Article

The International Journal of
Robotics Research
1–23
© The Author(s) 2015
Reprints and permissions:
sagepub.co.uk/journalsPermissions.nav
DOI: 10.1177/0278364915596588
ijr.sagepub.com
SAGE

Precise assembly of 3D truss structures using MLE-based error prediction and correction

Erik Komendera and Nikolaus Correll

Abstract

We describe a method to construct precise truss structures from non-precise commodity parts. Trusses with precision in the order of micrometers, such as the truss of a space telescope, have previously been built using precisely machined truss connection systems. This approach is expensive, heavy, and prone to failure, e.g. when a single element is lost. In the past, we have proposed a novel concept in which non-precise commodity parts can be aligned using intelligent precision jigging robots and then welded in place. Even when using highly precise sensors and actuators, this approach can still lead to errors due to thermal expansion and structural deformation. In this paper, we describe and evaluate algorithms for generating truss assembly sequences that reduce the expected error by (1) using a heuristic to generate build sequences that reduce the expected variance, and (2) during assembly, estimating the structure's pose using maximum likelihood estimation that combines local measurements by different intelligent precision jigging robots, improves this estimate during loop closures in the construction process, and uses this estimate to correct for errors during construction. We show through simulation and physical experiment that this combined approach reduces assembly error, enabling precision construction with commodity materials. While the model herein is based on truss structures, the proposed methods generalize to a larger class of incremental assembly problems, which exhibit continuous rather than discrete errors.

Keyword

Robotic assembly, distributed robotics, probabilistic robotics, maximum likelihood estimation, assembly sequence planning

1. Introduction

While high-precision assembly and welding have long been staples of factory automation, precise assembly in the field remains a difficult challenge. Industrial robots such as pick-and-place machines can achieve high precision with pre-programmed motions, and are a critical component of modern industry, but do not function outside of their carefully controlled workspaces. Research into field assembly often focuses on self-correcting, interlocking components, freeing the robots from the task of ensuring accuracy, but this does not address the use of commodity materials and on-line adjustments.

Large scale construction projects do not rely solely on self-correcting, interlocking components, and rigid bodies are not valid assumptions. Instead, it is common to cut or bend parts from commodity materials as needed, often incorporating the state of the structure (including any position errors). Likewise, welding and other thermal processes induce stresses on structures, which change the outcome slightly. If robots assembling the structure were to ignore error propagation, eventually error would accumulate, leading to a structure with such large error that significant repairs are needed, such as inducing forces on the structure to "jam" in parts, or disassembling and trying again.

Assembling structures in space (Zimpfer et al., 2005) has the potential to overcome payload limitations of earth-based missions, thereby enabling the manufacturing of scalable structures. Space telescopes that could be assembled on orbit are a high priority for NASA (Watson et al., 2002) and the space industry (Hoyt et al., 2014), with proposed diameters of tens of meters up to hundreds of meters; in contrast, the 6.5 m reflector on the James Webb Space Telescope is the upper limit for deployable telescopes

Department of Computer Science, University of Colorado at Boulder, Boulder, USA

Corresponding author:
Erik Komedera, Department of Computer Science, University of Colorado at Boulder, 430 UCB, Boulder, CO 80309, USA.
Email: erik.e.komendera@nasa.gov

(Dorsey et al., 2012). Many of the benefits and techniques for assembling large space observatories are discussed in Lillie (2006). One robotic telescope assembly experiment (Doggett, 2002) demonstrated the repeated assembly and disassembly of an 8 m telescope mirror, composed of 102 precisely machined truss members and 12 hexagonal panels, using an industrial manipulator arm and a rotating assembly platform. However, this approach for assembling large-scale telescope truss structures and systems in space has been perceived as very costly because they require high precision and custom components that rely on mechanical connections, increased launch mass, and the potential for critical failure if only one of the components is missing or defective (Dorsey et al., 2012). Space telescope optical benches are trusses that require struts with varying lengths due to the paraboloid curvature of the reflectors mounted to them. Pre-manufacturing these struts on the ground is an expensive process that cannot be automated due to the uniqueness of each strut and node, and the failure of any part could jeopardize the entire assembly. With commodity materials, these requirements can be avoided and the loss of an individual piece does not jeopardize the entire assembly.

1.1. Intelligent precision jigging robots

Intelligent Precision Jigging Robots (IPJRs), introduced in Dorsey et al. (2012) and explored further by Komendera et al. (2014a,b) and Komendera and Correll (2014), are robots that can precisely hold a pose between disconnected parts (in welding, this is known as *jigging*), enabling an external agent (such as a robotic arm) to permanently bond the parts. Figure 1 shows a 2D IPJR (top left) that is operated by a robotic arm, and a set of 3D IPJRs that require manual operation. IPJRs adjust the distances between the parts using highly precise actuators and precise sensors for distance measurement. IPJRs free the external agents from managing the parts in addition to bonding, reducing the complexity of the external agent. A set of IPJRs and external agents performing assembly are individually simpler than an all-purpose assembly robot, thus they can be made more inexpensively, enabling robustness through redundancy.

The IPJRs used in this paper are shown in Figure 1, bottom left, and are single-degree-of-freedom robots whose only purpose is to set the distance between truss nodes while an external agent fixes a strut between the nodes. A group of 3, at a minimum, is required to assemble rigid truss structures one tetrahedral cell at a time by attaching the apex to a previously built triangular base. This is a simplification of previous models, in which a single IPJR robot contains multiple such actuators in a prebuilt triangular unit for constructing triangular cells in two dimensions. The IPJR described in Komendera et al. (2014a) is shown in Figure 1, top left, and used linear stepper motors to adjust lengths to a precision of 8 microns, and laser sensors to measure the length to a precision of 1 micron. Although the 2D IPJRs were able to assemble structures with accuracies that far exceeded that of the materials and processes used (glue gun and wooden dowels in Komendera et al. (2014b), and spot welding of titanium rods facilitated by a large-scale robotic manipulator in Komendera et al. (2014a)), a fundamental problem that remained was bias due to thermal expansion or structural deformation. Based on these experiences, we believe that precise hardware can greatly benefit from state estimation and error correction during assembly.

1.2. Motivating example

Space telescopes require truss elements with variable geometries that serve as an "optical bench" for the paraboloid telescope mirror. As such, they are a suitable target for the demonstration of the IPJR paradigm. The work in this paper is motivated by the 84-node telescope truss design described in Watson et al. (2002) and shown in Figure 1, bottom right. The accuracy requirements for such an optical bench are in the order of micrometers for those nodes that lie on the convex inner surface of the paraboloid. Although the accuracy of any other node is much less important in this specific application, the fact that all nodes are connected via trusses and error therefore propagates, essentially makes the accuracy requirements the same for all nodes in a structure.

In this paper, we assume that the assembly of truss structures proceeds one node at a time, starting from a predefined origin node, and working incrementally until the entire structure is finished. For each new node, at least as many struts as there are nodal degrees of freedom must be attached to the structure simultaneously to guarantee a unique position for the node; any fewer, and the truss is a mechanism. In three dimensions, we therefore require exactly 3 IPJRs to guarantee a unique node position.

1.3. Contribution and outline of this paper

We consider the problem of accurate incremental assembly of truss structures made of commodity materials, with high accuracy and precision requirements on the truss nodes, and in which the assembling agents are far smaller than the structure itself, and do not have the benefit of a global sensor to monitor the assembly process. We assume trusses to be constructed incrementally, adding one node at a time, using only node-to-node measurements that lead to the accumulation of error. Incremental truss assembly and the error model are formally defined in Section 2. We introduce a two-step method that allows precise construction of trusses using commodity components and imprecise attachment methods, such as welding. The first step is to generate a sequence that has a low covariance when assembled in open loop (Section 3). The second step is to combine multiple measurements of the same structure using maximum likelihood estimation (MLE) and use this improved estimate for error correction, which we show to reduce the covariance further (Section 4). We demonstrate this by simulation and physical experiments using IPJRs, described in Section 5, to assemble 3D telescope trusses of various

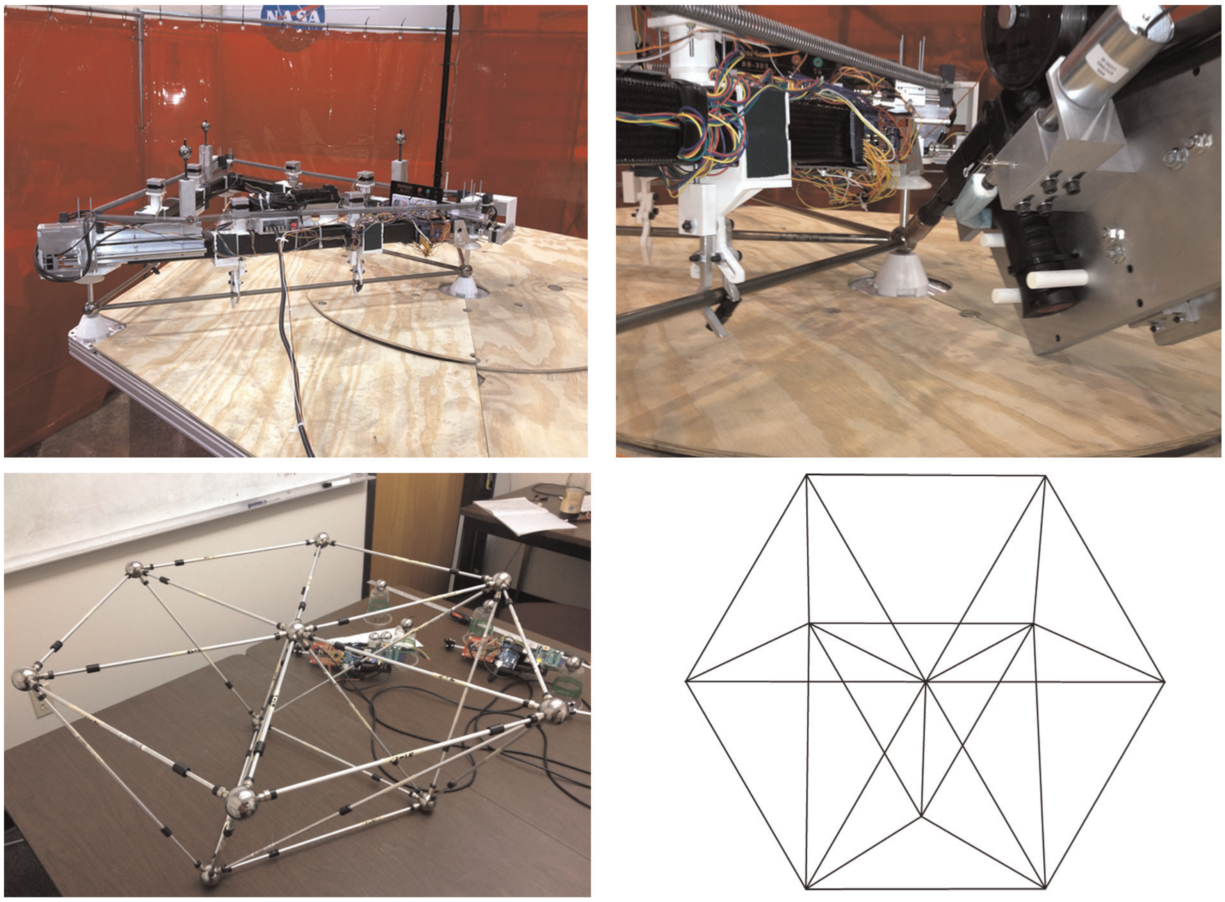


Fig. 1. Top left: The Intelligent Precision Jigging Robot (IPJR) from Komendera et al. (2014a) used Ultra-Motion linear actuators with 8μm step resolution, and had onboard length sensing using a KEYENCE laser sensor with 1μm resolution. The external manipulator shown at the left, known as the Lightweight Surface Manipulation System (LSMS), positioned the IPJR onto the site of the next cell to be assembled, and also positioned the IPJR over a strut canister to grasp new struts (not shown). Top right: the LSMS also welded the stock titanium struts to titanium node balls. Bottom left: a 10-node telescope truss assembled by the IPJRs presented in this paper and in Komendera and Correll (2014). Bottom right: a wireframe model of the telescope truss prototype assembled in Watson et al. (2002), assembled by simulated extravehicular activity using precisely machined struts and nodes.

sizes (Section 6). While the theory and algorithms in this paper are tailored to the incremental truss assembly problem and local-only measurements, the proposed concepts should extend to any incremental assembly problem without extensive reworking, which we discuss in Section 7.

This paper extends upon the results presented in Komendera and Correll (2014) by showing that an MLE scheme can further improve accuracy of the estimation problem, and thereby accuracy of the assembly, when compared with the Extended Kalman Filter (EKF) that was used as estimation framework in Komendera and Correll (2014).

1.4. Related work

Algorithms for mechanical assembly planning (via disassembly sequences) for problems in well-known environments such as assembly lines, and with few assembly robots, were explored in the 80s and 90s, resulting in algorithms for finding fully-ordered sequences (DeFazio and Whitney, 1987; Homem de Mello and Lee, 1991; Lee and Moradi, 1999; Röhrdanz et al., 1996; Wilson, 1992; Wolter, 1989), or using opportunistic assembly planning (Fox and Kempf, 1985) when necessary. An overview of the challenges in mechanical assembly planning can be found in Gottschlich et al. (1994). All of these approaches attempted to find a suitable order for assembly that avoids construction deadlock, and assumed rigid bodies. In reality, structural forces and minor assembly errors often lead to situations where parts of a structure must be forced open to jam a new component in. Such algorithms do not consider these situations and do not model the potential for slight misalignments.

Assembly of structures is an important application for multi-robot systems (Wawerla et al., 2002). Quadrotor assembly is a recent and popular development; cubic truss structures are assembled in Lindsey and Kumar (2013), a quadrotor assembly agent uses reinforcement learning to learn to handle unmodeled situations in Barros dos Santos et al. (2013), and Jimenez-Cano et al. (2013) demonstrates control with a manipulator arm attached. Other approaches include truss climbing and assembling robots (Detweiler et al., 2007), termite-inspired swarm assembly robots (Werfel et al., 2014), and a robot team that can build IKEA furniture in cluttered environments (Knepper et al., 2013). Mobile assemblers (Worcester et al., 2012) used visual feedback to estimate and correct errors in assembly. The equal distribution and parallelization of the truss assembly task is shown and demonstrated in Bolger et al. (2010) and Yun et al. (2011). With some exceptions such as amorphous assembly in Napp and Nagpal (2014), any of these methods rely on self-correcting, interlocking mechanisms, which would add extra mass and expense due to machining requirements, and we know of no such experiment that considers welding or cutting. Some distributed large-scale assembly tasks have used common construction materials instead of mock materials that were modified to simplify the task. Three different robots are shown to successfully dock a part to an assembly (Simmons et al., 2001). An experiment performed at NASA's Jet Propulsion Laboratory demonstrated the precision assembly of beams by a pair of cooperative robots using highly rigid motions to ensure precision (Stroupe et al., 2005a,b). A robust algorithm is described in Heger (2010); Heger and Singh (2010), which uses teams of robots to assemble structures while handling exceptions due to a wide range of failures, and relying only on a human operator when failures are beyond the scope of the assembly robots. Taking into account physical constraints such as structural stability and material properties into the build order has been shown in McEvoy et al. (2014). A major hurdle to overcome is precise assembly by mobile manipulators; peg-in-hole tasks are explored in Hamner et al. (2010) and Dogar et al. (2014).

