2}^\mathsf{T}]^\mathsf{T}$, and A_i can be constructed from $A_{n,n'}$s defined in (10).

B. Multi-resolution Transformation

When iterative methods such as the Gauss-Seidel method is directly applied to posegraph optimization, it generally requires a large number of iterations for convergence. This is because it requires many iterations for a local change of pose and its influence on the cost of neighboring links to propagate over the graph to cause macroscopic change of pose. The MR parametrization described in the previous section facilitates quick propagation of pose change through higher (coarser) levels, and at the same time, enables small adjustment of poses through lower (finer) levels.

In the following, we show that the MR parametrization transforms the gradient and the Hessian of the original minimization problem (6). Let us reorder the components of δx so that it expressed as

$$\delta x = \begin{bmatrix} \delta x_L^\mathsf{T} & \delta x_{L-1}^\mathsf{T} & \dots & \delta x_0^\mathsf{T} \end{bmatrix}^\mathsf{T} \tag{12}$$

In the MR parametrization, $\delta \hat{x}_i$ instead of δx_i becomes the decision variable. Therefore, the new decision variable is defined as:

$$\delta \hat{x} = \begin{bmatrix} \delta x_L^\mathsf{T} & \delta \hat{x}_{L-1}^\mathsf{T} & \dots & \delta \hat{x}_0^\mathsf{T} \end{bmatrix}^\mathsf{T} \tag{13}$$

Given $\delta \hat{x}$, one can obtain δx by applying (11) recursively from $i = L-1, \dots, 0$. Now, consider transforming the original cost minimization problem (6), whose decision variable is δx, to another minimization problem with respect to $\delta \hat{x}$. Consider the following transformation:

$$\begin{bmatrix} \delta x_{L:i+1} \\ \delta x_i \\ \delta \hat{x}_{i-1:0} \end{bmatrix} = \begin{bmatrix} I & O & O \\ A_i & I & O \\ O & O & I \end{bmatrix} \begin{bmatrix} \delta x_{L:i+1} \\ \delta \hat{x}_i \\ \delta \hat{x}_{i-1:0} \end{bmatrix} =: G_l \begin{bmatrix} \delta x_{L:i+1} \\ \delta \hat{x}_i \\ \delta \hat{x}_{i-1:0} \end{bmatrix} \tag{14}$$

This transformation replaces $\delta \hat{x}_i$ with δx_i. By sequentially applying this transformation from $i = L-1$ to $i = 0$, we obtain the transformation from $\delta \hat{x}$ to δx.

$$\delta x = \left[\prod_{i=0}^{L-1} G_l \right] \hat{x} =: G\, \delta \hat{x} \tag{15}$$

Using this transformation, the original minimization is transformed to

$$\min_{\delta \hat{x}} \; 2\hat{d}^\mathsf{T} \delta \hat{x} + \delta \hat{x}^\mathsf{T} \hat{H} \delta \hat{x} \tag{16}$$

where $\hat{d} = G^\mathsf{T} d$, $\hat{H} = G^\mathsf{T} H G$.

C. Detail of the Proposed Algorithm

The pseudocode of the proposed multi-resolution posegraph optimizer (MR-PGO) is shown in Algorithm 1 and 2. Algorithm 1 shows the outer loop of the Gauss-Newton method, while Algorithm 2 shows the inner loop, where δx is computed based on the MR parametrization. The maximum iteration count of Algorithm 1 is specified by the parameter n_{iter}. The basic idea of Algorithm 2 is to apply the block Gauss-Seidel method to the transformed equation (16). In Lines 1-3, the

Algorithm 2 MR-PGO: inner loop

```
1:  for $i \in 0, 1, \ldots, L-1$ do
2:      $d \leftarrow G_i^\mathsf{T} d, \quad H \leftarrow G_i^\mathsf{T} H G_i$
3:  end for
4:  loop $n_\mathrm{GS}$ times
5:      for $i \in L, L-1, \ldots, 0$ do
6:          $\delta\delta\hat{x}_l \leftarrow \mathrm{LINSOLVE}(H_{i,i}, -d_i)$
7:          for $i' \neq i$ do
8:              $d_{i'} \leftarrow d_{i'} + H_{i',i}\delta\delta\hat{x}_i$
9:          end for
10:         $\delta\hat{x}_i \leftarrow \delta\hat{x}_i + \delta\delta\hat{x}_i$
11:     end for
12: end loop
13: for $i \in L-1, \ldots, 0$ do
14:     $\delta x_i \leftarrow A_i \delta x_{L:i+1} + \delta\hat{x}_i$
15: end for
16: return $\delta x$
```

