Incremental Cycle Bases for Cycle-Based Pose Graph Optimization

Brendon Forsgren, Kevin Brink, Prashant Ganesh, *Member, IEEE*, and Timothy W. McLain

***Abstract*—Pose graph optimization is a special case of the simultaneous localization and mapping problem where the only variables to be estimated are pose variables and the only measurements are inter-pose constraints. The vast majority of pose graph optimization techniques are vertex based (variables are robot poses), but recent work has parameterized the pose graph optimization problem in a relative fashion (variables are the transformations between poses) that utilizes a minimum cycle basis to maximize the sparsity of the problem. We explore the construction of a cycle basis in an incremental manner while maximizing the sparsity. We validate an algorithm that constructs a sparse cycle basis incrementally and compare its performance with a minimum cycle basis. Additionally, we present an algorithm to approximate the minimum cycle basis of two graphs that are sparsely connected as is common in multi-agent scenarios. Lastly, the relative parameterization of pose graph optimization has been limited to using rigid body transforms on $SE(2)$ or $SE(3)$ as the constraints between poses. We introduce a methodology to allow for the use of lower-degree-of-freedom measurements in the relative pose graph optimization problem. We provide extensive validation of our algorithms on standard benchmarks, simulated datasets, and custom hardware.**

***Index Terms*—SLAM, localization, multi-robot systems.**

I. Introduction

THE simultaneous localization and mapping (SLAM) is often modeled as a factor graph with variable nodes consisting of both pose and landmark variables, and factor nodes representing constraints between variables. A special case of SLAM, called pose graph optimization (PGO), occurs when the only variable nodes are robot poses and the only factor nodes are constraints between poses.

In robotics, PGO has become a common technique to estimate the discrete-time trajectory of a robot when only the robot trajectory is of interest, as in the cases described in [1], [2]. PGO is usually solved using nonlinear least squares (NLS) optimization techniques and, as such, requires an initial estimate to start the optimization. The convergence of the problem to the global minima requires that the initial guess be in the basin of convergence of the global minima or otherwise a suboptimal local minima will be found.

A. Related Work

The birth of modern SLAM algorithms started with the seminal paper by Lu and Milos [3]. Since then, the robotics community has developed many fast and robust algorithms to enable autonomous agents to solve the SLAM problem. Various techniques have been developed to increase the robustness of SLAM algorithms including obtaining a good initialization [4], [5], using robust cost functions [6], [7], using convex relaxations [8], [9], [10], and finding globally optimal solutions [11], [12]. At their core, these algorithms exploit the sparse nature of SLAM to efficiently compute estimates of the variables using sparse nonlinear solvers such as GTSAM [13] or g2o [14]. The sparsity of the problem is maintained by methods such as variable reordering [15]. The iSAM2 solver presented in [16], in addition to doing online variable reordering, provides a method to incrementally update the solution by only updating the affected variables.

Recent work has focused on a relative parameterization of PGO [17], [18], [19] where each robot pose is expressed in the frame of the pose that precedes it. Olson et al. [20] notes that the relative parameterization of PGO loses the sparse property present in the traditional global parameterization. Jackson et al. [18] show that the relative parameterization is better conditioned than the traditional global parameterization. Bai et al. [17] reformulate the problem as a constrained optimization problem by enforcing that cycles in the graph return to the origin when traversed. Bai notes that the set of cycles must form a cycle basis for the graph being optimized. In his most recent work [19], Bai presents a method to maximize the sparsity of the relative parameterization by forcing the set of cycle constraints to form a minimum cycle basis (MCB) and shows that the run time of the relative parameterization is comparable to that of the global parameterization. The primary drawback to the method is that computing a MCB is an expensive procedure, showcasing the need for methods to incrementally update the MCB.

Constructing a MCB is a well studied process in graph theory with the first polynomial time algorithm being developed by Horton [21]. The MCB can be found by first finding the all pairs shortest paths (APSP) data structure and then creating a Horton

Manuscript received 13 September 2022; accepted 27 December 2022. Date of publication 12 January 2023; date of current version 18 January 2023. This letter was recommended for publication by Associate Editor D. Gu and Editor S. Behnke upon evaluation of the reviewers' comments This work was supported in part by the Center for Unmanned Aircraft Systems (C-UAS), and in part by National Science Foundation Industry University Cooperative Research Center (IUCRC) under NSF Award IIP-1650547. *(Corresponding author: Brendon Forsgren.)*

Brendon Forsgren and Timothy W. McLain are with the Department of Mechanical Engineering, Brigham Young University, Provo, UT 84606 USA (e-mail: bforsgren29@gmail.com; mclain@byu.edu).

Kevin Brink is with Air Force Research Laboratory, Eglin Air Force Base, Eglin AFB, FL 32542 USA (e-mail: kevin.brink2@eglin.af.mil).

Prashant Ganesh is with the Department of Mechanical and Aerospace Engineering, University of Florida, Shalimar, FL 32579 USA (e-mail: prashant.ganesh@ufl.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3236580, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3236580

set. The Horton set is a set of cycles known to be a superset of a MCB. The set is ordered by weight and the MCB is found by taking the shortest cycles that form a cycle basis for the graph. Other algorithms to construct or approximate a MCB have been proposed [22], [23], [24], [25] that are less computationally complex than Horton's algorithm. However, all of these methods are for static graphs, or graphs without edge insertions or deletions. To our knowledge, no incremental algorithm to construct a MCB exists. Bai describes an algorithm in [17] based on a fundamental cycle basis that uses the odometry backbone to generate a sparse fundamental cycle basis. In his dissertation [26], Bai describes a greedy algorithm to incrementally approximate a MCB but does not evaluate the algorithm in the dissertation or any peer reviewed work.

The contributions made in this paper are as follows:

1) Extensive validation of the greedy incremental algorithm presented in [26].
2) An algorithm to generate a cycle basis of multiple graphs that are sparsely connected.
3) A methodology to use low-degree-of-freedom measurements in the relative parameterization of PGO.
4) Validation of the proposed algorithms on standard benchmarks, simulated data, and in hardware experiments.