Algorithms for distributed assembly are often simple and are only applicable to the specific class of structures under consideration, and include modeling the assembly process as a differential equation (Berman et al., 2009), using raster scanning techniques (Lindsey and Kumar, 2013), or distributing the assembly of structures layer-by-layer through Voronoi diagrams (Yun, 2010). Finding assembly sequences by reversing the disassembly sequences is used in many approaches, since disassembly sequencing with only geometric constraints does not require backtracking (Homem de Mello and Lee, 1991). AND/OR graphs (Homem de Mello and Lee, 1991) assume that subassemblies can be made independently. Complete and correct algorithms cannot escape the worst case $O(n!)$ for enumerating all build sequences (Homem de Mello and Sanderson, 1991), rendering these algorithms useful only for structures with tens of parts, and making this problem NP-Hard. Partial ordering (DeFazio and Whitney, 1987) may be used reduce the search space for assemblies, but is still exponential in the worst case. A* search with pruning (Wolter, 1989) is shown to find optimal paths subject to identifying good heuristics. Non-directional blocking graphs (Wilson, 1992) can find assemblies in polynomial time where assembly steps are valid if they can be moved into place without violating constraints. The difficulty of solving this problem made it fall out of fashion, but interest was renewed with the popularity of distributed robotics and stochastic methods, including genetic algorithms (Lazzerini and Marcelloni, 2000), and probabilistic roadmaps (Sundaram et al., 2001). A robust distributed algorithm based on opportunistic disassembly sequencing is described in (Heger, 2010), which uses teams of robots to assemble structures while handling exceptions due to a wide range of failures, and relying on a human operator only when failures are beyond the scope of the assembly robots.

While the IPJR concept used in this paper is moved around the structure by humans, and the IPJR prototype in Komendera et al. (2014a) by a large robotic crane, both IPJRs, welding robots and materials could be transported by climbing robots, which has been extensively studied on trusses (Bonani et al., 2009; Chu et al., 2010; Nigl et al., 2013; Vona et al., 2008; Yun et al., 2009) and unstructured terrain (Bretl et al., 2006).

2. Definitions

Both the assembly sequencing algorithm and the error reduction algorithm require a truss model that defines both the topology and the geometry of both the truss structure and the estimated state of the partially completed structure. A truss structure is an undirected graph $G=(V, E)$ embedded in Euclidean space. Each vertex $i \in V$ has a mapping $V \rightarrow \mathbb{R}^3$ to a structural *node* X_i with a 3-dimensional position $(x_i, y_i, z_i)^{\mathrm{T}}$. The nominal position for a node is $\bar{X}_i$, and the error is the difference $X_i - \bar{X}_i$. The number of nodes in the structure, and the size of V, is N. As nodes are spherical and struts can be attached anywhere, the orientation of each node is not important. Each edge $E_{ij}=\{i, j\}$, is associated with two vertices and corresponds to a *strut* of the structure. The length of the strut between two nodes i and j is $||X_i - X_j||$. The number of edges is $||E|| = O(N)$, and truss structures are *sparse*. If $D_v N = E$ is the expression for the number of edges, D_v is the *vertex half-degree*, or half the average number of struts touching a node. We chose to model the geometry of the telescope trusses in a global Cartesian coordinate system, for two reasons: the upper surface of the truss must conform to a globally defined paraboloid surface, and furthermore, this surface is not symmetric, eliminating any advantages that may be used by a different basis system, e. g. polar coordinates.

The *starting triangle* is the name given to the first three nodes assembled. The orthonormal basis for the structure is defined by the position of the starting triangle; the first node is fixed to the origin, the second node varies only on the X-axis with $y, z = 0$, and the third node varies only on the XY-plane with $z = 0$. This fixes the frame of reference to the structure itself, and does not require an external frame. In the algorithms, the starting triangle is denoted Δ.

The complete structure description is $\mathbf{X} \in \mathbb{R}^{3N-6}$, in which the -6 term corresponds to the reduced degrees of freedom of the starting triangle. The nominal structure, which is known prior to assembly, is $\overline{\mathbf{X}}$.

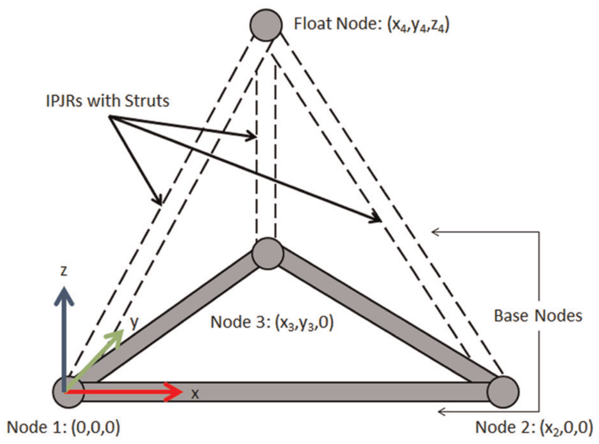


Fig. 2. Example of the placement of the fourth node on a base consisting of three nodes. The base nodes and struts, on the bottom, have been bonded. IPJRs, attached to struts and the base nodes, position the float node. The dotted lines indicate that bonding has not yet taken place, and the IPJRs are free to adjust their own lengths. Additionally, the initial three nodes are the starting triangle, and form the global origin of the structure.

IPJRs assemble trusses by attaching new nodes to nodes already on the structure. We call the new node the *floating node* to indicate that it is not fixed until bonding is performed, and the nodes to which the floating node is attached are the *base triple*, in which each node is a *base node*. In this paper, the floating node is indexed as f, and the base nodes $(i, j, k) = B_f$. In the model, one node is attached at a time by using exactly as many IPJRs as the node has degrees of freedom: one IPJR for one dimension (reserved only for the X-axis node) two IPJRs for two dimensions (the XY-plane node), and three IPJRs for three dimensions (all other nodes). All of the aforementioned concepts are shown in the diagram in Figure 2. In the specific case of tetrahedra, the distances between base nodes i, j, k and floating node f are the lengths that the IPJRs must satisfy, and are found as

$$\begin{aligned} L_{i,f} &= ||X_i - X_f|| \\ L_{j,f} &= ||X_j - X_f|| \\ L_{k,f} &= ||X_k - X_f|| \\ X_f &= \mathcal{F}_f(X_i, X_j, X_k, L_{i,f}, L_{j,f}, L_{k,f}) \end{aligned} \tag{1}$$

The function $\mathcal{F}_f()$ can be found by algebraically solving the system for X_f, a solution of which is provided in the appendix.

Equation 1 has two solutions based on the ordering of i, j, k, each a mirror image of the other with respect to the base. Determining which is correct is determined by the right hand rule: the path from i to k is counterclockwise, and the position of f is in the direction of the rotation vector defined by the path using the right hand rule. Thus, B_F is an ordered triple.

Using more IPJRs than is required results in an overdefined system, which in the physical world would result in large stresses on the structure unless the extra struts passively adjusted their lengths. All truss structures require $3N - 6$ struts to be stable. When a node has more adjacent nodes than the base nodes used for assembly, the extra struts (and the structure itself) are *redundant*. IPJRs also attach to redundant struts, but passively allow the struts to adjust since their lengths are a function of the assembly struts' lengths.

Choosing the best base nodes for each floating node is the key to solving the error minimization problem. The edges in E that are actively set by IPJRs are labeled E_A, and the redundant struts are labeled E_R

$$\begin{aligned} E_A &\subseteq E \\ \|E_A\| &= 3N - 6 \\ E_R &= E \setminus E_A \end{aligned} \tag{2}$$

The full structure $\mathbf{X}$ is a vector containing all of the node positions. The length of the vector is the same as the length of the actively assembled struts, $3N - 6$, which is also the total degrees of freedom in the structure

$$\mathbf{X} = (x_2, x_3, y_3, x_4, y_4, z_4 \ldots x_N, y_N, z_N) \tag{3}$$

The full vector of lengths L_E contains the positive, real-valued lengths for the assembly struts E_A, and is ordered in the assembly order

$$\begin{aligned} L_E &= (L_{2,1} \ldots L_{last}) \\ \|L_E\| &= 3N - 6 \end{aligned} \tag{4}$$

Prior to the completion of the structure, a substructure with K nodes is denoted V_K, E_K. E_K also includes passive struts, where $E_K \subset (E_A \cup E_R)$.

2.1. Assembly sequences

An *assembly sequence* A is an N-tuple consisting of pairs of assembly *steps*; each pair is defined as $\langle f, B_f \rangle$. A sample assembly sequence may be

$$A = [\langle 1, () \rangle, \langle 2, (1) \rangle, \langle 3, (1,2) \rangle, \langle 4, (1,2,3) \rangle, \langle 5, (2,3,4) \rangle \ldots \langle f, (i,j,k) \rangle] \tag{5}$$

Although we model A as a sequence, it is possible to define it as a set, since the assembly order is implicit in each individual step. If A is a complete structure, a partially complete structure A_{made} is a subsequence whose first element is the first element in A.

If two nodes i, j can be added to a structure independently of each other, then their relative ordering is irrelevant. For example, if an assembly sequence produces two branches, nodes added to one branch can be added independently from nodes on the other branch. This allows for parallelization: multiple groups of IPJRs could work on separate branches of structures independently. If one assumes all independent additions to a structure are made simultaneously, and each addition takes one unit of time, a parallel sequence can be defined, in which each floating node f is assigned a value t_f, where t_f is the earliest *time* at which the node f can be assembled, and is also the minimum number of nodes that need to be assembled before it can be assembled (the ancestor count) plus one

$$t_f = 1 + \max\,(t_i : i \in B_f) \tag{6}$$

In other words, a floating node can only be assembled after the last node of its base triple has been assembled. The starting triangle is assembled at $t = 1$, 2, 3; the nodes whose base is the starting triangle can be assembled at at $t = 4$, and so on. A can be partitioned into a *parallel assembly sequence* P by t values, each one corresponding to an assembly layer

$$P = (A_{t=1}, A_{t=2} \ldots) \tag{7}$$

Two assembly sequences with different but equivalent permutations can be compared by their parallel orders

$$A_i = A_j \text{ if } P_i = P_j \tag{8}$$

An assembly order A_i can be assembled by parallel IPJRs in less time than A_j if $\|P_i\| < \|P_j\|$. A *stepwise-minimal assembly order* for a starting triangle Δ is an $A_{minsteps}$ such that $\|P_{minsteps}\| \leq \|P_{all}\|$ for Δ. A *central triangle* is a triangle $\Delta_{central}$ such that $\left\|P_{\Delta_{central}, minsteps}\right\| \leq \left\|P_{\Delta_i, minsteps}\right\|$ for Δ_i — that is to say, the distance between a central triangle and the most distant node in step count is less than the distance from other starting triangles to their respective most distant nodes.

2.2. Measurement model

The IPJR model makes few assumptions about what kinds of measurements are available. While global position sensors, cameras, and other sensors will likely be used in an on-orbit experiment, they are not necessary, as we show here. Instead, we assume that each IPJR has the capability to measure the length of the strut it is currently attached to (pre- or post-weld). They include measurements of both the assembly struts and the passive struts, as long as the IPJRs are attached. Measurements $M_{i,j}$, between nodes i and j, are taken after welding is complete and X_i, X_j are permanent. We assume the error to be normally distributed with mean $\|X_i - X_j\|$, and *measurement variance* σ_M^2

$$M_{i,j} \sim \mathcal{N}\left(\left\|X_i - X_j\right\|, \sigma_M^2\right) \tag{9}$$

An IPJR builds up a map of its surroundings with local measurements, including its own position (expressed as the two nodes it is connected to). When redundant struts are added between a floating node and extra nodes in the structure not in the base triple, IPJRs passively adjust. These IPJRs are assumed to be capable of measuring the passive struts after they have been welded. Thus, the set of measurements M defines the full set of edges E for the substructure of size K

$$M_K = \left\{\mathcal{N}\left(\left\|X_i - X_j\right\|, \sigma_M^2\right) : (i,j) \in E_K\right\} \tag{10}$$

The extra measurements of the redundant edge set E_R enhance the effect of "loop closure" in the MLE algorithm that follows. Without these extra measurements, the MLE algorithm finds the structure that independently maximizes the likelihood of each strut based on only its own measurements; this is because the number of measurements is equal to the number of degrees of freedom. The extra measurements force an overdefined system, where each strut estimate is a function of all of the other struts' noises and measurement noises. A commonly chosen analogy is that of a spring-mass system, in which springs are struts and masses are nodes, as seen in Thrun and Montemerlo (2006). If the system has as many degrees of freedom as there are springs, the springs will settle on their equilibrium lengths. However, redundant springs (measurements) between previously disconnected nodes will induce a geometry change in the spring mass system. Precise measurements can be thought of as springs with high stiffness in the system and will have a greater effect on the estimate than the imprecise priors.

2.3. Truss probability model

For a space telescope optical bench, the nodes on which the reflectors will be mounted must be positioned very accurately in relation to a fixed geometry matching the optical properties of the mirror. For example, in the motivating 84-node truss, the reflector is paraboloid, and attachment points are offset from the paraboloid. Since the accuracies of the attachment nodes are dependent on the accuracies of the remaining nodes, we consider the position of each node, attachment or otherwise, to be of equal importance.

Errors in the truss structure arise from errors in extending and contracting the IPJRs. These errors build up over the assembly; distant nodes with long chains of ancestors are far more likely to have larger covariances. The truss probability model is based on expected positional errors of the nodes, assuming that no onboard error detection and correction is used when assembling. This model is referred to as the *open loop* model. In this section, we describe the probability model and the reasoning for the using the covariance trace as the error metric.

To derive an expression for the covariance, the following assumptions are made: IPJR errors are small relative to the structure, and the length of each strut is a Gaussian noise variable $L_{i,j} = \mathcal{N}(\bar{L}_{i,j}, \sigma_L^2)$, where $\bar{L}_{i,j}$ is the nominal length plus noise with *process variance* σ_L^2. Recall the definition of the node assembly function in equation 1.

