Best of two worlds: Cartesian sampling and volume computation for high dimensional configuration spaces using Cayley coordinates

Meera Sitharam [*, †] Yichi Zhang[*]

October 3, 2022

1 Abstract

We give an algorithm that uniformly samples and computes a discrete volume measure of a space of configurations of rigid bodies satisfying a system of distance inequality constraints belonging to a large natural class that occurs in several application scenarios. The algorithm views the configuration space as a branched covering and uses a recent theory of Cayley or distance coordinates to convexify the base space. By employing an on-demand grid traversal datastructure, the algorithm runs in linear time and empirically sublinear space in the number of grid cubes that are used to define the discrete volume measure and that intersect the configuration space. A software implementation and comparison with existing methods is provided.

2 Introduction

A common type of configuration space is a feasible region of a distance constraint system between rigid bodies, i.e., consisting of configurations of a finite collection of *rigid bodies* in $\mathbb{R}^d$ that satisfy a distance constraint system (equalities or inequalities) between points in different rigid bodies. A *rigid body* is an equivalence class of point sets modulo a group of isometries, which, in the case of Euclidean distance constraint systems, typically consists of rotations and translations. Examples of such configuration spaces occur in the study of kinematic mechanisms, (underconstrained) mechanical CAD designs, molecular or particle assemblies, metamaterials, etc.

2.1 Preliminaries and Background

A typical *distance constraint system* is specified by a finite set S of rigid bodies together with a constraint graph G each of whose vertices v is a point on the rigid body $S(v)$ and whose edges represent distance (interval) constraints. The variables are the *Cartesian* orientations T_X for $X \in S$, given by $\binom{d+1}{2}$ scalars specifying X's rotation and translation relative to some fixed $O \in S$, where T_O is assumed to be the identity. The entire constraint system ($\mathbf{C}$) is specified as follows.

- **(C1)** For every pair (A, B) in S and every pair of points $a \in A$ and $b \in B$, $||T_A(a) - T_B(b)|| \geq l(a, b)$,
- **(C2)** For every edge $(a, b) \in G$, $l(a, b) \leq ||T_{S(a)}(a) - T_{S(b)}(b)|| \leq h(a, b)$

Here h and l are given positive scalar functions. For the problems considered in this paper, we will assume – essentially without loss of generality – that (i) the limiting case where the interval size $h(a, b) - l(a, b)$ tends to 0 as this case is both more difficult and more interesting; and (ii) that G is independent and flexible in the sense of combinatorial rigidity [1] [2]. Furthermore, while this paper deals only with Euclidean distance constraints, the concepts generalize to non-Euclidean norm or even other metric distance constraints.

Typically, such Cartesian configuration spaces are topologically complex semi-algebraic subsets of a high dimensional ambient space (k rigid bodies give $m = (k-1)\binom{d+1}{2}$ ambient dimensions). Generically, when the constraints as in (C2) above satisfy $l = h$, the configuration space is a real algebraic (quadratic) variety of co-dimension $|E(G)|$. However, in the abovementioned applications, typically, the distance constraints are either unidirectional inequalities, or, if they are bidirectional then the specified intervals of allowable distances are typically "small" as indicated in the assumption above.

A common example is a discretized version of a so-called Lennard-Jones potential constraint between 2 atoms in assembling rigid molecules [3]. Such potentials encode a variety of types of weak interactions, including Van der Waals, hydrogen bonds, as well as electrostatic, hydrophobic, hydrophilic as well as quantum-level interactions.

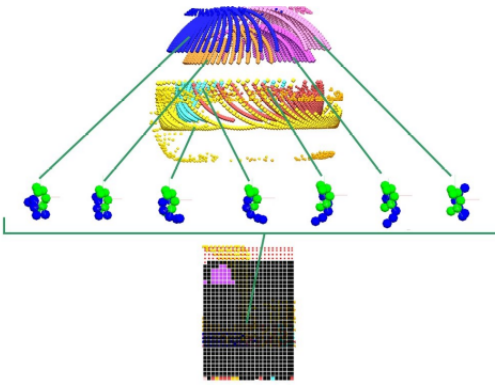
Each such interval constraint effectively reduces the dimension of the configuration space by one so that we are dealing with a configuration space that is effectively of much smaller dimension than the ambient dimension. For example $m - 1$ (resp. $m - 2$) such "small" interval constraints would generically yield a configuration space that is a "thick" curve (resp. sheet) in the ambient m dimensional space, with the thickness tending to 0 in the limiting case. See examples in Figure 1a.