II. Problem Definition

Here we define the PGO problem to be solved, relevant technical definitions, and other notation. A graph $\mathcal{G}$ is defined as $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E})$ where $\mathcal{G}$ is the graph, $\mathcal{V}$ are vertices, and $\mathcal{E}$ are the edges. A path, $\mathcal{P}$, in $\mathcal{G}$ is a subgraph of $\mathcal{G}$ where all vertices have degree two except for two vertices, which have degree one. A cycle in $\mathcal{G}$, denoted as $\mathcal{C}$, is a subgraph of $\mathcal{G}$ where all vertices have an even degree. The cycle space of $\mathcal{G}$ is the set of all the cycles in $\mathcal{G}$ and a cycle basis of $\mathcal{G}$, denoted as $\mathcal{B}$, is a set of independent cycles from which any cycle in $\mathcal{G}$ can be created by combining cycles in $\mathcal{B}$. It is known that the dimension of the cycle basis is $\nu = |\mathcal{E}| - |\mathcal{V}| + 1$ [19], where $|\cdot|$ denotes the cardinality of the set.

Now we define the PGO problem that we wish to solve. Given a graph $\mathcal{G} = \mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ are robot poses and $\mathcal{E}$ are the rigid body transformations (in 2D or 3D) between poses, we wish to estimate the edges in $\mathcal{E}$. The estimates are found by solving the following problem defined in [19]

$$\{\mathbf{T}_k\}_{k\in\mathcal{E}} = \arg\min \sum_{k\in\mathcal{E}} \|\mathrm{Log}(\tilde{\mathbf{T}}_k^{-1}\mathbf{T}_k)\|^2_{\Sigma_k}$$
$$\text{s.t. } \mathbf{I} = \prod_{\mathbf{T}_k\in\mathcal{C}_i} \mathbf{T}_k \;\; \forall \;\; \mathcal{C}_i \in \mathcal{B} \tag{1}$$

where $\mathbf{T}_k$ is an edge in $\mathcal{E}$ to be estimated, $\tilde{\mathbf{T}}_k$ is the measurement of edge $\mathbf{T}_k$, and $\mathcal{C}$ is a cycle in $\mathcal{B}$. From this solution the robot poses can be calculated by composing the rigid body transformations from the origin to the desired vertex along any path in $\mathcal{G}$.

III. Methods

Given the incremental nature of PGO problems in robotics, being able to incrementally create a sparse cycle basis to constrain the problem defined in (1) is desirable. However, it is known that a MCB does not behave well under standard graph operations (insertions, deletions, etc.), [27] meaning that there is no known way to update the MCB of a graph under a given operation. In this section we provide an overview of the incremental algorithm to construct a cycle basis presented in [26]. We additionally present an algorithm that will incrementally approximate the MCB of two disjoint graphs that become connected as is the case in multi-agent scenarios where inter-vehicle measurements are obtained. Lastly, we present a framework that will enable low degree of freedom measurements to be used in the cycle-based PGO problem defined in (1).

A. Incremental Cycle Basis Algorithm

Bai, in his dissertation [26], presents the following algorithm to incrementally approximate a MCB. Assume that a previously valid cycle basis, $\mathcal{B}_k$ for a graph $\mathcal{G}$ exists and that a new edge e_{ij} is introduced into the graph where i and j are non-sequential nodes. An approximate MCB of $\mathcal{G}$ can be defined as $\mathcal{B}_{k+1} = \mathcal{B}_k \cup \mathcal{C}_{ij}$ where $\mathcal{C}_{ij}$ is any cycle that contains e_{ij}. The sparsity of $\mathcal{B}_{k+1}$ can be maximized by defining the cycle $\mathcal{C}_{ij}$ such that it has minimum length. This is accomplished by letting $\mathcal{C}_{ij} = \mathcal{P}(i,j) \cup e_{ij}$ where $\mathcal{P}(i,j)$ is the shortest path between nodes i and j. A proof that this algorithm forms a valid cycle basis can be found in [26].

The complexity of this algorithm is much lower than that of the MCB algorithm presented in [19]. Bai notes that the most complex portion of his MCB algorithm is constructing an APSP structure and has a complexity of $O(nm\log n + n^2\log n + nm)$ where $n = |\mathcal{V}|$ and $m = |\mathcal{E}|$. In comparison, the incremental algorithm only requires finding the shortest path between two nodes, which can be done with a breadth first search and a complexity of $O(n+m)$.

B. Multi-Agent Incremental Cycle Basis Algorithm

In this section we propose an algorithm that will approximate the MCB of a graph that is created in multi-agent scenarios. Assume two robots are collecting data and each are forming their own graphs $\mathcal{G}^i$, where the superscript denotes the robot ID. Each robot is maintaining and updating its own cycle basis, be it an approximation or an exact MCB, $\mathcal{B}^i_k$, where the subscript k denotes at timestep k. The robots communicate periodically to share data. The data is used to form the graph $\mathcal{G}^{ij}$ and to estimate any inter-robot loop closures. The question becomes can the MCB, $\mathcal{B}^{ij}_k$, be approximated, where the superscript denotes the union of the two graphs and the connecting edges.