The recursive nature of equation 1 is readily apparent; for each X_i, replace it with $\mathcal{F}_i$, and repeat down to the base cases: $X_1 = 0, X_2 \sim \mathcal{N}(\bar{L}_{1,2}, \sigma_L^2)$. Let the function returning the full structure state $\mathbf{X}$ as a function of the full set of struts in E_A be $\mathcal{F}_V$

$$\mathbf{X} = \mathcal{F}_V(L_E) \tag{11}$$

Note that while the error model for each individual strut $\mathcal{N}(\bar{L}_{i,j}, \sigma_L^2)$ is independent, error accumulates through node assembly. The probability distribution of the full structure state $\mathbf{X}$ is therefore the product of the probability distributions of each of the lengths $p_{L_{i,j}}$, where each is the aforementioned distribution and the variable is the length between nodes

$$p(\mathbf{X}) = \prod_{i,j \in E_A} p\left(\left\|X_i - X_j\right\| - \bar{L}_{i,j}\right) \tag{12}$$

This generic definition can be approximated as a linear model around the nominal structure $\overline{\mathbf{X}}$ for very small noise, a valid assumption for space telescope trusses with standard deviations on the order of 10^{-6} m. As described in Crassidis and Junkins (2011), the approximate covariance of a nonlinear function of Gaussian noises can be found as follows. Because the length noises are independent of each other (i.e. zero-mean Gaussian noise added to a nominal length), the Σ_{L_E} term is a diagonal matrix with entries σ_L^2, allowing for simplification

$$\begin{aligned} p(\mathbf{X}) &\approx \mathcal{N}(\overline{\mathbf{X}}, \Sigma_X) \\ \Sigma_X &= J \Sigma_{L_E} J^{\mathrm{T}} = \sigma_L^2 J J^{\mathrm{T}} \end{aligned} \tag{13}$$

The term J is the Jacobian matrix of the full structure state with respect to L_E with rows $L_{i,j}$, evaluated at the nominal lengths $\bar{L}_E$

$$J = \left.\frac{d\mathcal{F}_V(L_E)}{dL_E}\right|_{\bar{L}_E} \tag{14}$$

We chose to use the covariance matrix trace as the key metric for minimization for the following reasons: the trace is the sum of the expected squared errors of each variable, $Tr(\Sigma_X) = \sum_i E[(X_i - \bar{X}_i)^2]$

$$A_{optimal} = \underset{A_i}{\operatorname{argmin}} \left(\operatorname{Tr}(\Sigma_{X,A_i})\right) \tag{15}$$

When expanded out using the definitions of $\mathbf{X}$ and L_E in Equations 3 and 4, equation 13 becomes

$$\begin{aligned} \Sigma_X &= \begin{pmatrix} \frac{\mathrm{d}x_2}{\mathrm{d}L_{2,1}} & \cdots & \frac{\mathrm{d}x_2}{\mathrm{d}L_{last}} \\ \vdots & \ddots & \vdots \\ \frac{\mathrm{d}z_N}{\mathrm{d}L_{2,1}} & \cdots & \frac{\mathrm{d}z_N}{\mathrm{d}L_{last}} \end{pmatrix} \cdot \begin{pmatrix} \sigma_L^2 & 0 & \cdots \\ 0 & \sigma_L^2 & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix} \cdot \\ &\quad \begin{pmatrix} \frac{\mathrm{d}x_2}{\mathrm{d}L_{2,1}} & \cdots & \frac{\mathrm{d}z_N}{\mathrm{d}L_{2,1}} \\ \vdots & \ddots & \vdots \\ \frac{\mathrm{d}x_2}{\mathrm{d}L_{last}} & \cdots & \frac{\mathrm{d}z_N}{\mathrm{d}L_{last}} \end{pmatrix} \\ &= \begin{pmatrix} \sum_{i,j \in E} \sigma_L^2 \frac{\mathrm{d}x_2}{\mathrm{d}L_{i,j}}^2 & \cdots & \cdots \\ \cdots & \ddots & \cdots \\ \cdots & \cdots & \sum_{i,j \in E} \sigma_L^2 \frac{\mathrm{d}z_N}{\mathrm{d}L_{i,j}}^2 \end{pmatrix} \end{aligned} \tag{16}$$

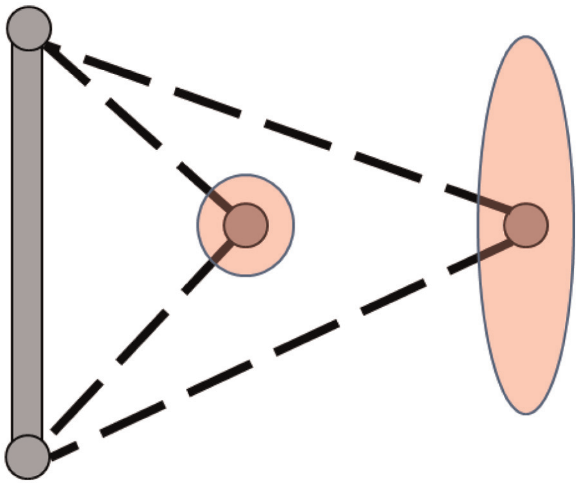


Fig. 3. The covariance of a floating node with respect to its base can be reduced by making the struts as close to orthogonal as possible. As the struts come closer to parallel, the covariance will grow in the perpendicular axis.

The trace of Σ_X, and the expected squared error of $\mathbf{X}$, is simply the length variance σ_L^2 multiplied by the sum of the derivatives of each node variable with respect to each length variable — the sum of the squared elements of the Jacobian

$$\operatorname{Tr}(\Sigma_X) = \sigma_L^2 \sum_{i=1}^{N} \sum_{j,k \in E_A} \frac{\mathrm{d}X_i}{\mathrm{d}L_{j,k}}^2 \tag{17}$$

In the algorithms presented in Sections 3 and 4, $\mathbf{X}$ is a vector representing the entire structure, but if the set of strut lengths L_E is instead a partial structure $L_{E,candidate}$ as is the case in several algorithms, the unbuilt nodes in $\mathbf{X}$ will not be calculated, and will instead be 0 by default. Likewise, the corresponding elements in Σ_X will be 0.

2.4. Covariance trace and cell geometry

The covariance trace of a node with respect to only the struts connecting it to its base varies with the orthogonality of the struts. The closer to orthogonal the set of float-base struts is, the smaller the covariance trace of the float with respect to the base. This concept is illustrated in Figure 3.

Strut orthogonality is an important consideration for both assembly sequencing and structure design. For sequencing, the trace of float base pairs will increase the farther from orthogonal the struts are. In extreme cases of nearly degenerate tetrahedra, small perturbations can lead to computational errors, resulting in physically meaningless traces. However, this will be a problem only for naïve random sequencing, except in cases where all build paths must use nearly degenerate tetrahedra; this is discussed further in Section 6.2.

2.5. Approximating the Jacobian

The closed form derivative of $\mathcal{F}_V(L_E)$ is impractical to calculate due to the number of terms and the embedded square

Table 1. Comparison of the assembly sequence mean errors of the 45-node telescope truss using combinations of any starting triangles or central starting triangles; and random, greedy, and local search.

Assembly sequence type	Mean ± Standard Deviation (m^2)
Any random sequence from random starting triangles	Degenerate
Any random sequence from central starting triangles	Degenerate
Greedy assembly from random starting triangles	$7.92 \times 10^{-4} \pm 2.06 \times 10^{-4}$ m^2
Greedy assembly from central starting triangles	$5.08 \times 10^{-4} \pm 2.88 \times 10^{-4}$ m^2
Local search from greedy assembly from random starting triangles	$6.18 \times 10^{-4} \pm 1.15 \times 10^{-4}$ m^2
Local search from greedy assembly from central starting triangles	$4.94 \times 10^{-4} \pm 2.70 \times 10^{-5}$ m^2

root functions, so we chose to approximate it instead. The central difference formula, described in many sources including Mathews and Fink (2004), approximates it with a small step size h, and has $O(h^2)$ error

$$\frac{\mathrm{d}f(x)}{\mathrm{d}x} = \frac{f(x+h) - f(x-h)}{2h} + O(h^2) \tag{18}$$

To find the derivatives of all node positions $\mathbf{X}$ with respect to a single length $L_{i,j}$, simulate two assemblies on a structure with all lengths unchanged except for $L_{i,j}$, to which h is either added or subtracted, then follow the formula

$$\frac{\mathrm{d}f_V(L_E)}{\mathrm{d}L_{i,j}} = \frac{f_V(L_E|_{L_{i,j}=\bar{L}_{i,j}+h}) - f_V(L_E|_{L_{i,j}=\bar{L}_{i,j}-h})}{2h} + O(h^2) \tag{19}$$

The value we used for h is $h = 5 \times 10^{-7}$, making the Jacobian approximation error $O(10^{-13}m)$.

3. Finding a locally optimal assembly sequence

In this section, we discuss an algorithm that can find assembly sequences with low variance in polynomial runtime by using a combination of heuristics. The set of assembly sequences is *exponential* in size, and the problem is NP-Hard, so the optimal sequence cannot be found except through brute force search on very small structures. However, a few heuristics may be employed to find very good sequences.

- A centralized starting point, building incrementally outward, will lead to better results, because a node is likely to be more precise if the chain of steps leading to it is smaller. This algorithm, **AllCentralTriangles** (shown in Table 2 with all other **bold** algorithms), searches for triangles that minimize the maximum step count between the starting triangle and the last assembled cell.
- At each step, greedily choosing the next cell so that the trace is minimized leads to a good, but not optimal, assembly sequence. This algorithm is **Greedy Assembly**.
- Better sequences can be found by modifying a sequence one step at a time, eventually finding a local minimum, in which any swapped step leads to an error that is at least as bad as the local minimum. This algorithm is **AssemblyLocalSearch**.

In practice, we found that implementing all of the above heuristics results in an algorithm that can reliably produce assembly sequences with smaller traces than any heuristic alone. This algorithm is **FindLocallyOptimalSequence**, which first picks a central starting triangle, generates an initial sequence by greedy choice, then swaps steps until a minimum trace is found.

When combined with a state estimation and correction algorithm such as the Maximum Likelihood Estimator (MLE) assembly algorithm presented here and the EKF assembly algorithm presented in Komendera and Correll (2014), the difference between a locally optimal sequence and the globally optimal sequence is often reduced substantially, lessening the necessity of a globally optimal sequencer in practice.

3.1. Finding a central starting triangle

Starting centrally has an intuitive explanation. For example, assembling a truss in a zig-zag fashion results in very large errors due to the long chain of error propagation for each node. Starting from the center and proceeding outward in concentric circles is likely to be better. We encode this heuristic by finding a starting triangle that minimizes the number of steps needed to assemble the most distant node. This is analogous to finding a *central vertex* in a graph. The distinction is that a node cannot be reached on the assembly trajectory unless at least three of its adjacent nodes are in place. To find the set of central triangles, first all triangles must be enumerated, then a stepwise-minimal sequence must be generated for each triangle to determining how many time steps are needed.

The function 3-Cliques, called by **AllCentralTriangles**, is simply a brute force search for sets of 3 nodes that have edges between them. Each starting triangle will have a number of stepwise-minimal sequences, but the length of $P_{minsteps}$ for that starting triangle will vary: a centralized triangle will have a shorter $P_{minsteps}$. The length of $P_{minsteps}$ can be found by randomly generating a sequence using

Table 2. **AllCentralTriangles** finds the starting triangles with the shortest parallel assembly sequences, **RandomSequence** returns a random sequence, either by taking any random sequence, or finding a stepwise-minimal sequence, **PossibleNextNodes** returns all of the steps that can be added to the structure, and **AdjacentSequences** returns all of the valid sequences neighboring the sequence in question. **FindLocallyOptimalSequence** starts from a random central triangle, performs a **GreedyAssembly** minimizing trace each step, then performs a **AssemblyLocalSearch** to find a local minimum. Finally, **MLEAssembly** performs an assembly with measurement feedback, either by simulation or by physical trial. Note: where the notation $\alpha \cup (\beta)$ appears, β is appended to the tuple α.

```
AllCentralTriangles($V, E$)
1: $\Delta_{all} \leftarrow$ 3-Cliques$(V, E)$
2: $P_{minsteps} \leftarrow \emptyset, \Delta_{minsteps} \leftarrow \emptyset$
3: for all $\Delta_i \in \Delta_{all}$ do
4:   $P_i \leftarrow$ RandomSequence$(V, E, \Delta_i, \text{minsteps})$
5:   if $P_{minsteps} = \emptyset$ or $\|P_i\| < \|P_{minsteps}\|$ then
6:     $P_{minsteps} \leftarrow P_i, \Delta_{minsteps} \leftarrow \{\Delta_i\}$
7:   else if $\|P_i\| = \|P_{minsteps}\|$ then
8:     $\Delta_{minsteps} \leftarrow \Delta_{minsteps} \cup \{\Delta_i\}$
9: return $\Delta_{minsteps}$
```

```
RandomSequence($V, E, \Delta_{start}, Q$)
1: $P_{partial} \leftarrow (\Delta_{start})$
2: while $\sum_t \|P_{partial,t}\| < \|V\|$ do
3:   $V_{made} \leftarrow \{f : f \in P_{partial}\}$
4:   $S_{next} \leftarrow$ PossibleNextNodes$(V_{made}, V \setminus V_{made}, E)$
5:   if $S_{next} = \emptyset$ then
6:     return No Assembly Sequence
7:   if $Q =$Minsteps then
8:     $L_{next} \leftarrow \{(i, B_{random}) : \forall i \in S_{next}, (i, B_{random}) \in \{(i, B) : i \in S_{next}\}\}$
9:   else if $Q =$Any then
10:    $L_{next} \leftarrow \{\text{RandomChoice}(S_{next})\}$
11:  $P_{partial} \leftarrow P_{partial} \cup (L_{next})$
12: return $P_{partial}$
```

```
PossibleNextNodes($V_{made}, V_{candidates}, E$)
1: $V_{possible} \leftarrow \{j : \{i, j\} \in E, i \in V_{made}, j \in V_{candidates}\}$
2: $S_{next} \leftarrow \emptyset$
3: for $f \in V_{possible}$ do
4:   $V_{f,allbases} = \{b : \{f, b\} \in E, b \in V_{made}\}$
5:   $S_{next} \leftarrow S_{next} \cup \{(f, B) : B \subseteq V_{f,allbases}, \|B\| = 3\}$
6: return $S_{next}$
```

```
AdjacentSequences($A, V, E$)
1: $A_{adj} \leftarrow \emptyset$
2: $E_A \leftarrow \{(b, f) : (f, B) \in A, b \in B\}$
3: $\Delta_{all} \leftarrow$ 3-Cliques$(V, E_A)$
4: $A_\Delta \leftarrow \{$RandomSequence$(V, E_A, \Delta_i, \text{minsteps}) : \Delta_i \in \Delta_{all}\}$
5: for all $(f, B) \in A$ do
6:   $V_{independent} \leftarrow \{i : i \text{ does not depend on } f\}$
7:   $S_{f,others} \leftarrow$ PossibleNextNodes$(V_{independent}, \{f\}, E) \setminus \{(f, B)\}$
8:   for all $(f, B_{other}) \in S_{f,others}$ do
9:     $A_{f,other} \leftarrow ((i, B_i) \in A : i \neq f) \cup (f, B_{other})$
10:    $A_{adj} \leftarrow A_{adj} \cup \{A_{f,other}\}$
11: $A_{adj} \leftarrow A_{adj} \cup A_\Delta$
12: return $A_{adj}$
```

```
FindLocallyOptimalSequence($\bar{\mathbf{X}}, V, E, \sigma_L, \sigma_M, h$)
1: $\Delta_{central} \leftarrow$ RandomChoice(AllCentralTriangles$(V, E)$)
2: $A_{greedy} \leftarrow$ GreedyAssembly$(V, E, \Delta_{greedy}, \sigma_L)$
3: $A_{best} \leftarrow$ AssemblyLocalSearch$(A_{greedy}, V, E, \sigma_L)$
4: return $A_{best}$
```

```
GreedyAssembly($V, E, \Delta, \sigma_L$)
1: $A_{made} \leftarrow \Delta$
2: while $\|A_{made}\| \neq V$ do
3:   $V_{made} \leftarrow \{f : (f, B) \in A_{made}\}$
4:   $S_{next} \leftarrow$ PossibleNextNodes$(V_{made}, V \setminus V_{made}, E)$
5:   $A_{best} \leftarrow \emptyset, T_{best} \leftarrow \infty$
6:   for all $(f, B) \in S_{next}$ do
7:     $A_{candidate} \leftarrow A_{made} \cup (f, B)$
8:     $T_{candidate} \leftarrow \sigma_L^2 \left.\frac{\mathrm{d}f_V(L_{E,candidate})}{\mathrm{d}L_{E,candidate}}\right|^2_{\bar{L}_{E,candidate}}$
9:     if $T_{candidate} < T_{best}$ then
10:      $A_{best} \leftarrow A_{candidate}, T_{best} \leftarrow T_{candidate}$
11:  $A_{made} \leftarrow A_{candidate}$
12: return $A_{made}$
```

```
AssemblyLocalSearch($A_{start}, V, E, \sigma_L$)
1: $A_{best} \leftarrow A_{start}$
2: $T_{best} \leftarrow \sigma_L^2 \left.\frac{\mathrm{d}f_V(L_{E,best})}{\mathrm{d}L_{E,best}}\right|^2_{\bar{L}_{E,best}}$
3: $Min \leftarrow False$
4: while $Min = False$ do
5:   $Min \leftarrow True$
6:   $A_{adjacents} \leftarrow$ AdjacentSequences$(A_{best}, V, E)$
7:   for all $A_{candidate} \in A_{adjacents}$ do
8:     $T_{candidate} \leftarrow \sigma_L^2 \left.\frac{\mathrm{d}f_V(L_{E,candidate})}{\mathrm{d}L_{E,candidate}}\right|^2_{\bar{L}_{E,candidate}}$
9:     if $T_{candidate} < T_{best}$ then
10:      $Min \leftarrow False$
11:      $A_{best} \leftarrow A_{candidate}, T_{best} \leftarrow T_{candidate}$
12: return $A_{best}$
```

```
MLEAssembly($\bar{\mathbf{X}}, E, A, \sigma_L, \sigma_M$)
1: $\hat{\mathbf{X}} \leftarrow 0$
2: $M_K \leftarrow 0$
3: for all $(f, B) \in A$ do
4:   $L_{f,B} \leftarrow \|\bar{X}_f - \hat{X}_b\| \forall b \in B$
5:   $X_f \leftarrow \mathcal{F}_f(X_B, L_{f,B} + \mathcal{N}(0, \sigma_L^2))$, $X_f$ is hidden.
6:   $M_K \leftarrow M_K \cup (\|X_f - X_i\| + \mathcal{N}(0, \sigma_M^2)) \forall \{f, i\} \in E_K$
7:   $\bar{X}_L \leftarrow \mathcal{F}_{V_K}(L_{K,A})$
8:   $p_{log} = \sum_{f,b \in B,(f,B) \in A_K} -\frac{(\|X_f - X_b\| - L_{f,b})^2}{2\sigma_L^2} + \sum_{i,j \in M_K} -\frac{(\|X_i - X_j\| - M_{i,j})^2}{2\sigma_M^2}$
9: $\hat{\mathbf{X}} \leftarrow$ GradientAscent$(p_{log})$ starting at $\bar{\mathbf{X}}$
10: return $\hat{\mathbf{X}}$
```