Whether dealing with distance equality or inequality constraints, the task of traversing or sampling configurations while staying within the effectively lower dimensional and topologically complex Cartesian configuration space is typically achieved using an onerous type of "gradient descent", i.e. repeated linear tangential steps (e.g. by computing the Jacobian of the distance map of the constraint system), alternating with projections / corrections back to the feasible region. When prevailing methods are used, including molecular dynamics or Monte Carlo based methods, this gradient descent process pervades many common tasks such as finding optimal or extremal configurations, finding paths, path lengths, region volumes (configurational entropy), path probabilities for transition networks, sampling configurations or paths etc.

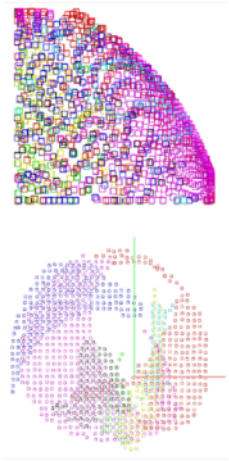
2.2 Previous Work

The papers [4] [5], [6], [7], [8] [9] [10] solved the problem of traversing or sampling – while staying within – a distance constrained, Cartesian configuration space, by treating it as

[*]CISE Department, University of Florida, partially supported by NSF DMS 1564480, NSF DMS 1563234, and DARPA HR00111720031.
[†]Affiliate in Department of Mathematics



(a) Top: Two dimensional Cartesian configuration space R of a distance constraint system in dimension 3, nonuniformly sampled; colors represent different flips of R viewed as a branched covering space. Middle: some sampled feasible configurations of rigid bodies satisfying distance constraints of the type (C1) and (C2), one from each flip Bottom: corresponding convex base space with uniform sampling in Cayley coordinates



(b) Bottom: Uniform Cartesian sampling, colors represent different flips of another two dimensional configuration space R (also of a distance constraint system in dimension 3 not shown) as a branched covering space. Top: Corresponding (nonuniform) sampling of the convex base space in Cayley coordinates, note that some Cayley configurations have multiple colors - each such configuration has preimages belong to multiple flips

a *branched covering space* . More precisely, a configuration space in m ambient dimensions, defined as in (C2) above by a constraint graph G of $|E(G)|$ generically independent distance (interval) constraints, is mapped by a *covering map* to a *base space* that is a subset of the space spanned by $m-|E|$ *Cayley* coordinates. A *Cayley coordinate* is an unconstrained pairwise squared distance associated with a nonedge in the distance constraint graph. A *Cayley configuration* is a tuple of $m-|E|$ Cayley coordinate values; the base space consists of Cayley configurations. By definition of a covering map, the pre-image of a Cayley configuration is finite, i.e. has generically at most finitely many feasible configurations mapped to it by the covering map. Accordingly, the pre-image of the base space is the union of finitely many almost disjoint "sheets" or *flips*, which are branches of the covering space. The flips or branches may intersect on a set of dimension strictly smaller than $m-|E|$. See Figure 1a.

For dimension $d \leq 3$ and a substantial *nice* class $\mathcal{C}_d$ of distance (interval) constraint graphs G, [4][10][5] used the properties of the cone of Euclidean squared pairwise distances of a point set [11] (generalizable to other norms [12]) to show the following properties: (1) there is a covering map π_G given by chosen Cayley coordinates or non-edges of G, guaranteeing a convex base space (consisting of feasible Cayley configurations with nonempty pre-images of the covering map); (2) for a Cayley configuration in the base space, computing the pre-image configurations of the covering map has linear output complexity; and (3) determining whether a Cayley configuration is feasible has linear time complexity in the problem size i.e. the number of points specifying the rigid bodies. We note that the original paper [4] states these properties for a larger graph whose edges - interpreted in the context of this paper - include the distance constraints between point pairs within each rigid body. However the the graph G in papers [10][5] refers only to the distance (interval) constraints (C2) between rigid bodies.