We begin by proving that a cycle basis can be constructed incrementally and then outline a heuristic to decrease the computational complexity. We define the set of inter-agent relative pose measurements $\mathcal{E}^{ab}_i = (e_0, e_1, \ldots, e_i)$, and $\mathcal{P}(a_i, a_j)$ as the shortest path between nodes i and j on agent a and likewise for agent b. Assume that cycle bases $\mathcal{B}^a$ and $\mathcal{B}^b$ are valid when the second inter-vehicle measurement is taken and $\mathcal{E}^{ab}_1$ is formed. It is easy to identify that $\mathcal{B}^a \cup \mathcal{B}^b \cup \mathcal{B}^{ab}_1$, where $\mathcal{B}^{ab}_1 = e_0 \cup \mathcal{P}(a_0, a_1) \cup e_1 \cup \mathcal{P}(b_0, b_1)$, is a valid cycle basis since edges e_0 and e_1 are in no cycles in either $\mathcal{B}^a$ or $\mathcal{B}^b$. Assuming that cycle bases $\mathcal{B}^a$ and $\mathcal{B}^b$ stay valid when the third inter-vehicle measurement is taken then, cycle basis $\mathcal{B}^{ab}_2 = \mathcal{B}^{ab}_1 \cup \mathcal{C}(e_2, \mathcal{E}^{ab}_1)$ is valid where $\mathcal{C}(e_2, \mathcal{E}^{ab}_1)$ is a cycle containing edges e_2 and any

edge in $\mathcal{E}_1^{ab}$. This process can be continued to show that the cycle basis at timestep k is valid so long as the new cycle added to the basis contains edges e_k and any edge in $\mathcal{E}_{k-1}^{ab}$.

Having shown that we can incrementally form a cycle basis of the union of connection of two graphs we note that the shortest cycle containing the new edge, e_k can be created by identifying which edge in $\mathcal{E}_{k-1}^{ab}$ makes the shortest cycle. This becomes computationally complex as the number of edges in $\mathcal{E}_{k-1}^{ab}$ increases. We present our algorithm in Algorithm 1 where we make the design decision to always use edges e_k and e_{k-1} when constructing loops to cut down on the computational complexity. Our algorithm is presented for the general case where multiple inter-vehicle relative pose measurements may be detected at one time as in the case when data is shared after a long period with no communication.

We note that our choice to use edges e_k and e_{k-1} when forming a loop comes with no performance guarantees and that this choice of ordering in Algorithm 1 will potentially produce longer cycles than another heuristic. While a more optimal solution would be to select the edge in $\mathcal{E}_{k-1}^{ab}$ that results in the shortest cycle, this becomes computationally complex as the number of edges in $\mathcal{E}^{ab}$ increases since a large number of shortest-path operations will need to be computed. Additional orderings of the newly arrived edges could be used or the whole cycle basis could be recomputed using an arbitrary order of the edges in $\mathcal{E}_k^{ab}$. The order of the edges would surely impact the lengths of the produced cycles but determining an optimal or near optimal ordering in an efficient manner is not obvious and is outside the scope of this paper and thus left as a topic of future work.

We note that the algorithm in [26] would also work for constructing a cycle basis of the the union of graphs $\mathcal{G}^i$ and $\mathcal{G}^j$. However, there are several reasons why our algorithm would be preferable. The first is that our algorithm is more scalable. Bai's algorithm operates over a single graph that will be growing faster than the individual graphs of each agent, meaning that graph operations such as BFS will take longer. The second benefit of our algorithm is that ours is parallelizable. Since our algorithm consists of operations over disconnected graphs, operations on different graphs can be done in parallel. Constructing cycles in parallel also improves the scalability of the algorithm by allowing more cycles to be constructed in the same processing time.

Additionally, our algorithm is beneficial if there is sufficient communication bandwidth for the agents to share their respective cycle bases. The communication of the cycle bases $\mathcal{B}_k^i$ and $\mathcal{B}_k^j$ eliminate the need for these cycle bases to be recomputed. If such communication is possible, then the presented algorithm will eliminate the duplication of effort and the only cycles to be computed are those that exist between agents. Lastly, we note that this algorithm is also incremental and allows for the cycle basis $\mathcal{B}^{ij}$ to be constructed as the inter-vehicle measurements arrive, resulting in fast update times due to the low complexity of shortest path operations.

Algorithm 1: Algorithm for finding a cycle basis incrementally for the connection of two graphs.
INPUT: $\mathcal{G}_i, \mathcal{G}_j, \mathcal{B}_{k-1}^{ij}, \mathcal{E}_{k-1}^{ij}, \mathcal{E}$.
OUTPUT: Updated Cycle Basis $\mathcal{B}_k^{ij}$.

function CONNECTEDCYCLEBASIS($\mathcal{G}_i, \mathcal{G}_j, \mathcal{E}_{k-1}^{ij}, \mathcal{E}$)
  $\mathcal{B}_k^{ij} \leftarrow \mathcal{B}_{k-1}^{ij}$
  for e_k in $\mathcal{E}$ **do**
    $e_{k-1} \leftarrow \mathcal{E}_{k-1}^{ij}[k-1]$
    $\mathcal{P}_i \leftarrow$ Shortest path between nodes a_k and a_{k-1}
    $\mathcal{P}_j \leftarrow$ Shortest path between nodes b_k and b_{k-1}
    $\mathcal{C} \leftarrow \mathcal{P}_i \cup e_{k-1} \cup \mathcal{P}_j \cup e_k$
    $\mathcal{B}_k^{ij} \leftarrow \mathcal{B}_k^{ij} \cup \mathcal{C}$
    $\mathcal{E}_{k-1}^{ij} \leftarrow \mathcal{E}_{k-1}^{ij} \cup e_k$
  end for
  $\mathcal{E}_k^{ij} \leftarrow \mathcal{E}_{k-1}^{ij}$
end function

C. Low-Degree-of-Freedom Measurements

One of the shortcomings of the relative parameterization of PGO is that it requires that the measurements between poses be a relative pose. This requirement arises because of the cycle constraints used in the optimization. However, measuring a relative pose between two poses can be difficult, especially in multi-agent scenarios, and often low-degree-of-freedom (DOF) measurements (range, bearing, etc.) are easier to obtain in real world scenarios. The cycle constraints currently used to constrain the relative PGO problem are not well suited to such measurements.