RandomSequence with the argument Q=Minsteps. Beginning with the starting triangle, and continuing until the number of assembly steps in the partial parallel sequence $P_{partial}$ is N, the set of all possible additions to the structure S_{next} is found by calling **PossibleNextNodes**, which returns all neighboring nodes and all the possible base triples for each neighboring node. When Q=Minsteps, a set of additions L_{next} is chosen: for every neighboring node, randomly pick one of the steps containing that node. In this way, a flood fill is performed, but since there may be multiple bases for any node, it is randomized. While its base is randomly chosen, that node will always have the same t value for independent runs of **RandomSequence**. The set of new nodes L_{next} is then appended to the partial parallel order $P_{partial}$. The complete parallel order is returned.

The algorithm **AllCentralTriangles** requires **Random Sequence** to be called for all triangles to determine which ones produce the shortest parallel sequences.

3.2. Assembly by greedily minimizing trace

The algorithm **GreedyAssembly** assembles a structure incrementally by choosing the assembly step that minimizes the overall trace. While this will not always produce the best option, it does not require backtracking, and returns good sequences. In addition, it can be used as a starting point for a local search to find a better sequence. It is conceptually simple: beginning from the starting triangle, look at each possible next node, and choose the float-base pair that minimizes structure covariance. Calling **GreedyAssembly** for a central starting triangle has a better chance of finding a lower covariance than starting from other triangles, as we show in Section 6.

3.3. Descent by local search around complete sequences

Instead of starting from a starting triangle (or nothing at all) and incrementally assembling a structure, one can find assembly sequences by tweaking existing sequences. **AssemblyLocalSearch** examines every single neighboring sequence of a given sequence and moves to the sequence whose trace is the smallest. The process is repeated for the new sequence until no neighboring sequence has a better trace. This is a local minimum in the space of assembly sequences.

The neighboring sequences for a given sequence is generated by **AdjacentSequences**. We define two sequences as adjacent if one of the following holds:

- A_i and A_j have the same assembly edge set E_A, but different starting triangles;
- A_i and A_j differ by a single assembly step $\langle f,(i, j, k)\rangle$.

The set of sequences with identical E_A and different starting triangles is found first, and saved as A_Δ. For the reduced set E_A, or any set E with exactly $3N-6$ edges, only one assembly sequence exists, so **RandomSequence** will only return that sequence. Not all triangles will have a sequence; when **RandomSequence** returns no assembly sequence, nothing is added to A_Δ for that triangle.

The set of sequences with one altered step is found next. For every step (f, B_f) in A, all other possible bases B_{other} are considered that do not violate the assembly sequence. The set $V_{independent}$ is the set of all vertices that can be assembled when (f, B) is removed from A, and is the set from which all base triples will be drawn. The full set of base pairs for f, $S_{f,others}$, is found by calling **PossibleNextNodes** with $V_{independent}$ and $\{f\}$, and removing the current step (f, B). The set A_{adj} is the full set of assembly sequences found by replacing (f, B) with each member of $S_{f,others}$. The union of A_{adj} and A_Δ is then returned.

3.4. Runtime

Theorem 1. ***FindLocallyOptimalSequence*** *is $O(S(N)N^3)$, where $S(N)$ is the number of swaps required to find the local minimum in* ***AssemblyLocalSearch***.

Proof. The first step is to call **AllCentralTriangles**. Its first call is to the 3-Cliques function, which returns $O(N)$ triangles in $O(N)$ time for sparse truss structures, due to the following fact: since each vertex has $O(1)$ neighbors, iterating through the adjacency list of V, E to find triangles can be made more efficient by looking at one vertex at a time, looking at each pair in that vertex's adjacency list ($O(1)$ pairs), then for each vertex in the pair, try to find the original and other vertex (each step also being $O(1)$). This is repeated N times.

The **RandomSequence** algorithm, for a starting triangle, must take N assembly steps, each time calling **PossibleNextNodes**. Collecting all base nodes for a given float node takes $O(D_v)=O(1)$ time, and iterating all 3-subsets takes $O(D_V^3)=O(1)$ time. This multiplied for all $O(V_{possible})$ nodes and added to the time for possible edges and nodes results in $O(V_{made}+V_{possible})$ time and $O(V_{made})$ possible nodes. On average, $\|V_{made}\| = \|V_{possible}\| = \frac{N}{2}$, making **PossibleNextNodes** $O(N)$ in time and in next node count. For all steps, **RandomSequence** as presented is $O(N^2)$. However, **RandomSequence** can be tweaked to run in $O(N)$ time by considering neighboring nodes of only the latest node added, and adding the set to a maintained set of previously found possible next nodes. With the above optimization included, **AllCentral Triangles** is $O(N^2)$.

Next is to call **GreedyAssembly**, which makes N calls to the assembly loop until assembly is finished. Using the same tweak previously described, the **PossibleNextNodes** step can be considered $O(1)$. Finding the covariance trace is $O(N^2)$; for each of the $3N-6$ lengths, $\mathcal{F}_V$ is called twice to generate two modified structures $\mathbf{X}_{\pm h}$, each an $O(N)$ call. All told, **GreedyAssembly** is $O(N^3)$.

Finally, **AssemblyLocalSearch** is called. Before the while loop, the covariance trace is $O(N^2)$. Inside the loop, **AdjacentSequences** is called once. **AdjacentSequences** can only return $O(N)$ sequences. The sequences with different starting triangles number only $O(N)$, and the swapped sequences are $O(N)$ for a similar reason: each node is a part of only $O(1)$ triangles, resulting in $O(1)$ swapped steps per node. Each call of **PossibleNextNodes** is only $O(1)$ because only the node in question is examined. However, **AdjacentSequences** runs in $O(N^2)$ time, since it calls **RandomSequence**N times. The inner for loop in **AssemblyLocalSearch** runs $O(N)$ times for the adjacent sequences, calculating the covariance trace each time, making the interior of the while loop $O(N^3)$.

The last step is to determine how many times the while loop is executed before the local minimum is found. Any sequence can be changed to any other sequence through $O(N)$ steps by swapping one step at a time. The simulations in the following sections suggest that the length is roughly linear. However, greedily choosing neighboring sequences may not take a linear path to the local minimum, so we say that the number of swaps is $S(N)$, and **FindLocally OptimalSequence** is $O(S(N)N^3)$. We conjecture that $S(N) = O(N)$, but have yet to prove it.

These results are also corroborated by brute-force enumeration of all possible build paths for a number of truss structures with different truss geometries and shapes, which are available in Komendera (2014).

4. MLE assembly algorithm

The MLE assembly algorithm incorporates a maximum likelihood estimator (MLE) with an assembly algorithm that can adjust the assembly parameters based on the measured errors in the previously-completed steps. The estimator uses the full set of measurements taken since the start of the assembly to estimate the state at each step of the assembly. This is in contrast to an estimator that uses only the current measurement such as the EKF, which was chosen in Komendera and Correll (2014). While the EKF would also update the estimates of all nodes in a structure during a loop closure, the EKF runs into its limitations when the error is growing too big, which we demonstrate on a selected structure.

The MLE is conceptually simple, but depends on a gradient ascent of a joint probability distribution function (also called the likelihood function), which typically is in a high-dimensional space, and is approximately 0 nearly everywhere except near its modes. The standard method to mitigate this problem is to find the maximum of the logarithm of the likelihood function. Since the number of variables in the joint PDF is $3N - 6 + M$ (in these experiments, on the order of hundreds of variables), the state space is not too large to rule out log likelihood gradient ascent. The log likelihood of a normal distribution function is

$$\log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}\right) = \frac{1}{\sqrt{2\pi\sigma^2}} - \frac{(x-\mu)^2}{2\sigma^2} \qquad (20)$$

Since $1/\sqrt{2\pi\sigma^2}$ does not vary with $\mathbf{X}$, only the $-(x-\mu)^2/(2\sigma^2)$ term is needed. Recall from equation 12 that the process noises on each edge are independent, as are the measurements. For a substructure K, this becomes

$$p_{\log} = \sum_{f,b \in B, (f,B) \in A_K} -\frac{\left(\|X_f - X_b\| - L_{f,b}\right)^2}{2\sigma_L^2} + \sum_{i,j \in M_K} -\frac{\left(M_{i,j} - \|X_i - X_j\|\right)^2}{2\sigma_M^2} \qquad (21)$$

Where the left term represents the prior desired lengths, and the right term represents the posterior measurements. The most likely state $\hat{X}$ is

$$\hat{X}_K = \underset{X_K}{\operatorname{argmax}}\,(p_{log}) \qquad (22)$$

One obvious method to solve for the maximum likelihood is to solve the derivative $\frac{dp_{log}}{d\mathbf{X}} = 0$, but this results in an unwieldy system of equations due to the derivative of each term having the form $-\frac{(v_i - v_j)(\|X_i - X_j\| - V_{i,j})}{\sigma^2 \|X_i - X_j\|}$, where v stands in for x, y, or z, and V stands in for L or M.

It is computationally simpler to find the maximum numerically. Mirrored cells are a problem, as previously discussed for $\mathcal{F}$. There exists a local maximum for every combination of mirrored cells, so it is important to start in the neighborhood of the correct solution. The solution is to start the search at the length-wise nominal structure $\mathbf{X}_L{}^1$. A gradient ascent function can be used to find the local maximum of p_{log} with respect to the node positions.

4.1. Assembly with MLE

The assembly function, **MLEAssembly**, begins with a predetermined nominal structure and assembly sequence, and sets the estimated structure $\widehat{\mathbf{X}}$ and measurement set M_K to $\mathbf{0}$. For each step (f, B) of the assembly sequence, until the assembly is complete, the strut lengths are calculated by finding the differences between the nominal float node position $\bar{X}_f$ and the current estimate of the base nodes, $\hat{X}_B$. The next two lines, 5 and 6, represent the bonding of the new struts and the post-bond measurements. The implementation is dependent on whether this is a simulation or a physical trial.