transformation of the gradient and the Hessian is computed. In Lines 4-12, the G-S iteration is performed. The number of iterations is fixed and given by n_GS. It will be shown in numerical examples that setting n_GS to as small as 1 can achieve good convergence. In each iteration, from $i = L$ to $i = 0$, a sub-equation corresponding to the i-th level is solved (Line 6) and the gradient of other levels is updated (Lines 7-9). Here, LINSOLVE(A, b) solves a linear system of equations $Ax = b$ for a symmetric positive definite matrix A and a coefficient vector b and returns the solution x. A sparse direct solver CHOLMOD is used to implement this routine to fully exploit the sparsity of the Hessian, Moreover, as described in Section III-A, nodes that belong to different blocks of the same level are disjoint. Thanks to this property, each $H_{i,i}$ with $i < L$ is a block-diagonal matrix. Therefore, there is no need to solve LINSOLVE($H_{i,i}, -d_i$) in Line 6 as a single linear equation, but it can be broken down into smaller equations corresponding to blocks and computed in parallel. This is a unique advantage of the proposed spanning tree-based approach.

IV. Numerical Examples

The proposed method were compared with g2o [8] and SE-Sync [15] using public and custom datasets. The proposed algorithm was implemented as a C++ program. Multi-thread parallelization based on OpenMP is used for parts of the algorithm that can be executed in parallel, as described earlier. For all solvers, Microsoft Visual Studio was used as a C/C++ compiler, Intel MKL Lapack/BLAS was used for linear algebraic computation. For the proposed method and g2o, SuiteSparse CHOLMOD was used for a linear solver. The executables were run of a Windows computer with AMD Ryzen 9 5950X CPU with 16 cores and 32 parallel threads. The proposed method and g2o perform spanning tree-based initialization prior to iterative optimization, while SE-Sync performs chordal initialization.

The name, size and space (SE2 or SE3) of datasets used for evaluation are summarized in Table I. Among them, the original Rim dataset included non-positive definite Hessians, which make the optimization problem infeasible. Therefore, before evaluation, this dataset was sanitized by replacing all Hessians with $100I$ (I is the identity matrix). For some datasets, posegraphs before and after optimization are visualized in Fig. 3. In these figures, black dots indicate nodes, while red lines indicate links. For each dataset, the upper and lower figures indicate the posegraph before and after optimization.

The sparsity pattern of the Hessian matrices of the datasets is visualized in Fig. 4 and Fig. 5. Figure 4 shows the sparsity pattern of the original Hessian H, which reflects the topology of the posegraph. Fig. 5, on the other hand, shows the sparsity pattern of the transformed Hessian $\hat{H}$ in (16). In addition to the transformation from H to $\hat{H}$, reordering of rows and columns is applied so that nodes are sorted with respect to level in the descending order. Nodes in the same level are further sorted with respect to block index in the ascending order. In each figure, blocks are indicated by squares with a color unique to each level. It can be observed that each level presents characteristic block-diagonal structure, which can be utilized for parallelization.

The lower part of Table I summarizes the total computation time and final cost of the methods compared. For the proposed method, the number of the G-S iterations was set as $n_\mathrm{GS} = 1$, and the number of outer iterations was fixed to $n_\mathrm{iter} = 10$. The SE-Sync solver optimizes an objective function that is different from the one used in g2o. Therefore, the value of cost produced by SE-Sync together with its equivalent value of the g2o cost are shown in the table. The latter is computed by reloading the posegraph optimized by SE-Sync on g2o. The converted cost for 2D datasets are not listed due to minor technical issue. For datasets with moderate sizes, all method performed well, while g2o generally showed least computation time. For large datasets such as Grid3D, Globe10k, and Globe100k, the computation time of g2o and SE-Sync increased dramatically. In these cases, the proposed method with $L = 2, 4$ performed better in terms of total computation time. However, the proposed method showed poorer rate of convergence with greater L, and as a result, the solution did not converge close enough to the optimum within 10 iterations. It must also be noted that evaluation of pure algorithmic efficiency becomes difficult for large datasets because file I/O and memory allocation take large portion of total computation time.