If low-DOF were to be used there would be several instances where it becomes unclear how to traverse a cycle. This is especially true in multi-agent scenarios when there is no single odometry backbone and two inter-agent measurements are required to form a cycle. For example, there is no intuitive way to traverse a cycle that contains two range measurements between different vehicles because there is not enough information present to do so. In such scenarios, the cycle constraints lose their usefulness, despite the fact that cycles still exist in the graph and the graph structure is unchanged.

Using low-DOF measurements requires a different approach. One possible approach is to use the low-DOF measurements to estimate a relative pose as done in [28], [29] and then use the relative pose in a cycle constraint. Another possible approach would be to formulate a different kind of constraint that would use n measurements, where n measurements would be sufficient to uniquely satisfy the constraint. While the number of measurements n is known for many measurement types, however, there is no concept or tool in graph theory, to our knowledge, that can aid in the generation of such a constraint. As such, we adopt the method of condensing low-DOF measurements into a single rigid-body transformation.

The remainder of this section will apply this method using inter-vehicle range and bearing measurements.

1) Range and Bearing: Estimating a relative pose from low-DOF measurements is a topic that has received some attention in the robotics community. In [29] Zhou shows that in three dimensions, the rotation between the two poses is unobservable for two range and bearing measurements because the rotation around a specific unit vector is undetermined. This can be overcome by using a third range and bearing measurement and

solving a system of linear equations for the rotation matrix. Another method assumes that each robot is equipped with an IMU. An onboard IMU makes the roll and pitch angles of each agent observable because the IMU is able to measure the gravity vector which, when combined with vehicle motion, allows the roll and pitch to be estimated accurately. Assuming the roll and pitch estimates are accurate, they can be treated as known constants, making the problem of estimating the relative rotation a two dimensional problem where the relative heading can be solved for uniquely.

We solve for the relative pose using optimization techniques, as opposed to the algebraic solutions in [29], to obtain greater accuracy in our relative pose estimates. The problem can be solved quickly given its small size. The optimization problem is defined as follows

$$T = \arg\min_T \sum_{i=1}^{n} \|z - h(T, T_a, T_b)\|^2_{\Sigma_k} \tag{2}$$

and $h(T, T_a, T_b)$ is defined as

$$h(T, T_a, T_b) = \begin{pmatrix} \|dt\| \\ \text{atan2}(dy, dx) \end{pmatrix} \tag{3}$$

where dt is the translation component of $(T_a^{-1}(T\,T_b))$ and dx and dy are the components of dt. We solve the problem using Ceres [30] assuming that T_a and T_b are known and constant. We can then use the transform T in the cycle constraints defined in (1). A minimum of two measurements are needed to solve (2) but more can be used depending on the quality of the measurements.

IV. Results

In this section we provide extensive validation of the algorithms described above. We outline the experiments used to evaluate all the algorithms presented in Section III and provide a discussion of their implications.

A. Incremental Algorithm Validation

Since no experimental validation of the incremental cycle basis algorithm in [26] has been done, we first perform experiments to compare the performance of the incremental and minimum cycle bases.

Cycle Basis Density: Our first experiment was designed to compare the sparsity of the incremental cycle basis with the MCB. This was deemed important since the use of the MCB was done to maximize the sparsity of the problem. We define a metric to measure the density as the following

$$\rho = \frac{\sum_{\mathcal{C}_i \in \mathcal{C}} |\mathcal{C}_i|}{|\mathcal{E}|}. \tag{4}$$

The metric sums the number of edges in each cycle in the cycle basis, which corresponds to the number of non-zero entries in the Jacobian, and normalizes it by the number of edges in the graph. The MCB will have the minimum achievable density. Our goal is to compare how well the incremental cycle basis approximates the density of the MCB. We provide the comparison on several standard benchmark datasets.

Fig. 2 contains a visualization of how the density of both cycle bases change as the graph grows for the M3500 dataset [20]

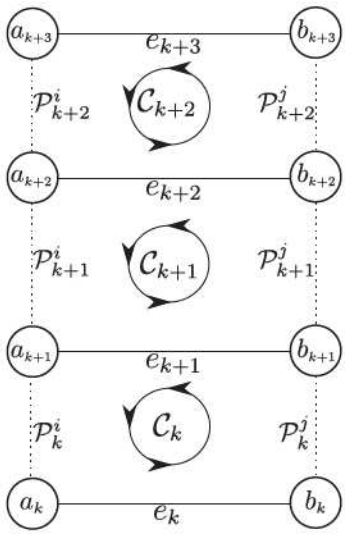

Fig. 1. A visual description of the cycles created in Algorithm 1. Dotted lines denote the shortest path between two nodes while solid lines denote the edge connecting two nodes.

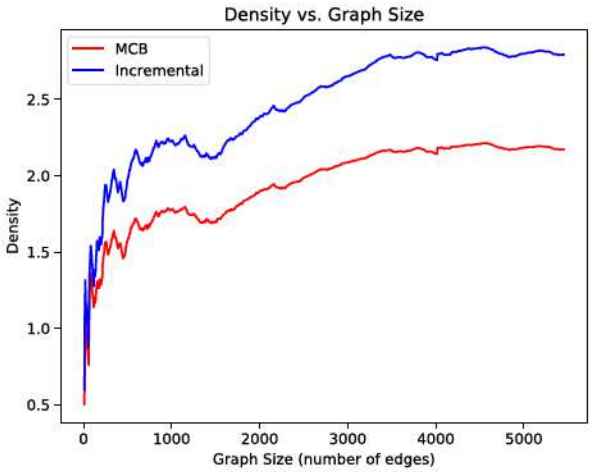

Fig. 2. Visual comparison of the density of the incremental cycle basis and the MCB as the graph grows for the M3500 dataset.

plotted against the number of edges in the graph. Table I contains the density for the entirety of each dataset found in [4], [5]. As is expected, the incremental algorithm produces a cycle basis that is denser than the MCB. Observing the values in Table I shows that the incremental algorithm produces a cycle basis that is about 1.3 times more dense than the MCB. However, many of these datasets (M3500, City10k, Torus1000, and Sphere2500) produce graphs that are more dense than those in many PGO applications. The incremental algorithm is capable of producing a cycle basis that has almost the same density of the MCB in sparser graphs, such as those in the MITb and Kitti datasets.