- In a simulation, the hidden state X_f is calculated by adding random noise with variance σ_L^2 to the nominal lengths. Then, measurements are simulated with variance σ_M^2 on the set of edges between the float and the structure, *including* passive struts.
- In a physical trial, the IPJRs are commanded to move to the nominal lengths. Then, the IPJRs measure their

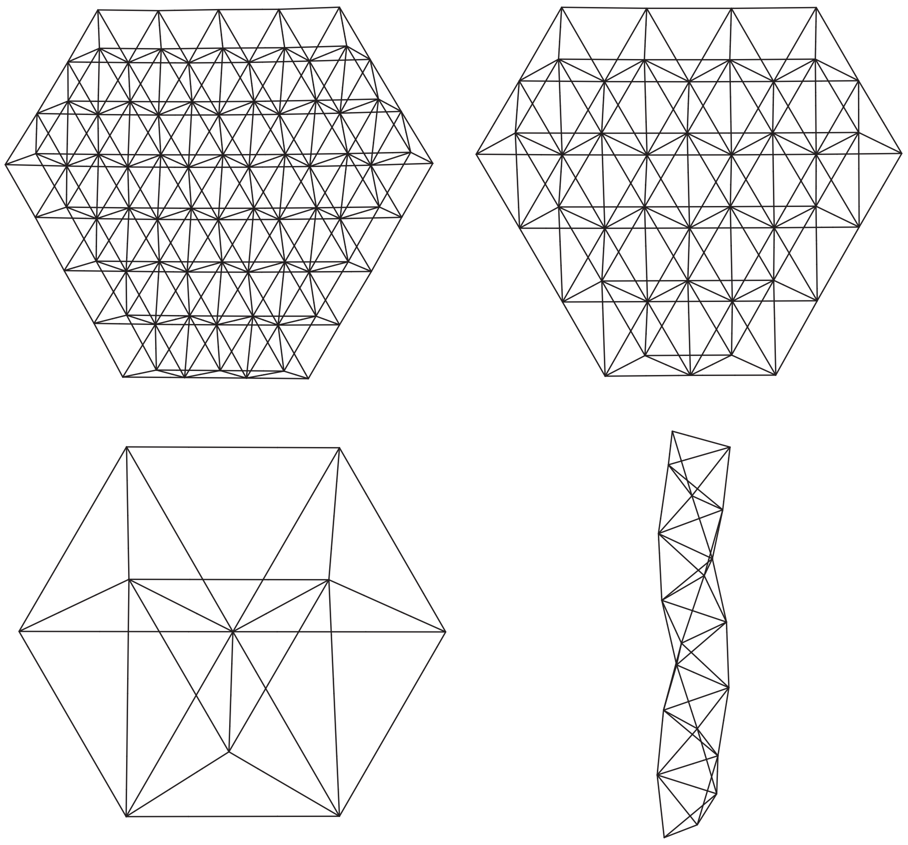


Fig. 4. Upper left: the full 84-node telescope truss from Watson et al. (2002). Upper right: the innermost 45 nodes of the full truss. Lower left: a set of 10 nodes from the full truss. Lower right: a 20-node truss composed of regular tetrahedra.

lengths and append the data to the full set of measurements M_K.

The state estimate $\overline{\mathbf{X}}$ is then found by performing the MLE gradient ascent using the nominal lengths L_K and measurements M_K from all previous steps. This distinguishes it from the EKF, in which only the latest measurements are used to update the state estimate. After the position estimates of all nodes have been updated, the algorithm calculates the lengths of the next non-redundant struts to accurately position the next node at its nominal position.

5. Experimental setup

For the purpose of practical experimentation, we divided the 84-node truss into two smaller variants: the innermost 45 nodes and the innermost 10 nodes. Additionally, we considered a tower truss composed of regular tetrahedra. Each structure we considered is shown in Figure 4. The 45-node truss is used to simulate the assembly sequencing algorithm, large enough to show trends but small enough to enable fast simulations. To simulate the MLE assembly algorithm, we compare the MLE-based, EKF-based, and open loop assembly on an 84-node truss sequence generated by the sequencing algorithm. We also show the distinction between using MLE and EKF as an estimator on a tower truss made of regular tetrahedra. Finally, we both simulate and physically assemble a 10-node subset of the full telescope truss.

In order to forgo premade accurate building materials as in Doggett (2002), we chose commodity telescoping aluminum rods (30” long, 1/4” and 7/32” diameter) that can be locked in place with a shaft collar and that connect to the node balls via neodymium magnets. These design choices make assemblies temporary and allow us to reuse materials in different experiments. Each IPJR is designed to adjust the lengths of the telescoping struts by attaching to them during the assembly of a new part of the structure.

Since these components are reusable, permanent welding is not performed, and fixed struts can be easily adjusted. Thus, to simulate the expected on-orbit application with welding, the strut is measured after it is fixed to the structure, and is considered permanent until the trial is complete.

We built three IPJRs for use in this experiment. Each IPJR, shown in Figure 5, is an autonomous robot, consisting of a Raspberry Pi Model B for high level algorithmic control and communications, an Arduino for actuation and sensing, a custom motor driver board, and an Edimax WiFi dongle. The principal actuator is a Firgelli L-16 linear actuator with 140 mm extension and potentiometer length feedback discretized to actuator steps of length $\delta_{step} = 0.138$

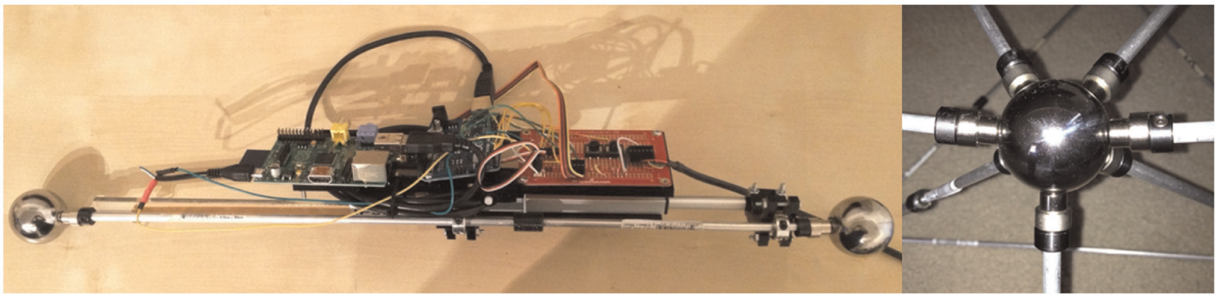


Fig. 5. The IPJR model, for use with telescoping aluminum struts and 3D trusses, is shown attached to a strut, with a node ball at either end attached to the strut by neodymium magnets. The control hardware consists of a Raspberry Pi Model B for high level control, an Arduino for low level control and sensing, and a custom motor driver board. The hardware design permits future expansions, such as motor-mounted cameras and automated strut attachment and detachment mechanisms.

mm. To attach to both tubes of a strut, each IPJR has two shaft collars, enabling the actuator to extend and contract the strut. All components are fixed to a frame consisting of laser cut, 1/4" acrylic sheets. Each IPJR was powered by a power supply that provided 5V and 12V output for the electronics and motors.

While each IPJR is fully capable of running our implementation of the MLE assembly algorithm as-is, we chose to run the algorithm from a central PC to avoid communication challenges. The Raspberry Pis instead run an HTTP server, which receives commands in the form of GET requests, and returns the data in response. Commands used in this experiment include checking the length potentiometer voltage, commanding the IPJR to move until it stops near a commanded location (without further control), and checking to see if the IPJR has finished moving. The control PC executes the experiment using a *Mathematica* implementation of the algorithms presented here, which controls and monitors the IPJRs through GET requests.

While we have previously demonstrated fully robotic external manipulation and onboard precise distance measurements on a two-dimensional test platform (Komendera et al., 2014a), for this experiment, a human external manipulator is required to attach the IPJRs to struts and to place each new cell in a coarse configuration. Once placed, the IPJRs refine the cell by using measurements of the distances between node balls, either measured by the IPJRs themselves or measured by an external sensor. For this prototype, a ruler was used in order to obtain an unbiased measurement with a variance of $\sigma_M^2 = 0.0625\,\text{mm}^2$, not subject to bending due to the weight of the IPJR or play in the shaft collar attachment mechanism. The ruler was also used for verification of the physical experiments.

The setting of an IPJR's length is performed by a simple feedback controller, in which the linear actuator adjusts its length based on the difference in measured length vs desired length. When the measured length is within a specified tolerance of the desired length L, the shaft collars are fixed and the IPJR is removed. The range of the IPJR's linear actuator is 140 mm, so to handle a larger range of strut lengths, the IPJR should be attached to the strut such that the final length is within the range. This length is the initial length, which is measured as M_L. Then an iterative loop converges to the desired tolerance by commanding the IPJR to move $\delta_{step}(L - M_L)$ steps and taking a new measurement M_L, until M_L is within a tolerance $\epsilon = 0.5\,\text{mm}$. While this controller will diverge if the actual step size is larger than the estimated step size, in our experience the actuators converged almost always within three steps, and the simple controller was sufficient.

6. Results

This section presents the results of the sequencing algorithm and the MLE assembly algorithm on the 84-node telescope truss in Watson et al. (2002), a 45-node subset of that truss, and a 10-node subset of that truss. To compare the MLE assembly algorithm with the EKF assembly algorithm we introduced in Komendera and Correll (2014), we also simulated a 20-node tower. All four structures are shown in Figure 4. We demonstrated the efficacy of the locally optimal assembly sequencer using the 45-node truss due to the high polynomial degree of **FindLocallyOptimalSequence**. To compare MLE to EKF and open loop assembly, we show the results on the 84-node truss and the tower. Finally, we performed physical trials with the 10-node truss, and compare the results with simulation.

6.1. Process and measurement variance

To estimate the σ_L process noise, we measured the pre- and post-fix lengths of 75 aluminum struts, giving us a process noise of $\sigma_L^2 = 3.57 \times 10^{-7}\,\text{m}^2 = 0.357\,\text{mm}^2$. This value is used in all simulations unless otherwise noted, and is used in the MLE algorithm for the physical experiments. The ruler variance σ_M^2 was based on visual estimation of length consistently being within 0.5 mm of the actual length; if 0.5 mm is considered the two-sigma length, $\sigma_M^2 = 6.25 \times 10^{-8}\,\text{m}^2 = 0.0625\,\text{mm}^2$.

6.2. Assembly sequencing

The assembly sequencing simulations were performed on a 45-node subset of the 84-node telescope, chosen for the faster simulation runtimes while showing trends. The 45-node telescope has 1260 possible starting triangles, 72 of which are central based on the minimum parallel order length. We used **FindLocallyOptimalSequence** and compared the resulting orders at each step. Additionally, we tweaked the algorithm to allow any triangle to be chosen and compared this to the central triangle approach. The results, shown in 1, are derived from:

- any random sequence;
- any random sequence from a central starting triangle;
- greedy assembly from the starting triangles used in any random sequence;
- greedy assembly from the central starting triangles;
- local search from greedy assembly from any starting triangle;
- local search from greedy assembly from central starting triangles.

Random sequences for the telescope almost always include nearly degenerate tetrahedra, which contribute enormous errors to the overall trace; while human assembly planners would naturally avoid the degenerate steps, the topology permits their existence as possible assembly steps. The mean traces of these random sequences blew up as a result, making their values meaningless. Therefore, random sequences are not suitable for precision assembly.

Using the same starting triangles as the random sequence experiment, the greedy assembly algorithm is an enormous improvement, and avoids degenerate tetrahedra. The means and standard deviations are $7.92 \times 10^{-4} \pm 2.06 \times 10^{-4} \mathrm{m}^2$ for any starting triangle, and $5.08 \times 10^{-4} \pm 2.88 \times 10^{-4} \mathrm{m}^2$ for central triangles. On average, greedy assembly on central starting triangles are 1.56 times more precise than assembly from any triangle, further justifying building from the center outward.

The last optimization, local search, was performed with the greedy sequences. The trajectories taken from the greedy to the local minimum are shown in Figure 6. The means and standard deviations are $6.18 \times 10^{-4} \pm 1.15 \times 10^{-4} \mathrm{m}^2$ for any starting triangle, and $4.94 \times 10^{-4} \pm 2.70 \times 10^{-5} \mathrm{m}^2$ for central triangles. For the cases starting from any triangle, locating a local minimum results in an average improvement of a factor of 1.25, requiring an average of 14.28 step changes before finding the minimum. The improvement for central triangle steepest descent is smaller: a factor of 1.03, making an average of 6.96 steps. The reduced number of steps is due to the central triangle greedy sequence starting closer to a minimum. After steepest descent, starting from a central triangle is on average 1.25 times more precise than starting from anywhere. The central triangle steepest descent sequences never found a better starting triangle than a central one, while 20% of the descents starting from any triangle changed their starting triangles to one of the central triangles.

In all of the trace histograms, there are clusters with larger variance than the others due to some of the assembly sequences having a skewed tetrahedral cell contributing a large variance. While these are not so skewed as to blow up the trace calculation, the fact that such cells were found by greedy assembly and subsequently unchanged by the local search shows that multiple random restarts should be used to find sequences without this problem.

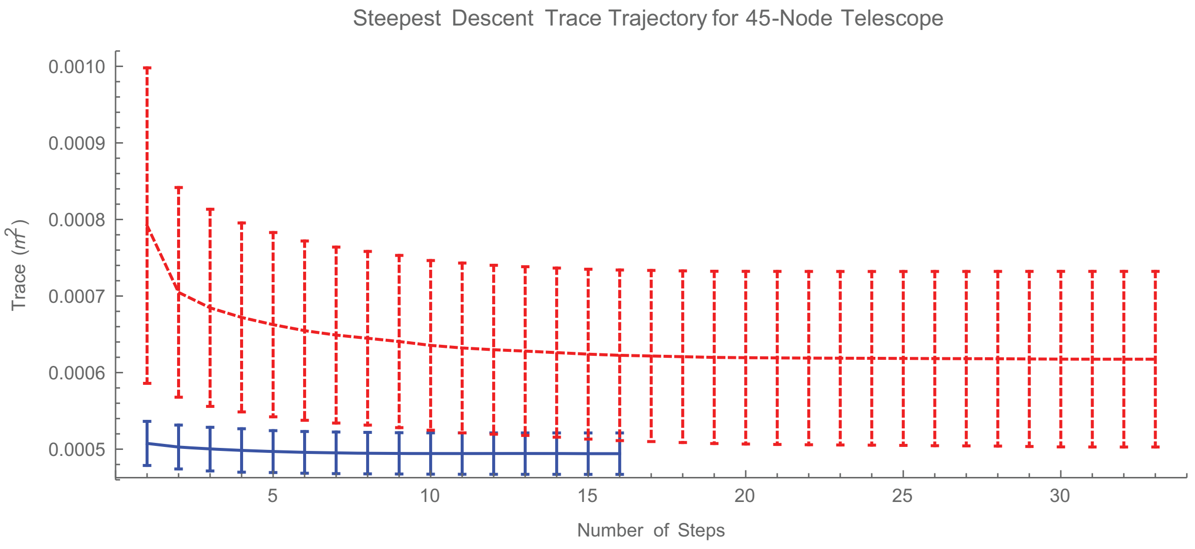


Fig. 6. The trajectories taken by the steepest descent algorithm, starting a greedy assembly on any starting triangle (dashed) and a central starting triangle (solid), averaged and extended in step count to the longest such trajectory. Descending from a greedy sequence generated from a central starting triangle finds better local minima in fewer steps.

6.3. MLE assembly compared with EKF assembly

We compared the MLE assembly algorithm to the EKF assembly algorithm in Komendera and Correll (2014) on two structures.

- The full 84-node telescope truss, shown on the top of Figure 7, with edge lengths on the order of 2 m. The 84-node truss assembly sequence was found by applying greedy assembly to a central starting triangle, then using the steepest descent descent algorithm.
- A 20-node tower composed of regular tetrahedra of edge length 1 m, shown in Figure 8, used for comparing MLE to the EKF.

The EKF assembly algorithm, presented in Komendera and Correll (2014), was suitable for locally optimal assembly sequences, but deviated when nonlinearities became large. For the 84-node telescope truss, using the IPJR and ruler variances, the lower right of Figure 7 shows that MLE and EKF are comparable, and both provide a substantial improvement over the open loop scenario. The open loop trace is $1.55 \times 10^{-3}\text{m}^2$, while the MLE and EKF traces are $2.28 \times 10^{-4}\text{m}^2$ and $2.29 \times 10^{-4}\text{m}^2$, too close to separate.