In the following, the proposed method in three different settings of resolution levels ($L = 0, 2, 4$) and g2o are compared in terms of rate of convergence and computation time per iteration. Figure 6 shows the reduction of cost in the first 15 iterations. The convergence rate of first several iterations are roughly equal for all methods. In close observation, however, the convergence of the proposed method tends to slow down near the optimal solution, and this tendency becomes more significant with greater values of L. As L becomes larger, the original system of equations is decomposed into a greater number of smaller blocks. This, together with the fact that only one subiteration of block Gauss-Seidel is performed, seem to have contributed to this slight deterioration of convergence.

Figure 7 shows computation time for one iteration (average of 100 iterations). For the proposed method, a breakdown of

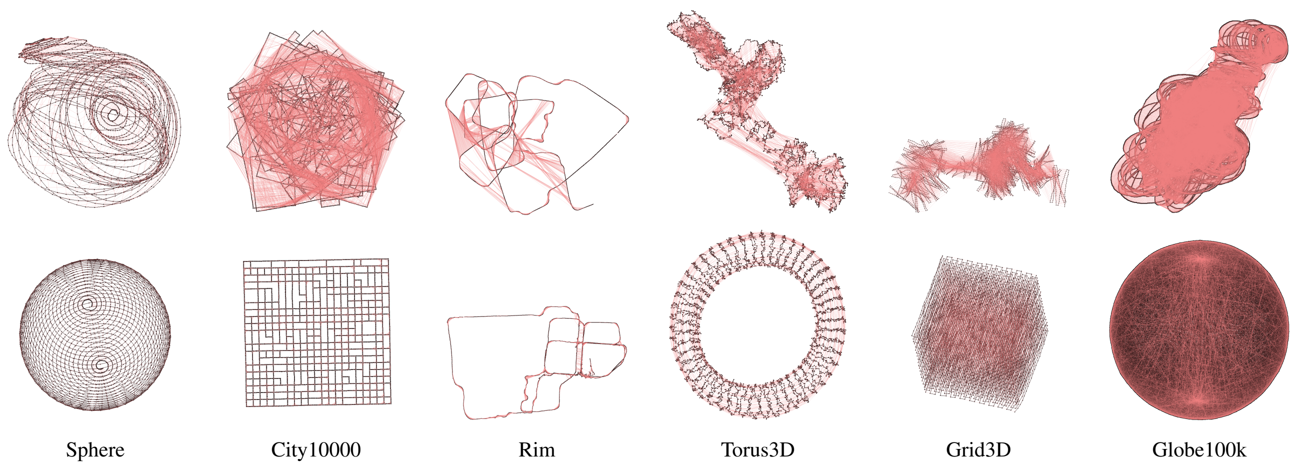


Fig. 3. Posegraphs before and after optimization

TABLE I
COMPARISON OF COMPUTATION TIME AND OPTIMAL COST

		Intel	Sphere	City10000	Rim	Torus3D	Grid3D	Globe10k	Globe100k
	num. of nodes	1,228	2,500	10,000	10,195	5,000	8,000	10,000	99,856
	num. of edges	1,483	4,949	20,687	29,743	9,048	22,236	20,899	200,395
	space	SE2	SE3	SE2	SE3	SE3	SE3	SE3	SE3
proposed $L = 0$	comp.time [s]	0.077	0.631	1.194	2.954	1.440	20.73	12.49	130.3
	cost	214.17	727.24	511.986	5978.59	14576.77	49216.79	6131.69	40254.69
proposed $L = 2$	comp.time [s]	0.085	0.342	0.950	1.703	0.640	2.497	3.37	71.9
	cost	230.29	829.89	523.40	6957.23	14896.57	51300.87	7389.12	41430.71
proposed $L = 4$	comp.time [s]	0.096	0.317	0.974	1.676	0.586	1.575	2.54	53.56
	cost	255.11	1355.69	575.93	7719.90	16365.64	59070.62	10790.95	57623.52
g2o	comp.time [s]	0.0089	0.170	0.196	1.744	0.406	8.60	9.19	133.0
	cost	215.83	727.15	511.98	5925.96	14574.76	49204.03	6131.63	40254.50
SE-Sync	comp.time [s]	0.186	0.748	11.86	8.06	1.48	8.09	9.82	476.029
	cost (native)	393.7	1687.01	638.62	9107.58	24227	84319	9456.98	63952.6
	cost (g2o chi2)	-	858.81	-	6312.96	14916.03	51627.72	6417.82	40654.49