Algorithm Complexity: The complexity of both the incremental algorithm and the MCB algorithm in [19] were presented earlier. We provide additional validation on the runtime of each algorithm. We test the time that it takes for both the incremental algorithm and the MCB algorithm to update their respective cycle bases. Since no incremental update to an MCB exists we report the time required to recompute the cycle basis. We

TABLE I
DENSITY FOR ALL OF THE BENCHMARK DATASETS WHEN ALL DATA HAS BEEN USED

	INTEL	MITb	M3500	City10k	Torus10000	Sphere2500	Kitti
ICB	2.62	1.34	2.79	3.25	3.31	2.48	1.40
MCB	2.06	1.31	2.17	2.36	2.42	1.99	1.32

ICB indicates using the incremental cycle basis algorithm and **MCB** indicates the minimum cycle basis algorithm.

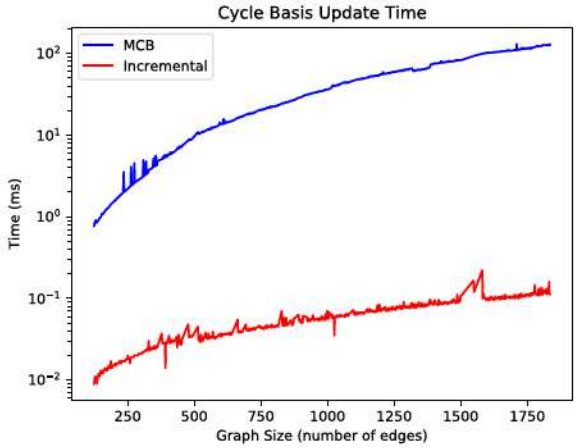

Fig. 3. Time required to update the cycle bases for the incremental algorithm and MCB algorithm on the INTEL dataset. *Note the log scale on the vertical axis.*

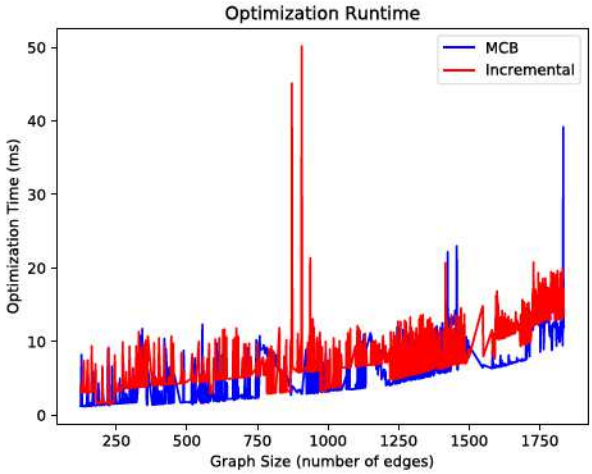

Fig. 4. Time required for the optimization to converge on the INTEL dataset.

then record the update time to be plotted against the number of edges in the graph. We also record the time required to solve the optimization problem in (1) to determine the effect of the increased cycle basis density on the optimization runtime. As with the sparsity experiment, we provide a comparison across multiple benchmark datasets.

Fig. 3 shows the time it takes to update the cycle bases on the INTEL dataset and Fig. 4 shows the time it takes for the optimization to reach a solution. Looking at Figs. 3 and 4 we see that the incremental algorithm can update a cycle basis in a time three orders of magnitude faster than the MCB algorithm while having comparable optimization speeds. Observing Table II shows a similar trend where the incremental cycle basis update is several orders of magnitude faster than the MCB calculation, but sacrifices comparatively little optimization speed despite the increased cycle basis density.

B. Multi-Agent Cycle Basis Validation

In this section we provide validation of the multi-agent incremental cycle basis (MA-ICB) algorithm in Algorithm 1. We first compare the density and cycle basis update times of Algorithm 1 with those of the algorithm in [26]. We also compare the error statistics of using the MCB and the cycle basis found in Algorithm 1 in both simulation and hardware experiments.

1) Performance Evaluation: We validate Algorithm 1 using the same density and algorithm complexity experiments presented in Section IV-A on the same datasets but with modifications to simulate multi-agent scenarios. Each dataset was split in half and the two halves were assumed to be collected simultaneously. We recomputed the density, using (4), after every inter-agent loop closure. We show the benefit of being able to parallelize Algorithm 1 by measuring the time to update the cycle basis for both algorithms and comparing them. Additionally, to show the value of recomputing inter-agent loops, we show the density after recomputing the inter-agent loops at the end of the simulation.

We present our results in Table III where we present the density at the end of the simulation, the average time to update the cycle basis, and the density of the cycle basis using Algorithm 1 when the inter-agent loops have been recomputed at the end of the simulation. As can be seen Algorithm 1 improves the update time by around 30 percent on most datasets. We note that there is significant variation of the density of the cycle basis produced by Algorithm 1 and this is primarily a result of the edge ordering we utilize in Algorithm 1. However, the density can be siginifcantly reduced by recomputing inter-agent loops as the graphs for each individual agent become more developed. In our experiment we recomputed all inter-agent loops and this usually required tens of milliseconds to complete. Further examination of the original loops produced by Algorithm 1 shows that a minority of the loops contribute to significant portion of the density. A more targeted recomputation strategy could reduce the time required while significantly decreasing the density by identifying and recomputing the longest inter-agent loops.