To show a scenario in which MLE and EKF estimators yield distinct results, we simulated the assembly of a simple tower of regular tetrahedra, in which each new node connects to the previously placed node. For edge lengths of 1 m, we altered σ_L to be 0.1 m and σ_M to be 0.01 m to force nonlinearity into the system. The results, in Figure 8, show that EKF starts to diverge from MLE at the 10-node mark, where the traces of the open loop, MLE, and EKF structures are $28.76\,\text{m}^2$, $1.39\,\text{m}^2$ and $2.41\,\text{m}^2$.

The central figure of Figure 8 shows the EKF case in green, revealing that all the node means are offset, resulting in shorter structures on average. The reason for this is the following: in the nonlinear model, when there is a large deviation in a strut, the truss tilts; the larger the deviation, the larger the angle of tilt. But the EKF model, using the linearization, follows the tangent when calculating the posterior, which deviates from the true value when the strut error is large. The result is that the height of the nodes is overestimated, and the remaining lengths are set smaller than they should be, resulting in a stunted tower.

The MLE does not have tangent deviation issue, due to the fact that it is calculated on the nonlinear joint probability distribution function. This extreme case highlights the benefits of estimating the posterior using the true probability distribution function. We note, however, that for precise assembly applications using smartly chosen assembly sequences, both EKF and MLE are suitable.

6.4. MLE assembly on optimal trace vs median trace telescope truss

For very small structures, such as the 10-node telescope, all assembly sequences can be enumerated. The 10-node telescope truss in Figure 9 has 12,708 distinct assembly sequences. We simulated MLE assembly on both the global optimal sequence and the median sequence to observe what benefits MLE may provide to suboptimal sequences. The results are shown in Figure 9. We ran 1000 trials of the MLE algorithm to generate the MLE covariance matrix and marginal ellipsoids shown in the bottom row.

The trace of the optimal sequence is $2.84 \times 10^{-5}\text{m}^2$, and the median sequence trace is $5.98 \times 10^{-5}\text{m}^2$, an increase of a factor of 2.1. However, as the top figure of Figure 9 shows, using MLE is much more effective on the median sequence. The traces of the optimal and median assembly sequences using MLE are $1.56 \times 10^{-5}\ \text{m}^2$ and $1.86 \times 10^{-5}\ \text{m}^2$, the median trace being only 1.19 times worse.

These results show that an optimal assembly sequence will perform better overall with the MLE estimator, but the distinction between the optimal and other sequences is not as severe as in the open loop cases. Therefore, algorithms that can find locally optimal sequences quickly, such as the steepest descent algorithm, are sufficient for all but the most strict precision demands.

6.5. Physical experiments

To test the validity of the MLE algorithm on real hardware, we performed assembly trials of the optimal 10-node telescope sequence shown in Figure 9; three open loop trials, and three MLE trials. Assembly experiments were performed with the calibrated variances $\sigma_L^2 = 3.57 \times 10^{-7}\text{m}^2$ and $\sigma_M^2 = 6.25 \times 10^{-8}\text{m}^2$. We did not model physical phenomena such as gravity, strut deflection under the weight of the IPJRs, and free play in the fixed struts; we predicted that the MLE algorithm would be able to overcome the biases those phenomena would add to the system. Each trial took one or more hours to run, mostly due to issues attaching and detaching the IPJRs. Occasionally, the magnets on the struts would attract to each other instead of the node balls, requiring careful detachment. A few restarts were necessary.

A global positioning system was not available for use as an external comparison, so to obtain a reliable ground truth, we measured the distances between a more complete set of node pairs — every such pair whose distance apart was within 0.7 m and not limited to nodes connected by struts — and performed each measurement three times, and used MLE to estimate the ground truth to an accuracy comparable to widely available global systems. In addition to using a much larger number and variety of node pair measurements, the ground truth did not use priors, removing the MLE assembly algorithm bias. We simulated 1000 sets of measurements and compared the estimated result with the hidden structure and found our ground truth method to be very accurate, a maximum mean squared error of just $0.25\,\text{mm}^2$— these are the error bars in the individual trial plots in the middle row of Figure 10.

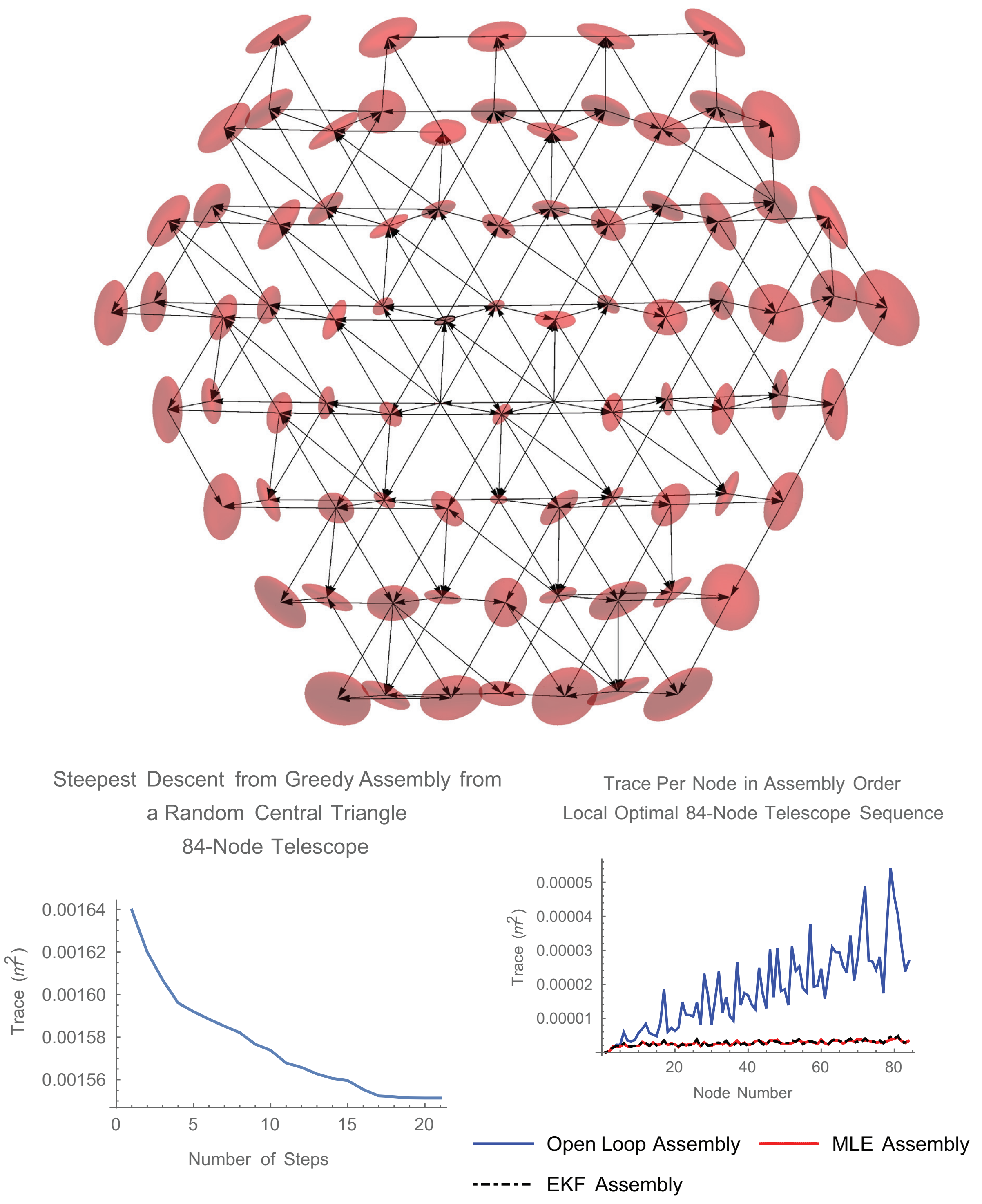
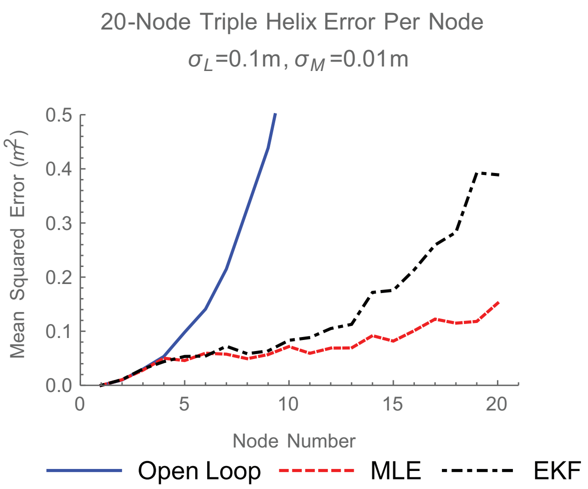


Fig. 7. Results for a single assembly sequence of an 84-node telescope truss (top), generated by the combined central triangle / greedy assembly / steepest descent shown in the lower left. The lower right shows the calculated trace of the open loop assembly, and the estimated trace of 100 trials of MLE and EKF for $\sigma_L^2 = 3.57 \times 10^{-7}\,\mathrm{m}^2$ and $\sigma_M^2 = 6.25 \times 10^{-8}\,\mathrm{m}^2$. The MLE and EKF are indistinguishable for this case. The ellipsoids are exaggerated to be visible. The arrows indicate the base-float relationship, where the float is at the pointed end.

The results are shown in Figure 10. The physical trials followed the predicted patterns in simulation. On average, the open loop mean squared error is 4.33×10^{-5} mm^2, while the MLE mean squared error is 1.09×10^{-5} mm^2. The open loop trials averaged 1.6 times worse than predicted, while the MLE trials averaged 1.4 times *better* than



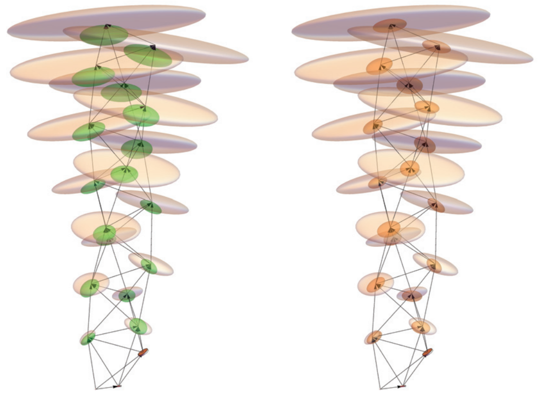


Fig. 8. To show the results of extreme nonlinearity, we use the regular tetrahedron tower. For a strut nominal length of 1 m, $\sigma_L = 0.1$ m, $\sigma_M = 0.01$ m, the EKF assembly algorithm in Komendera and Correll (2014) is shown in green, and the MLE assembly algorithm is shown in orange. While both filters produce substantially improved results, the MLE algorithm is better suited for large nonlinearities. The ellipsoids are *not* exaggerated for these renders. The arrows indicate the base-float relationship, where the float is at the pointed end.

predicted. Due to the small number of trials, and therefore the large error on the means, this is not surprising. The low error on node 7 in one of the open loop trials, for example, skewed the average to be better than the MLE average for that node. Given that node 7 depends on nodes 1 through 5, the almost-perfect positioning of node 7 in that single trial is coincidental.

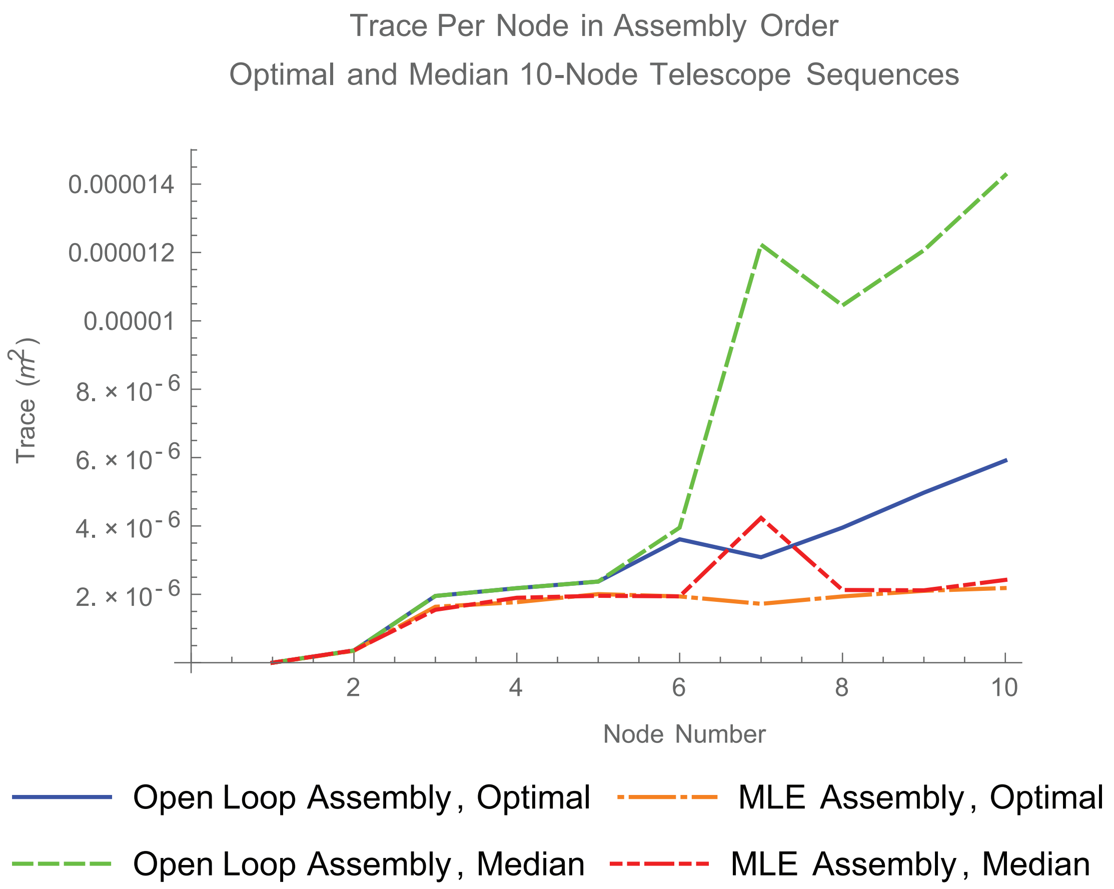
The aforementioned physical biases were not an obstacle for MLE, as predicted, but the length offsets do show a small bias, which can be seen in the lower right of Figure 10. Most edges ended up slightly shorter on average; when all edges are considered, the mean error is −0.13mm. This may be due to a combination of free play in the joints of each strut, and the large mass of the 38.1mm -diameter steel balls compressing the struts. We did expect to see a lengthening of struts due to the bowing caused by the weight of the IPJRs, and the subsequent restoration of linearity when the IPJRs were removed, but we did not observe this independently of the other biases. The variance of the edge length errors is $\sigma_L^2 = 3.54 \times 10^{-7}\text{m}^2$, almost equal to the previously estimated value of σ_L^2.