computation time is also shown. Here, 'prepare' indicates the amount of time spent for calculating the gradient and Hessian and computing the multi-resolution transformation. Labels 'lv0', 'lv1'... indicate time spent for computing the linear equation of each level. As shown in Fig. 5, in most examples, each level contains not more than 30 blocks, whereas the computer used for evaluation was capable of running 32 parallel threads. This means that for each level, all sub-equations were computed in parallel, and consequently, the bottle-neck of computation time is the size of the largest block in each level. It would have taken longer time if the algorithm was run on a CPU with a smaller number of cores.

For small problems such as Intel, g2o performs faster than the proposed method. This is because for a small problem, the linear equation can be solved in quite a small amount of time without any transformation, and the transformation of the gradient and the Hessian requires almost as much time as solving the equation itself. For larger problems such as Rim, Torus3D, and Grid3D, the proposed method outperforms g2o in terms of computation time for single iteration. It can be seen that reduction in computation time for solving linear equations is so significant that the total computation time including preparation is much smaller than that of g2o. It was also an interesting finding that the computation time of g2o varied greatly between datasets having similar sizes. For example, although Rim and Grid3D are almost the same size, the computation time per iteration of the latter was ten times greater. One possible reason is that the supernodal factorization technique used in CHOLMOD did not work well for datasets having a certain sparsity pattern. For the proposed method, by contrast, the computation time per iteration regularly increased with respect to the size of the problem. This is considered to be a favorable characteristic for estimating the overall computation time before applying optimization to a new large-scale dataset.

V. CONCLUSION

This paper proposed a method for posegraph optimization that utilizes a spanning tree-based multi-resolution parametrization of posegraph transformation. Experimental results using open data set showed that the proposed multi-resolution strategy significantly reduces computation time while limiting the deterioration of convergence speed to an acceptable level. At the point of submission, the proposed algorithm is being prepared to be made publicly available as an open-source software library.