2) Simulation Experiment: Our simulation experiment was done to compare the error in both the edges and the trajectory with the ground truth data produced in the simulator. Our simulator generated 10 agents that operated in a Manhattan-world-like trajectory. Each robot has 500 odometry edges and 100 intra-vehicle loop closures. Between any given pair of robots there are 50 inter-vehicle loop closure measurements that are generated randomly. Noise was added to both the odometry and loop closure measurements using the noise characteristics of

TABLE II
TIME REQUIRED TO UPDATE THE CYCLE BASES AND RUN THE OPTIMIZATION WHEN FOR THE FOLLOWING DATASETS

		INTEL	MITb	M3500	City10k	Torus10000	Sphere2500	Kitti
Cycle Basis Update (ms)	**ICB**	0.057	0.076	0.14	0.59	0.81	0.13	0.17
	MCB	174	49.0	1896	2189	2702	1030	1865
Optimization (ms)	**ICB**	3.93	2.06	45.76	293	1668	234	6.56
	MCB	2.90	1.65	26.7	184	1277	240	7.86

ICB is using the incremental cycle basis and **MCB** is using the minimum cycle basis.

TABLE III
COMPARISON OF THE DENSITY AND AVERAGE UPDATE TIMES OF ALGORITHM 1 AND THE INCREMENTAL ALGORITHM IN [26]

		INTEL	M3500	City 10k	Torus10000	Sphere2500	Kitti
Average Update Time (ms)	**ICB**	0.026	0.122	0.40	0.43	0.13	0.39
	MA-ICB	0.039	0.082	0.29	0.27	0.052	0.19
Density	**ICB**	2.16	2.46	2.77	2.79	2.00	1.38
	MA-ICB	6.02	5.41	11.43	8.62	2.00	1.84
Recomputed Density	**MA-ICB**	2.52	2.62	3.32	3.38	2.00	1.40

TABLE IV
RESULTS COMPARING THE ERROR OF THE MCB AND THE MA-ICB ALGORITHM IN ALGORITHM 1

Algorithm	Rot. RMSE (deg)	Trans. RMSE (m)	ATE (m)
MCB	5.74	0.39	0.55
MA-ICB	5.74	0.39	0.56

the M3500 dataset in [4]. The optimization problem in (1) was solved using the cycle constraints from both the MCB and the cycle basis defined in Algorithm 1. After solving the optimization, we compute the root-mean-squared error (RMSE) of the edges, defined in (5), and the average trajectory error (ATE) of the poses, defined in (6), for all agents. The trans$(\cdot)$ operator in (6) indicates the translation component of the rigid-body transformation.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{N} ||\text{Log}(dT_i^{-1} d\hat{T}_i)||^2}{N}} \tag{5}$$

$$\text{ATE} = \sqrt{\frac{\sum_{i=1}^{N} ||\text{trans}(T_i^{-1} \hat{T}_i)||^2}{N}} \tag{6}$$

We report the RMSE of the translational components and the rotational components of the edges separately, and our results appear in Table IV. Observing Table IV we find that the two cycle basis algorithms produce nearly identical results, where the small differences can be attributed to different stopping points that satisfied the stopping criteria of the optimization. The time required to compute all the required cycles between graphs using Algorithm 1 was 118 ms while the time required using the MCB algorithm was 4170 ms, showing that we can achieve similar accuracy while using only a small fraction of the time to compute cycle constraints.

3) Hardware Experiment: In this section, we describe the experimental validation of the algorithms presented in Algorithm 1 and in [26]. The experiments are performed on Turtlebots carrying an Intel NUC processor on board running Ubuntu 18.04. The Robot Operating System is used to facilitate message passing between the sensors and agents. A motion capture system is used to generate edges every 0.5 meters or a change of 15 degrees in yaw as a stand-in for vision based alternatives like [31]. We test

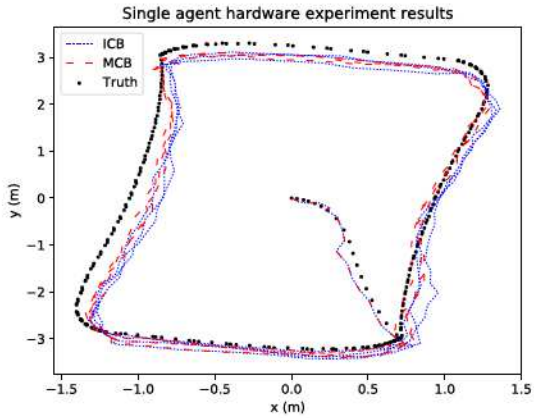

Fig. 5. Results of the single agent hardware experiment using a cycle basis described by Bai [26].

Bai's incremental algorithm in a single-agent scenario and our algorithm in a multi-agent experiment.

In the single-agent case, the ground robot follows a rectangular trajectory with feedback from the motion capture system to generate loop closures when the Turtlebot returned to a location it had previously been. The results of this experiment are shown in Fig. 5. The average RMS error for this experiment can be found in Table V. The algorithm took 1.46 ms to optimize over 210 edges using the MCB and 1.87 ms using the incremental cycle basis.

In the multi-agent scenario, three Turtlebots are used and follow rectangular trajectories. Each agent has its own processing unit and generates edges as described in the single-agent experiment. Periodically, agents will communicate with each other. During a communication, agents will share the data they have collected and obtain a relative pose between them. In this experiment, agents 0, 1 and 2 have 209, 217 and 193 edges, respectively, and 20 different communication events evenly distributed among pairs of robots. The results of this run are shown in Fig. 6 and show the results from the perspective of agent 0. The RMS error for this run when compared to the motion capture truth reference is shown in Table V, along with the ATE. We also compared the time to update the cycle bases on the last

TABLE V
ERROR RESULTS FOR BOTH THE SINGLE AND MULTI-AGENT HARDWARE EXPERIMENTS

Experiment	Algorithm	Rot. RMSE (deg)	Trans. RMSE(m)	ATE (m)
Single agent	MCB	2.65	0.070	0.19
	ICB	2.90	0.068	0.22
Multi-agent	MCB	0.63	0.024	0.052
	MA-ICB	0.63	0.023	0.033

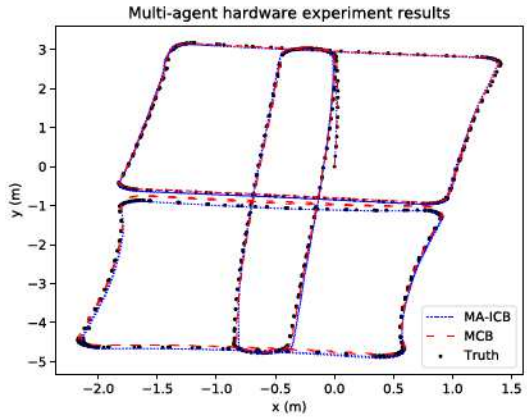

Fig. 6. Results of the multi-agent experiment using a cycle basis generated by Algorithm 1.