7. Discussion

We have shown that a smart choice of the build sequence and error correction based on an MLE-based estimation of the actual node positions can lead to assemblies with predictable accuracy using commodity parts. We validated this with simulations and physical trials, and showed that the MLE algorithm could mitigate both the choice of a suboptimal sequence and the unmodeled, non-Gaussian physical processes that occur during physical trials.

Our results show that without using MLE for error correction, assembling the 84-node space telescope optical bench to the required precision would require actuators that are technically not feasible. To achieve a tolerance of 10 μm on this particular structure requires that the worst case node have an average mean squared error of under $(0.00001\text{m})^2 = 10^{-10}\text{m}^2$. The locally optimal sequence discussed in Section 6.3 and shown in Figure 7 has a maximum mean squared error at node 80. To ensure that node 80 fits the level of precision, without using MLE during assembly, the process noise needs to be $\sigma_L^2 = 6.6 \times 10^{-13}\text{m}^2$. To put this into context, the standard deviation of the strut length must be, at most, *812 nanometers*. When processes such as thermal deformation under welding and sunlight are considered, even thermally favorable materials such as titanium, with its coefficient of thermal expansion of 8.6 microns per meter per Kelvin, cannot match that precision no matter what precision the actuator being used has[2].

Using MLE-based error estimation and correction, however, drastically changes this picture as it mitigates build-up of error. This might enable reaching the required accuracy with commodity sensors and actuators, such as the micrometer precise actuator and laser distance sensor that we used in Komendera et al. (2014a). The Ultra-Motion linear actuator we incorporated into the IPJR prototype has an estimated process standard deviation of $\sigma_L = 2$ μm based on the observation that at least 95% of the time, any desired length is within, at most, 4μm from an actuator step length — two standard deviations. Using the same reasoning, the KEYENCE laser distance sensor is a measurement standard deviation of $\sigma_M = 0.5$ μm. However, to be conservative in how we interpreted the precisions of the actuator and the laser sensor, and to allow room for the aforementioned physical processes, we inflated the noises to $\sigma_L = 8$ μm and $\sigma_M = 1$ μm, and simulated MLE



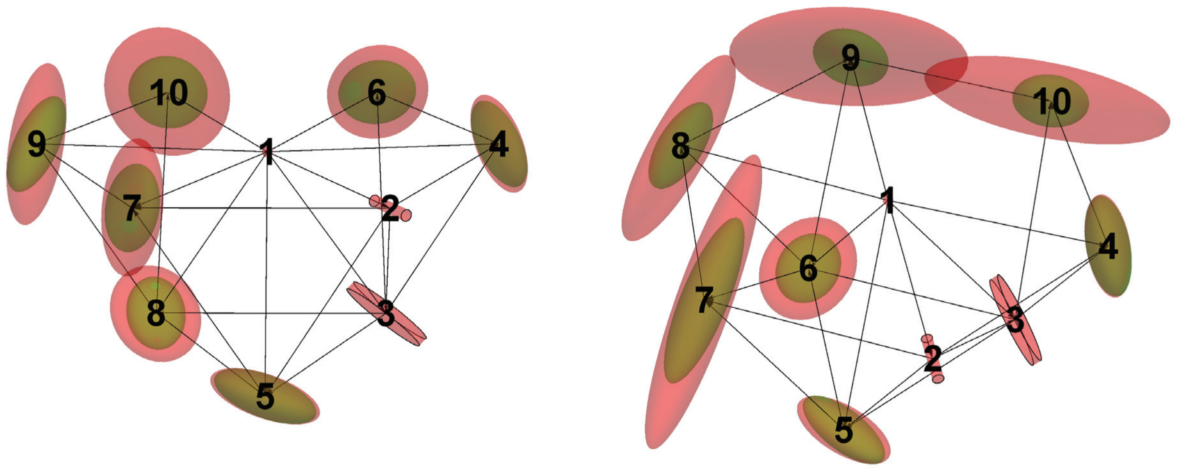


Fig. 9. The10-node space telescope has 12,708 assembly sequences. The optimal and median sequences, shown in the bottom row, are simulated with open loop and MLE, as shown in the top row. While the median trace is over two times worse, using MLE reduces this factor to just over one. The ellipsoids are exaggerated to be visible.

assembly trials on the sequence in Figure 7. It is obvious that each individual strut length can only be within 16 μm 95% of the time, meaning that the tolerance of 10 μm cannot be guaranteed with the hardware we used in Komendera et al. (2014a). That said, with MLE in the assembly process, the average trace is $3.13 \times 10^{-10} \mathrm{m}^2$, making the mean error 17.7 μm as shown in Figure 11, only slightly worse than the 16 μm per strut error. More importantly, error does not appreciably grow, meaning that if we used a better actuator, we could realistically expect to match the overall precision requirement. In a realistic deployment, however, local sensing would be complemented by an absolute laser-based measurement system. Such systems, albeit expensive, are able to provide

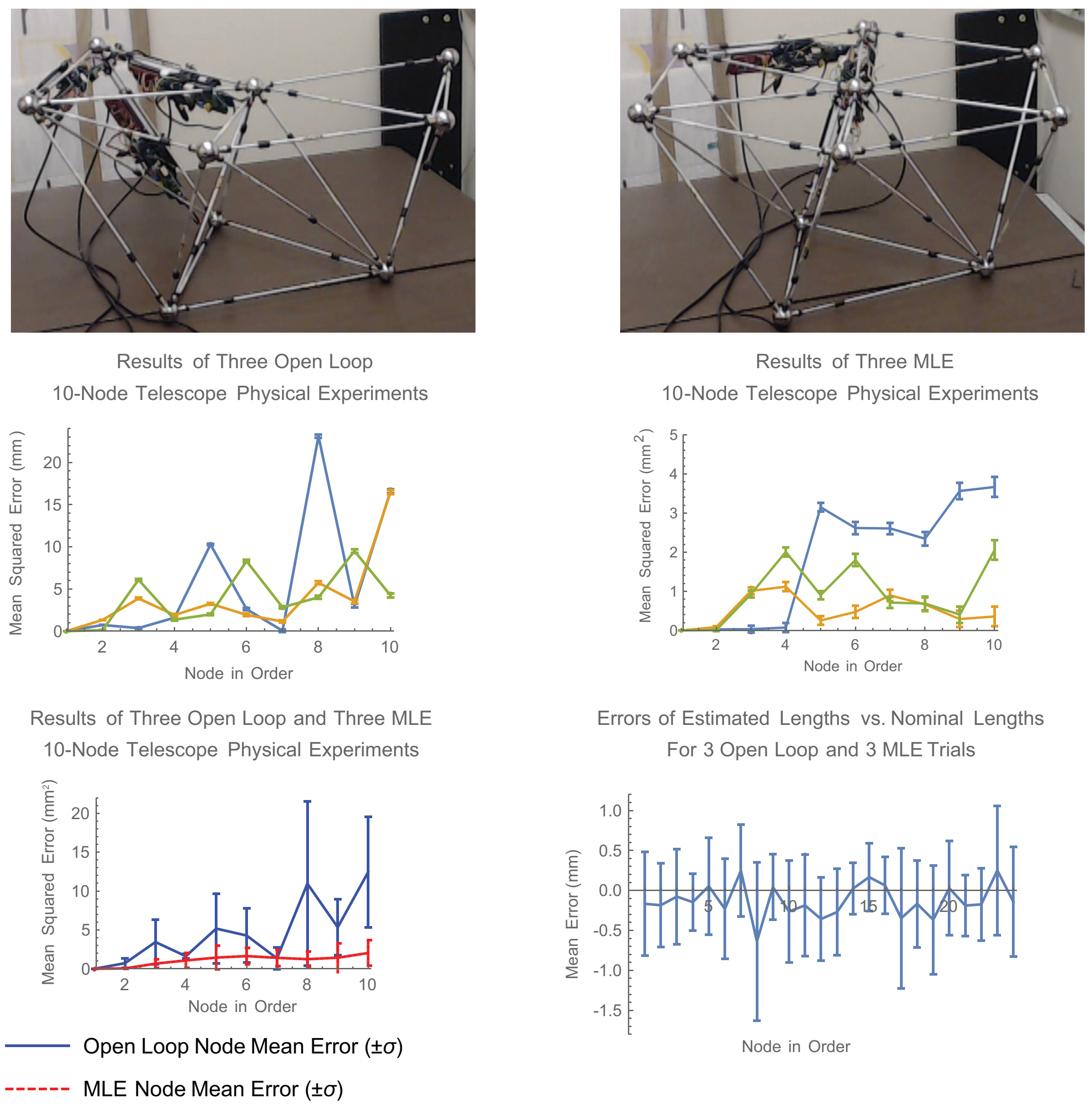


Fig. 10. The results of 3 open loop and 3 MLE physical assembly trials are shown. Top row: the final two assembly steps. Middle row: node squared errors (mm^2) for the open loop trials (left) and the MLE trials (right). Bottom left: the means of the open loop and MLE cases plotted together. Bottom right: the mean errors of the estimated final strut lengths vs the nominal lengths.

measurements in the order of micrometers over ranges of tens of meters, which would allow to provide lower bounds on assembly error across the structure, while still being able to take advantage of an estimation framework.

IPJRs are relatively simple: they only need to be able to grasp the nodes and to control their lengths with precision, a minimum of one degree of freedom if the graspers only have an on and off state. They should be less expensive and easier to make than robots with extra degrees of freedom. And given the parallelization definition presented in this paper, several groups consisting of three IPJRs can work in parallel on different branches of the structure. This also gives robustness to the assembly process; should any IPJR fail, the others will rearrange their groups to account for the reduced number. Only three are needed to assemble a structure. Since the precision is built in, the external manipulators and welders have a reduced need for precision of their own — only enough to grasp an IPJR, a strut, and weld — enabling them to be less expensive as well.

The exponential number of assembly sequences poses a challenge that is unlikely to be to overcome for larger structures. While starting from the center and performing a local search around a greedy sequence returned consistent results, the true global minimum trace may be significantly lower than any result found by the algorithm presented here. At first glance, given that trace is additive, one may

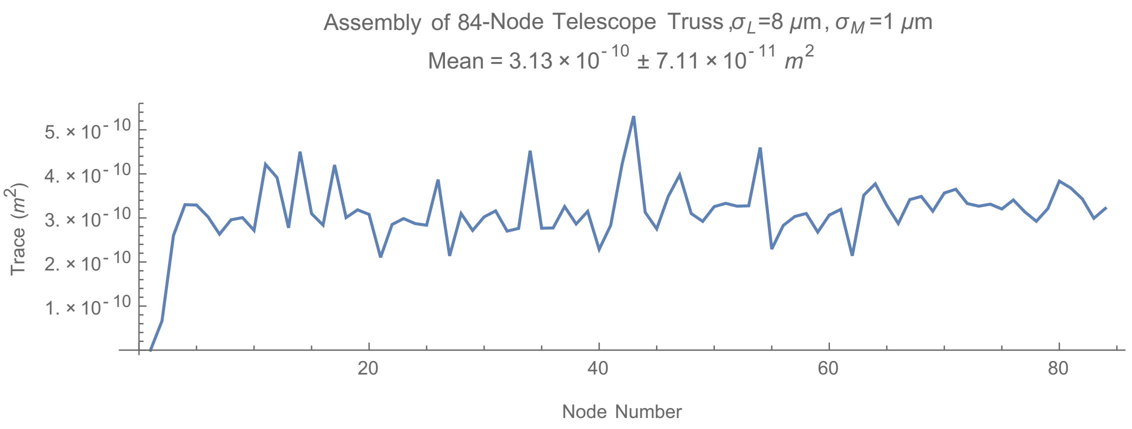


Fig. 11. The results of 200 simulations of the full 84-node telescope truss as built by the MLE assembly algorithm using the actuators and sensors in Komendera et al. (2014a) shows that the mean error does not visibly increase as node count increases, and that the mean node error is 17.7 µm, just over twice the standard deviation of a single strut.

think that finding a shortest path through the graph *V*, *E* using Dijkstra's algorithm would solve the problem quickly; but this approach does not work since a node's error depends on every strut leading up to it, and not just the previous node added. Likewise, an optimal configuration for a single node is not a requirement for the optimality of its descendants, ruling out a basic dynamic programming approach.

The $O(S(N)N^3)$ runtime of **FindLocallyOptimal Sequence** may be polynomial, but is still insufficient for structures with large numbers of parts. The algorithms presented here are not optimized; the first task is to reduce the number of times the covariance is calculated, taking advantage of shared assembly steps between sequences and saved trace values. Another method, the implementation of which is future work, is to divide the structure into smaller substructures that are assembled independently of one another and combined later, in which each substructure is considered a single element to be constructed. Doing so would constrain the set of assembly sequences that can be identified, almost certainly removing the true optimum from the set, but the speed gains may be worth it.

We note that the approach described here is conceptually similar to the Simultaneous Localization and Mapping (SLAM) problem, which is a canonical problem in mobile robotics. Similar to a mobile robot exploring an environment and mapping unique landmarks using odometry, the robots here use precise local measurements to initially localize truss nodes. Much like in SLAM, the key insight is that loop closures, that is revisiting a previously localized node, allow to update the posterior pose graph. Although not formally introduced as a SLAM problem in this paper, the conceptional similarity is important, as it might permit leveraging results from the SLAM community. In particular when considering truss-climbing IPJRs with inaccurate navigation, data association problems, or very large structures that require more efficient solutions. Our MLE assembly algorithm can be considered a full SLAM problem, which uses all of the measurements taken, while the EKF assembly algorithm we previously used can be considered an online SLAM problem, in which only the newest length measurements are incorporated into the previous state estimate.

8. Conclusion

We showed that truss structures made of commodity, imprecise components can be assembled by IPJRs to a high degree of precision. We first derived an error metric based on the covariance trace for open loop assembly. Next, we presented a combined algorithm for generating a locally optimal assembly sequence, which starts by choosing a central starting location, assembling a sequence by greedily choosing assembly steps with the minimum trace, and swapping steps of the resultant sequence until a minimum is found. We then described an algorithm based on the MLE, which builds a map of the structure and the IPJRs' positions while assembly is performed. Finally, we demonstrated that IPJRs can use the MLE assembly algorithm to produce precise truss structures made of steel node balls and aluminum telescoping tubes in spite of physical phenomena such as bending struts, which were not taken into account into the noise model.

The IPJR concept is not limited to long struts and node balls; it can extend to robots that can precisely position any components that need to be cut, welded, and measured. The key contributions of the IPJR paradigm are that robotic structure assembly can be precise, can use commodity materials, can be robust to failure, and can use teams of robots with distributed resources. The parallelization method described in this paper allows IPJRs to work in independent groups, each with an external manipulator to feed struts and to reposition the IPJRs, enabling swift and robust assembly. While this paper is specifically about truss structures assembled by strut-length-adjusting robots, we believe that the principles here — and in particular online error correction based on a MLE estimate of the actual error — hold for a larger class of incrementally assembled structures. When commodity components are used instead of self-interlocking components, and are cut and shaped as needed, care must be taken to keep track of the structure error to enable structures to be built reliably. Minimizing the covariance trace is the natural metric for choosing assembly sequences, adding it to the list of other metrics found in the literature. Modeling robotic assembly as a SLAM problem with added environmental manipulation has the potential to enable the construction of very large structures by teams of robots in open and uncertain environments, far from humans that can take over at a moment's notice.