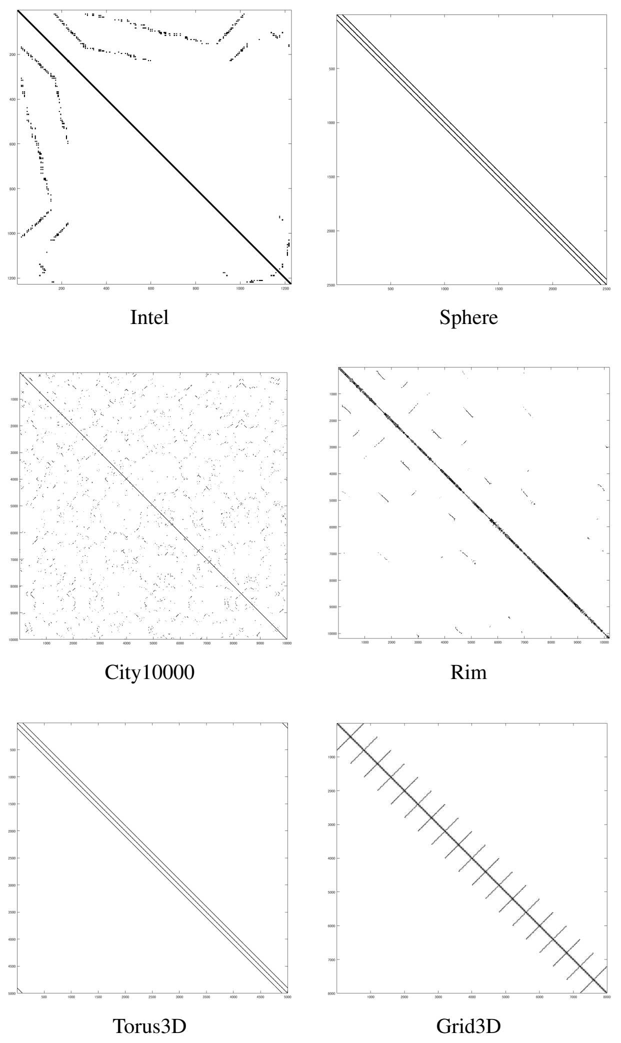


Fig. 4. Sparsity pattern of original Hessian

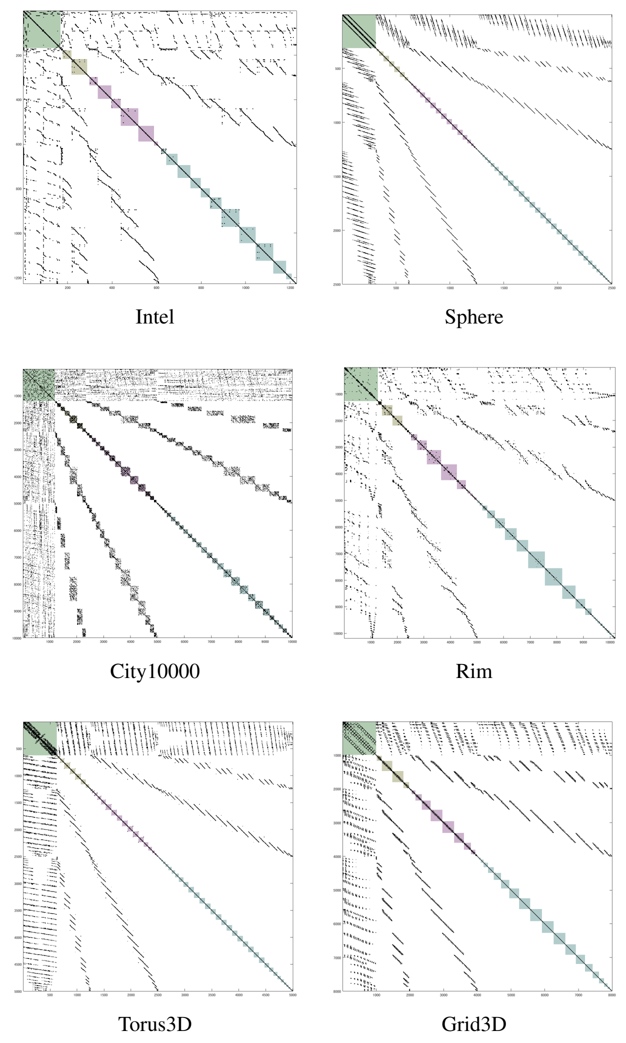


Fig. 5. Sparsity pattern of transformed and reordered Hessian

REFERENCES

[1] F. Lu and E. Milios: "Globally Consistent Range Scan Alignment for Environment Mapping", Autonomous Robots, Vol. 4, pp. 333–349, 1997.

[2] F. Dellaert and M. Kaess: "Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing," Int. J. Robotics Research, Vol.25, No.12, pp.1181-1203, 2006.

[3] M. Kaess, A. Ranganathan and F. Dellaert: "iSAM: Fast Incremental Smoothing and Mapping with Efficient Data Association," IEEE International Conference on Robotics and Automation, pp. 1670-1677, 2007.

[4] M. Kaess, H. Johannsson, R. Roberts, V. Ila2, J. J. Leonard and F. Dellaert: "iSAM2: Incremental smoothing and mapping using the Bayes tree", The International Journal of Robotics Research, Vol.31, No.2, pp.216-235, 2012.

[5] A. Cunningham, M. Paluri and F. Dellaert: "DDF-SAM: Fully distributed SLAM using Constrained Factor Graphs," IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 3025-3030, 2010.

[6] A. Cunningham, V. Indelman and F. Dellaert: "DDF-SAM 2.0: Consistent distributed smoothing and mapping," IEEE International Conference on Robotics and Automation, pp. 5220-5227, 2013.

[7] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam: "Algorithm 887: CHOLMOD, Supernodal Sparse Cholesky Factorization and Update/Downdate," ACM Trans. Mathematical Software, Vol.35, No. 3, pp.1-14, 2008.