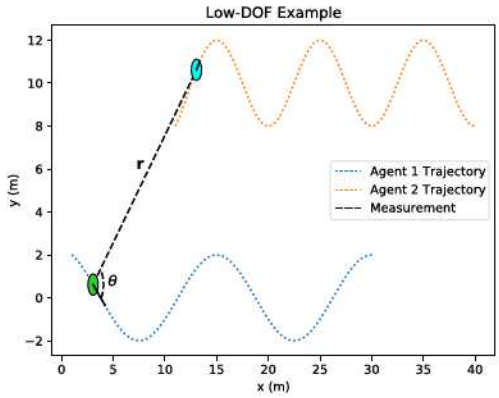

Fig. 7. Example trajectories for the two agents. Note that each agent travels in the direction it is facing. The range measurement $r = ||\mathbf{r}||$ and the bearing measurement is $\theta = \text{atan2}(\mathbf{r}_y, \mathbf{r}_x) - \theta_1$.

optimization. Updating the MCB took 49.6 ms, while updating the the cycle bases between all pairs of robots using Algorithm 1 required 0.66 ms.

Observing the results in Figs. 5, 6, and those in Table V show that choice of cycle basis has little impact on the accuracy obtained by the optimization. In combination with the results from Section IV-A, we show that we can obtain this accuracy and significantly lower the computational time, especially for large graphs, by using algorithms to approximate a MCB instead of exactly computing one.

C. Low-DOF Measurements

This experiment was designed to validate the use of low-DOF measurements in the relative parameterization of PGO. In this experiment, two agents were simulated in a planar environment traveling along sine waves of different frequencies and phase offsets. Odometry was simulated by taking two poses along the trajectory at times between 0.5 and 2.5 seconds apart, computing the transformation between them, and adding noise. This approach created between 75 and 95 poses per agent along a 120 s long simulation. 30 range and bearing measurements were computed at random intervals along the trajectories as shown in Fig. 7. These measurements were used to create inter-vehicle loop closures by solving (2) using pairs of sequential measurements. This resulted in 15 relative-pose measurements to be used in the cycle-based PGO problem. We ran 30 different trials to record and average the error in the relative localization between the agents. The noise characteristics used in the experiment are described below.

$$\Sigma_{\text{odom}} = \text{diag}(0.01\ \text{m}^2, 0.01\ \text{m}^2, 0.001\ \text{rad}^2)$$

$$\Sigma_{\text{meas}} = \text{diag}(0.1\ \text{m}^2, 0.01\ \text{rad}^2)$$

TABLE VI
RELATIVE POSE ERROR RESULTS FOR THE LOW-DOF MEASUREMENT EXPERIMENT

Time	Variable	Mean Relative Pose Error		Std. Deviation	
		CBPGO	GTSAM	CBPGO	GTSAM
0 (s)	x (m)	-0.0017	-0.12	0.84	1.39
	y (m)	0.019	0.024	0.94	1.40
	θ (deg)	-0.14	0.14	6.69	0.86
120 (s)	x (m)	0.010	-0.022	1.37	1.51
	y(m)	-0.78	-0.29	1.38	2.41
	θ (deg)	-3.02	-0.68	13.6	1.00

Since the purpose of multi-agent PGO is for the agents to localize relative to one another, we evaluate the error in the relative pose estimates at the start and the end of the trajectory. We compute the average and standard deviation of the error across the different trials and report the results in Table VI. We compare the results of our method with the results produced by GTSAM [13] when range and bearing factors are used to model inter-agent measurements.

As can be seen, we obtain accurate relative-pose estimates with the average error between the two agents being less than a meter in the translational components and less than three degrees in the relative orientation showing that low-DOF measurements can effectively be used in the cycle-based PGO problem described in (1). Additionally, our method obtains similar translational accuracy when compared with GTSAM but suffers both lower relative heading accuracy and larger variance on the relative heading error. This degredation in the relative heading is expected as we do not recompute the relative pose factors by solving (2) using updated estimates for T_a and T_b. The accuracy in the relative heading factors, T, are dependent on the quality of the odometry of each robot and the errors in the measurements.

V. Conclusion

In summary, we have validated a proposed but untested algorithm to incrementally compute a sparse cycle basis and shown that the incremental cycle basis closely approximates the MCB. Additionally, we presented and validated an algorithm to approximate the MCB of the sparse connection of two disjoint graphs and demonstrated its ability to run in real-time PGO scenarios and give accurate estimates of the edges in each robot's graph. Lastly, we have introduced a methodology to utilize low degree of freedom measurements in a relative PGO framework and demonstrated that the method produces accurate relative-pose measurements between the different agents. These improvements take steps toward allowing relative PGO to be efficiently used in multi-agent scenarios where obtaining a relative-pose measurement between agents can be difficult.

Acknowledgment

The authors would like to thank Paul Buzaud and Jared Paquet for their help setting up the hardware for the experiments in this letter. The authors would also like to thank the AFRL Munitions Directorate and other C-UAS industry members for their significant contribution.

References

[1] D. O. Wheeler, D. P. Koch, J. S. Jackson, T. W. McLain, and R. W. Beard, "Relative navigation: A keyframe-based approach for observable GPS-degraded navigation," *IEEE Control Syst. Mag.*, vol. 38, no. 4, pp. 30–48, Aug. 2018.