Numerous problems remain. Do better algorithms for generating optimal sequences exist? When real time measurements are too numerous for the MLE algorithm to function properly, are other SLAM algorithms developed for large-scale robotic problems more suitable? Does modeling the additional physical phenomena provide enough of a boost to precision to make it worthwhile? How does assembling several substructures independently and combining them at a later stage affect the overall precision? Answers to these questions are the subject of future research.

A dedicated on-orbit experiment to assemble a space telescope optical bench will require hardware that is even more precise than the sensors and actuators that we used in Komendera et al. (2014a). It will also make use of external measurement systems, such as precise cameras, to further improve the estimates. Yet, the value of the IPJR paradigm is that it can proceed in the absence of such precision measurements, e.g., when parts of the structure are occluded from the absolute position sensor or when individual IPJRs fail.

Acknowledgements

We wish to thank John Dorsey and Billy Doggett from NASA Langley Research Center for their continuous support and mentorship of this project.

Funding

This work was supported by a NASA Office of the Chief Technologist's Space Technology Research Fellowship (grant number NNX11AM83H).

Notes

1. Since the lengths L_K may include lengths that were altered from the nominal starting lengths, this is not equal to $\overline{\mathbf{X}}$.
2. Composite struts can be designed so that the coefficient of thermal expansion is zero under certain conditions, but these would not be commodity materials.

References

Barros dos Santos S, Givigi S and Nascimento C (2013) Autonomous construction of structures in a dynamic environment using reinforcement learning. In: *IEEE international systems conference (SysCon)*, Orlando, USA, 15–18 April, pp. 452–459. IEEE.

Berman S, Halasz A, Hsieh MA and Kumar V (2009) Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics* 25(4): 927–937.

Bolger A, Faulkner M, Stein D, White L, Yun S and Rus D (2010) Experiments in decentralized robot construction with tool delivery and assembly robots. In: *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, Taipei, Taiwan, 18–22 October, pp. 5085–5092. IEEE.

Bonani M, Magnenat S, Rétornaz P and Mondada F (2009) The hand-bot, a robot design for simultaneous climbing and manipulation. In: Xie M, Xiong Y, Xiong C and Hu Z (eds) *Intelligent Robotics and Applications*. Berlin Heidelberg: Springer-Verlag, pp. 11–22.

Bretl T, Rock S, Latombe JC, Kennedy B and Aghazarian H (2006) Free-climbing with a multi-use robot. In: Ang MH and Khatib O (eds) *Experimental Robotics IX*. Berlin Heidelberg: Springer-Verlag, pp. 449–458.

Chu B, Jung K, Han CS and Hong D (2010) A survey of climbing robots: Locomotion and adhesion. *International journal of precision engineering and manufacturing* 11(4): 633–647.

Crassidis JL and Junkins JL (2011) *Optimal estimation of dynamic systems*. Boca Raton: CRC press.

DeFazio T and Whitney D (1987) Simplified generation of all mechanical assembly sequences. *IEEE Journal of Robotics and Automation* RA-3(6): 640–658.

Detweiler C, Vona M, Yoon Y, Yun S and Rus D (2007) Self-assembling mobile linkages. *IEEE Robotics & Automation Magazine* 14(4): 45.

Dogar M, Knepper RA, Spielberg A, Choi C, Christensen HI and Rus D (2014) Towards coordinated precision assembly with robot teams. In: Hsieh MA, Kumar V and Khatib O (eds) *International symposium on experimental robotics*. Berlin Heidelberg: Springer-Verlag.

Doggett W (2002) Robotic assembly of truss structures for space systems and future research plans. In: *IEEE aerospace conference, volume 7*, Big Sky, USA, 9–16 March, pp. 3587–3598. Piscataway: IEEE.

Dorsey JT, Doggett W, Komendera E, Correll N, Hafley R and King BD (2012) An efficient and versatile means for assembling and manufacturing systems in space. In: *AIAA SPACE conference*, Pasadena, USA, 11–13 September, pp. 1–19. Red Hook: Curran Associates.

Fox BR and Kempf KG (1985) Opportunistic scheduling for robotic assembly. In: *International conference on robotics and automation*. St. Louis, USA, 25–28 March, pp. 880–889. Piscataway: IEEE.

Gottschlich S, Ramos C and Lyons D (1994) Assembly and task planning: A taxonomy. *IEEE Robotics and Automation Magazine* 1(3): 4–12.

Hamner B, Koterba S, Shi J, Simmons R and Singh S (2010) An autonomous mobile manipulator for assembly tasks. *Autonomous Robots* 28(1): 131–149.

Heger FW (2010) *Assembly planning in constrained environments: Building structures with multiple mobile robots*. PhD Thesis, Carnegie Mellon University, USA.

Heger FW and Singh S (2010) Robust robotic assembly through contingencies, plan repair and re-planning. In: *IEEE international conference on robotics and automation (ICRA)*, Anchorage, USA, 3–8 May, pp. 3825–3830. IEEE.

Homem de Mello LS and Lee S (1991) *Computer Aided Mechanical Assembly Planning*. Boston: Springer.

Homem de Mello LS and Sanderson AC (1991) Correct and complete algorithm for the generation of mechanical assembly sequences. *IEEE Transactions on Robotics and Automation* 7(2): 228–240.

Hoyt RP, Cushing JI, Jimmerson GJ, Luise JS and Slostad JT (2014) Trusselator: On-orbit fabrication of high-performance composite truss structures. In: *AIAA SPACE conference*, San Diego, USA, 4–7 August. Red Hook: Curran Associates.

Jimenez-Cano A, Martin J, Heredia G, Ollero A and Cano R (2013) Control of an aerial robot with multi-link arm for assembly tasks. In: *IEEE international conference on robotics and automation (ICRA)*, Karlsruhe, Germany, 6–10 May, pp. 4916–4921. IEEE.

Knepper RA, Layton T, Romanishin J and Rus D (2013) Ikeabot: An autonomous multi-robot coordinated furniture assembly system. In: *IEEE international conference on robotics and automation (ICRA)*, Karlsruhe, Germany, 6–10 May, pp. 855–862. IEEE.

Komendera E (2014) *Precise assembly of truss structures by distributed robots*. PhD Thesis, University of Colorado, USA.

Komendera E and Correll N (2014) Precise assembly of 3D truss structures using EKF-based error prediction and correction. In: Hsieh MA, Kumar V and Khatib O (eds) *International symposium on experimental robotics*. Berlin Heidelberg: Springer-Verlag.

Komendera E, Dorsey JT, Doggett WR and Correll N (2014a) Truss assembly and welding by intelligent precision jigging robots. In: *6th annual IEEE international conference on technologies for practical robot applications (TEPRA)*, Woburn, USA, 14–15 April.

Komendera E, Reishus D, Dorsey JT, Doggett WR and Correll N (2014b) Precise truss assembly using commodity parts and low precision welding. *Intelligent Service Robotics* 7(2): 93–102.

Lazzerini B and Marcelloni F (2000) A genetic algorithm for generating optimal assembly plans. *Artificial Intelligence in Engineering* 14: 319–329.

Lee S and Moradi H (1999) Disassembly sequencing and assembly sequence verification using flow networks. In: *International conference on robotics and automation*, Detroit, USA, 10–15 May. Red Hook: Curran Associates.

Lillie CF (2006) On-orbit assembly and servicing of future space observatories. *Proceedings of SPIE* 6265(62652D-1): 1–12.

Lindsey Q and Kumar V (2013) Distributed construction of truss structures. In: Frazzoli E, Lozano-Perez T, Roy N and Rus D (eds) *Algorithmic Foundations of Robotics X*. Berlin Heidelberg: Springer-Verlag, pp. 209–225.

Mathews JH and Fink KD (2004) *Numerical methods using MATLAB*. New York: Pearson.

McEvoy MA, Komendera E and Correll N (2014) Assembly path planning for stable robotic construction. In: *Sixth annual IEEE international conference on technologies for practical robot applications*, Woburn, USA, 14–15 April.

Napp N and Nagpal R (2014) Distributed amorphous ramp construction in unstructured environments. *Robotica* 32(02): 279–290.

Nigl F, Li S, Blum JE and Lipson H (2013) Structure-reconfiguring robots: Autonomous truss reconfiguration and manipulation. *Robotics & Automation Magazine, IEEE* 20(3): 60–71.

Röhrdanz F, Mosemann H and Wahl FM (1996) High lap: A high level system for generating, representing and evaluating assembly sequences. In: *International joint symposia on intelligence and systems*, Rockville, USA, 4–5 November, pp. 134–141. IEEE.

Simmons R, Singh S, Hershberger D, Ramos J and Smith T (2001) First results in the coordination of heterogeneous robots for large-scale assembly. In: Rus D and Singh S (eds) *Experimental Robotics VII*. Berlin Heidelberg: Springer-Verlag, pp. 323–332.

Stroupe A, Huntsberger T, Kennedy B, Aghazarian H, Baumgartner E, Ganino A, Garrett M, Okon A, Robinson M and Townsend J (2005a) Heterogeneous robotic systems for assembly and servicing. In: Battrick B (ed) *Eighth international symposium on artificial intelligence, robotics and automation in space, ESA, volume 603*. Noordwijk: ESA Publications, p. 82.

Stroupe A, Huntsberger T, Okon A and Aghazarian H (2005b) Precision manipulation with cooperative robots. In: Schultz AC, Parker LE and Schneider FE (eds) *Multi-robot systems volume III: From swarms to intelligent automata*. Dordrecht: Springer, pp. 235–248.

Sundaram S, Remmler I and Amato NM (2001) Disassembly sequencing using a motion planning appproach. In: *International conference on robotics and automation*, Seoul, South Korea, 21–26 May. Red Hook: Curran Associates.

Thrun S and Montemerlo M (2006) The graph slam algorithm with applications to large-scale mapping of urban structures. *The International Journal of Robotics Research* 25(5-6): 403–429.

Vona M, Detweiler C and Rus D (2008) Shady: Robust truss climbing with mechanical compliances. In: Khatib O, Kumar V and Rus D (eds) *Experimental Robotics*. Berlin Heidelberg: Springer-Verlag, pp. 431–440.

Watson JJ, Collins TJ and Bush HG (2002) A history of astronaut construction of large space structures at NASA Langley research center. In: *Aerospace conference*, *volume 7*, Big Sky, USA, 9–16 March, pp. 7–3569. Piscataway: IEEE.

Wawerla J, Sukhatme GS and Mataric MJ (2002) Collective construction with multiple robots. In: *IEEE/RSJ international conference on intelligent robots and systems*, *volume 3*, Lausanne, Switzerland, 20 September–4 October, pp. 2696–2701. IEEE.

Werfel J, Petersen K and Nagpal R (2014) Designing collective behavior in a termite-inspired robot construction team. *Science* 343(6172): 754–758.

Wilson RH (1992) *On geometric assembly planning*. PhD Thesis, Stanford University, USA.

Wolter J (1989) On the automatic generation of assembly plans. In: *International conference on robotics and automation*, Scottsdale, USA, 14–19 May, pp. 62–68. Red Hook: Curran Associates.

Worcester J, Lakaemper R and Hsieh MyA (2012) 3-dimensional tiling for distributed assembly by robot teams. In: Desai JP, Dudek G, Khatib O and Kumar V (eds) *13th international symposium on experimental robotics (ISER2012)*. Cham: Springer, pp. 143–154.

Yun Sk (2010) *Coordinating construction by a distributed multi-robot system*. PhD Thesis, Massachusetts Institute of Technology, USA.

Yun Sk, Hjelle DA, Schweikardt E, Lipson H and Rus D (2009) Planning the reconfiguration of grounded truss structures with truss climbing robots that carry truss elements. In: *IEEE international conference on robotics and automation (ICRA'09)*, Kobe, Japan, 12–17 May, pp. 1327–1333. IEEE.

Yun S, Schwager M and Rus D (2011) Coordinating construction of truss structures using distributed equal-mass partitioning. In: Predalier C, Siegwart G and Hirzinger G (eds) *Robotics Research*. Berlin Heidelberg: Springer-Verlag, pp. 607–623.

Zimpfer D, Kachmar P and Tuohy S (2005) Autonomous rendezvous, capture and in-space assembly: past, present and future. In: *1st space exploration conference: Continuing the voyage of discovery*, *volume 1*, Orlando, USA, 30 January–1 February, pp. 234–245. Red Hook: Curran Associates.

Appendix: Calculating the node position from base position and strut length

Equation 1 can be solved for X_f analytically as follows, where X_{j-i} is shorthand for $X_j - X_i$

$$\begin{aligned}
X_f = \quad & X_i + \frac{X_{j-i}\left(L_{i,f}^2 - L_{j,f}^2 + X_{j-i}.X_{j-i}\right)}{2X_{j-i}.X_{j-i}} \\
& + \left(\frac{X_{j-i} \times X_{k-i}}{\sqrt{X_{j-i} \times X_{k-i}.X_{j-i} \times X_{k-i}}} \times \frac{X_{j-i}}{\sqrt{X_{j-i}.X_{j-i}}}\right) \\
& \left(\frac{\left(L_{j,f}^2 - L_{i,f}^2\right)X_{j-i}.X_{k-i} + X_{j-i}.X_{j-i}\left(L_{i,f}^2 - L_{k,f}^2 - X_{j-i}.X_{k-i} + X_{k-i}.X_{k-i}\right)}{2X_{j-i}.X_{j-i}\sqrt{X_{k-i}.X_{k-i} - \frac{\left(X_{j-i}.X_{k-i}\right)^2}{X_{j-i}.X_{j-i}}}}\right) \\
& + \left(\frac{L_{i,f}\sqrt{1 - \frac{\left(L_{i,f}^2 - L_{j,f}^2 + X_{j-i}.X_{j-i}\right)^2}{4L_{i,f}^2 X_{j-i}.X_{j-i}}}X_{j-i} \times X_{k-i}}{\sqrt{X_{j-i} \times X_{k-i}.X_{j-i} \times X_{k-i}}}\right) \\
& \left(\sqrt{1 - \frac{\left(\left(L_{i,f}^2 - L_{j,f}^2\right)X_{j-i}.X_{k-i} + X_{j-i}.X_{j-i}\left(-L_{i,f}^2 + L_{k,f}^2 + X_{j-i}.X_{k-i} - X_{k-i}.X_{k-i}\right)\right)^2}{\left(\left(X_{j-i}.X_{k-i}\right)^2 - X_{j-i}.X_{j-i}X_{k-i}.X_{k-i}\right)\left(-2L_{j,f}^2\left(L_{i,f}^2 + X_{j-i}.X_{j-i}\right) + \left(X_{j-i}.X_{j-i} - L_{i,f}^2\right)^2 + L_{j,f}^4\right)}}\right)
\end{aligned} \tag{23}$$