[8] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige and W. Burgard: "G2o: A general framework for graph optimization," IEEE Int. Conf. Robotics and Automation, pp. 3607-3613, 2011.

[9] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss and W. Burgard: "Robust map optimization using dynamic covariance scaling," IEEE Int. Conf. Robotics and Automation, pp. 62-69, 2013.

[10] Y. Latif, C. Cadena and J. Neira: "Robust loop closing over time for pose graph SLAM," Int. J. Robotics Research, Vol.32, No.14, pp.1611-1626, 2013.

[11] N. Sünderhauf and P. Protzel: "Switchable constraints vs. max-mixture models vs. RRR - A comparison of three approaches to robust pose graph SLAM," IEEE Int. Conf. Robotics and Automation, pp. 5198-5203, 2013.

[12] H. Kretzschmar and C. Stachniss: "Information-theoretic compression of pose graphs for laser-based SLAM," Int. J. Robotics Research, Vol.31, No.11, pp.1219-1230, 2012.

[13] G. Huang, M. Kaess and J. J. Leonard: "Consistent sparsification for graph optimization," European Conf. Mobile Robots, pp. 150-157, 2013.

[14] J. Vallvé, J. Solà and J. Andrade-Cetto: "Pose-graph SLAM sparsification using factor descent," Robotics and Autonomous Systems, Vol.119, pp. 108-118, 2019.

[15] D. M. Rosen, L. Carlone, A. S. Bandeira, J. J. Leonard: "SE-Sync: A Certifiably Correct Algorithm for Synchronization over the Special Euclidean Group", The International Journal of Robotics Research, Vol. 38, No. 2-3, pp. 95-125, 2019.

[16] Y. Tian, K. Khosoussi, D. M. Rosen and J. P. How: "Distributed Certifiably Correct Pose-Graph Optimization", IEEE Transactions on Robotics, vol. 37, no. 6, pp. 2137-2156, 2021.

[17] F. Bai, T. Vidal-Calleja and G. Grisetti: "Sparse Pose Graph Optimization in Cycle Space", IEEE Transactions on Robotics, vol. 37, no. 5, pp. 1381-1400, 2021.