[2] G. Ellingson, K. Brink, and T. McLain, "Cooperative relative navigation of multiple aircraft in global positioning system-denied/degraded environments," *J. Aerosp. Inf. Syst.*, vol. 17, no. 8, pp. 470–480, 2020.

[3] F. Lu and E. Milios, "Globally consistent range scan alignment for environment mapping," *Auton. Robots*, vol. 4, no. 4, pp. 333–349, 1997.

[4] L. Carlone and A. Censi, "From angular manifolds to the integer lattice: Guaranteed orientation estimation with application to pose graph optimization," *IEEE Trans. Robot.*, vol. 30, no. 2, pp. 475–492, Apr. 2014.

[5] L. Carlone, R. Tron, K. Daniilidis, and F. Dellaert, "Initialization techniques for 3D SLAM: A survey on rotation estimation and its use in pose graph optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 4597–4604.

[6] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, "Robust map optimization using dynamic covariance scaling," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 62–69.

[7] I. Aloise and G. Grisetti, "Chordal based error function for 3-D pose-graph optimization," *IEEE Robot. Automat. Lett.*, vol. 5, no. 1, pp. 274–281, Jan. 2020.

[8] M. Liu, S. Huang, G. Dissanayake, and H. Wang, "A convex optimization based approach for pose SLAM problems," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 1898–1903.

[9] D. M. Rosen, C. DuHadway, and J. J. Leonard, "A convex relaxation for approximate global optimization in simultaneous localization and mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 5822–5829.

[10] D. M. Rosen, "Scalable low-rank semidefinite programming for certifiably correct machine perception," in *Proc. Int. Workshop Algorithmic Found. Robot.*, 2020, pp. 551–566.

[11] D. M. Rosen, L. Carlone, A. S. Bandeira, and J. J. Leonard, "SE-Sync: A certifiably correct algorithm for synchronization over the special euclidean group," *Int. J. Robot. Res.*, vol. 38, no. 2/3, pp. 95–125, 2019.

[12] J. G. Mangelson, J. Liu, R. M. Eustice, and R. Vasudevan, "Guaranteed globally optimal planar pose graph and landmark SLAM via sparse-bounded sums-of-squares programming," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 9306–9312.

[13] F. Dellaert, "Factor graphs and GTSAM: A hands-on introduction," Georgia Inst. Technol., Atlanta, GA, USA, Tech. Rep. GT-RIM-CP&R-2012-002, 2012.

[14] G. Grisetti, R. Kümmerle, H. Strasdat, and K. Konolige, "g2o: A general framework for (hyper) graph optimization," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 9–13.

[15] P. Agarwal and E. Olson, "Variable reordering strategies for SLAM," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 3844–3850.

[16] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 3281–3288.

[17] F. Bai, T. Vidal-Calleja, and S. Huang, "Robust incremental SLAM under constrained optimization formulation," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 1207–1214, Apr. 2018.

[18] J. Jackson, K. Brink, B. Forsgren, D. Wheeler, and T. McLain, "Direct relative edge optimization, a robust alternative for pose graph optimization," *IEEE Robot. Automat. Lett.*, vol. 4, no. 2, pp. 1932–1939, Apr. 2019.

[19] F. Bai, T. Vidal-Calleja, and G. Grisetti, "Sparse pose graph optimization in cycle space," *IEEE Trans. Robot.*, vol. 37, no. 5, pp. 1381–1400, Oct. 2021.

[20] S. Teller, E. Olson, and J. Leonard, "Fast iterative optimization of pose graphs with poor inital estimates," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2006, pp. 2262–2269.

[21] J. D. Horton, "A polynomial-time algorithm to find the shortest cycle basis of a graph," *SIAM J. Comput.*, vol. 16, no. 2, pp. 358–366, 1987.

[22] E. Amaldi, C. Iuliano, T. Jurkiewicz, K. Mehlhorn, and R. Rizzi, "Breaking the $O(m^2n)$ barrier for minimum cycle bases," in *Proc. Eur. Symp. Algorithms*, 2009, pp. 301–312.

[23] T. Kavitha, K. Mehlhorn, and D. Michail, "New approximation algorithms for minimum cycle bases of graphs," *Algorithmica*, vol. 59, no. 4, pp. 471–488, 2011.

[24] K. Mehlhorn and D. Michail, "Minimum cycle bases: Faster and simpler," *ACM Trans. Algorithms*, vol. 6, no. 1, pp. 1–13, 2009.

[25] A. Rathod, "Fast algorithms for minimum cycle basis and minimum homology basis," 2021, *arXiv:2109.04567*.

[26] F. Bai, "Two novel techniques for graph optimization—Cycle based formulation and change of optimal values," Ph.D. dissertation, Univ. Technol. Sydney, Ultimo, NSW, Australia, 2020.

[27] E. Zare and M. Rashtchi, "On the cycle basis join of two graphs," *Int. J. Algebra*, vol. 5, no. 22, pp. 1059–1063, 2011.

[28] N. Trawny, X. S. Zhou, K. Zhou, and S. I. Roumeliotis, "Interrobot transformations in 3-D," *IEEE Trans. Robot.*, vol. 26, no. 2, pp. 226–243, Apr. 2010.

[29] X. S. Zhou and S. I. Roumeliotis, "Determining 3-D relative transformations for any combination of range and bearing measurements," *IEEE Trans. Robot.*, vol. 29, no. 2, pp. 458–474, Apr. 2013.

[30] S. Agarwal et al., "Ceres solver," 2014. [Online]. Available: http://ceres-solver.org

[31] D. P. Koch, D. O. Wheeler, R. W. Beard, T. W. McLain, and K. M. Brink, "Relative multiplicative extended Kalman filter for observable GPS-denied navigation," *Int. J. Robot. Res.*, vol. 39, no. 9, pp. 1085–1121, 2020.