[18] U. Frese, P. Larsson and T. Duckett: "A Multilevel Relaxation Algorithm

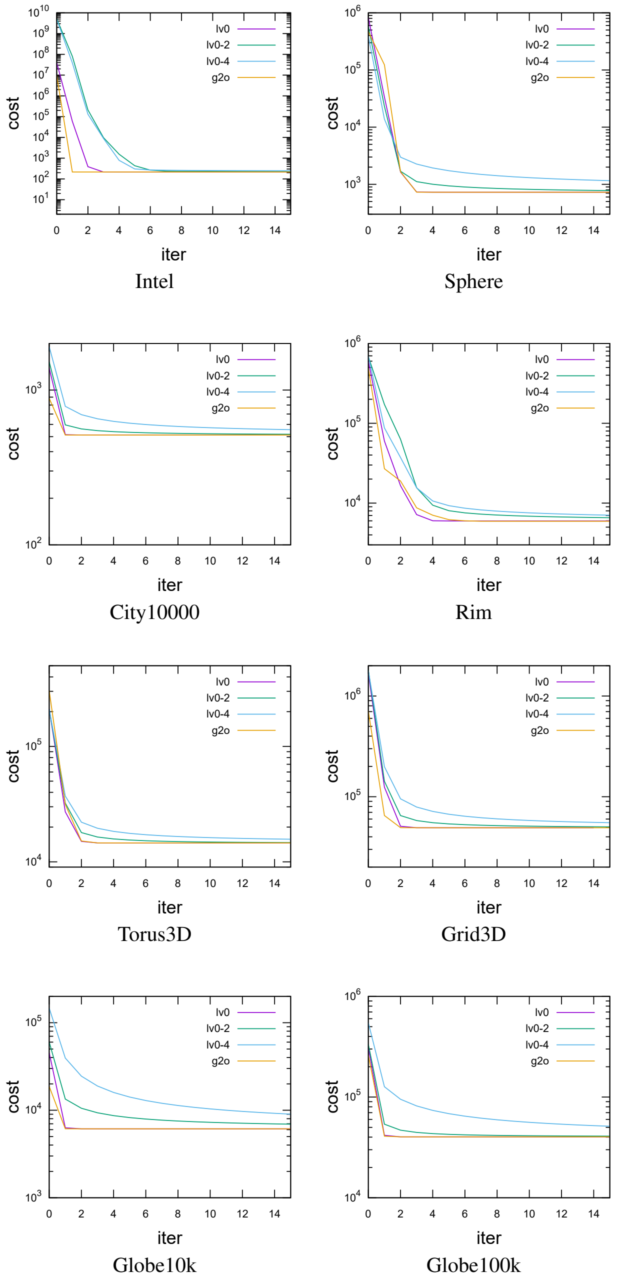


Fig. 6. Relationship between error and number of iterations.

for Simultaneous Localization and Mapping”, IEEE Trans. Robotics, Vol. 21, No. 2, pp. 196-207, 2005.

[19] G. Grisetti, R. Kummerle, C. Stachniss, U. Frese and C. Hertzberg: “Hierarchical optimization on manifolds for online 2D and 3D mapping,” IEEE Int. Conf. Robotics and Automation, pp. 273-278, 2010.

[20] S. Anderson, F. Dellaert and T. D. Barfoot: “A hierarchical wavelet decomposition for continuous-time SLAM,” IEEE Int. Conf. Robotics and Automation, pp. 373-380, 2014.

[21] T. Guadagnino, L. D. Giammarino and G. Grisetti: “HiPE: Hierarchical Initialization for Pose Graphs”. IEEE Robotics and Automation Letters, vol. 7, no. 1, pp. 287-294, 2022.

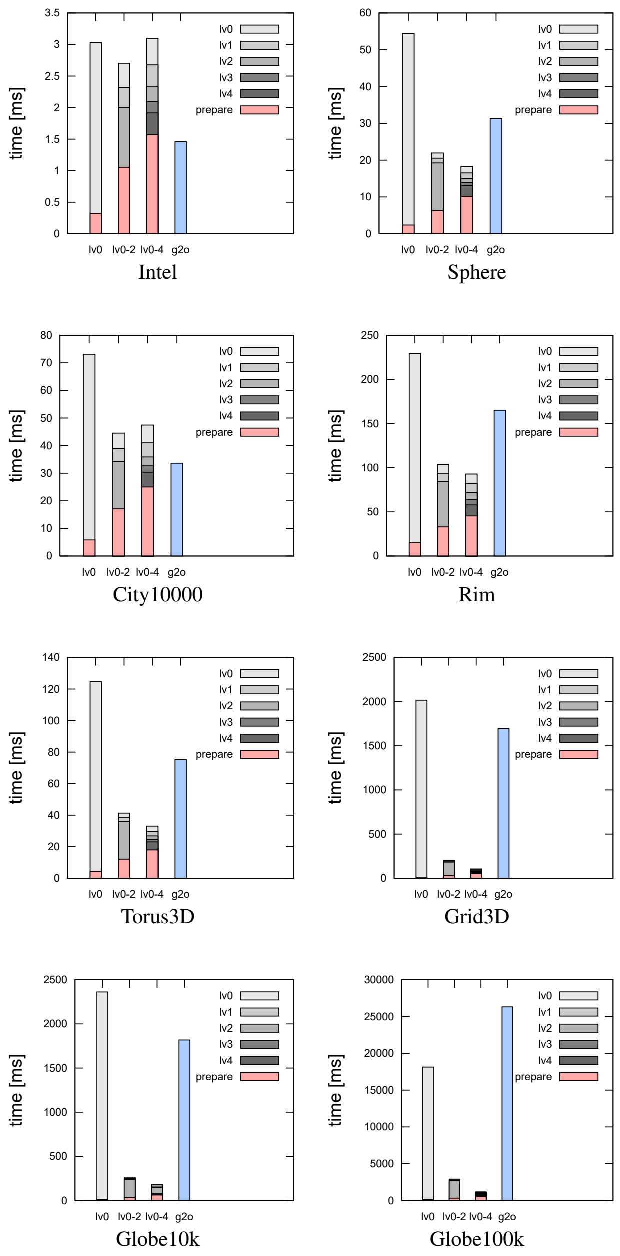


Fig. 7. Comparison of computation time per iteration.