Using these properties, we can efficiently traverse the effectively lower dimensional and topologically complex feasible configuration space R as follows. Traversing the base space of R is efficient by (1) and (3) above, and moreover a subset of the Cayley coordinate space, by definition; computing the pre-image configurations of the covering map is efficient by (2) above. This yields a traversal that does not leave the branched covering space R. Furthermore, the *boundaries* of the base space of R are explicitly detected and traversed. The boundaries represent two types of transitions: (i) the inequalities in (C1) and (C2) above become tight, or (ii) the real pre-image of the covering map becomes empty (the pre-image is complex); these are additionally the intersections of the branches or flips of the covering space.

The Cayley configuration methodology requires characterizing distance constraint graphs with convex base spaces of Cayley configurations. It draws upon a rich set of tools from graph rigidity, realization and distance geometry [2][1], generalizes to other norms [13] is closely related to a key finite forbidden minor property called flattenability of graphs [13][14][15], and leads to directions of independent interest to those areas. Furthermore, the methodology has been implemented as opensource software (EASAL [10][5] and CayMos [7] [6]) for respectively molecular and particle assembly modeling and kinematic mechanism analysis and design) and has led to several improvements in those areas, besides efficient algorithms for the core problem of distance constraint graph realization [16].

2.3 The Problems, the Obstacles and Current Approaches

While the Cayley coordinate representation significantly improves the efficiency of traversal, path finding, search for extremal configurations, etc. [10][5][17][18][7][6][8][9], for a configuration space specified by distance constraint graphs in the abovementioned class $\mathcal{C}_d$ of [4], it is not clear how to use it to sample the configuration space uniformly in the original *Cartesian* coordinates of the ambient space, or to compute volume measures (or path lengths), a frequent and important task for configurational entropy and free energy computations [19][20][21]. Such volume definitions are based on a Cartesian coordinate grid.

Problem 1 is to compute the ϵ-approximate *volume* of a (feasible) configuration space R in ambient m-dimensional Cartesian coordinate space, defined as the relative proportion of m-dimensional hypercubes of side length ϵ that intersect R (in a generic rectilinear grid subdivision of the

ambient space). **Problem 2** is to generate one point on R per intersecting hypercube, i.e. a uniform Cartesian sampling of configurations in R. Both problems assume distance constraint graphs in the nice class $\mathcal{C}_d$.

Clearly Problem 1 reduces to Problem 2 but it is conceivable that it could be solved directly. As we explain the approaches to Problem 2 below, we note the reasons why, in current practice, Problem 1 is solved essentially by solving Problem 2.

One obvious way (*) to try to solve both problems is to use the known efficient method [10] [5][7][6] for uniform sampling in Cayley coordinates of R's convex base space together with preimage computations as described above, since R's distance constraint graph is in the nice class $\mathcal{C}_d$. However, with no adjustments, this results in a highly nonuniform sampling of R, i.e. an unsatisfactory solution to Problem 2, see Figure 1b.

In fact, this is a decades-old problem in computational chemistry, referred to as "internal coordinate to Cartesian back transformation", that continues to be actively studied [22][23][24]. To clarify, the "internal coordinates" used in computational chemistry, e.g. in molecular dynamics, are different from Cayley coordinates, and to the best of our knowledge lack the underlying theory and tools available for Cayley coordinates.

The straightforward workaround is to traverse the base space of R in Cayley coordinates, but iteratively adjust the size of each Cayley step by computing a pseudoinverse of the linearization (Jacobian) of the covering map and ensuring uniform Cartesian sampling of the pre-image branched covering space R. This approach suffers from both inaccuracies due to linearization error as well as illconditioning problems [25]. A standard way to address these problems is to use the Hessian and higher derivatives of the covering map. However, such efforts are still underway[22][23][24]and the problem is by no means settled.

It should be noted that the use of higher derivatives of the covering map theoretically provides another approach to Problem 1 directly without Problem 2. I.e., one could avoid uniform Cartesian sampling of the configuration space R, but rather use the convexity advantage of the base space to compute its Cayley volume in polynomial time [26][27][28][29][30] using a random walk, while using the higher derivatives of the covering map to compute the volume of R and solve Problem 1. In any case, this too involves some form of randomized or deterministic adaptive sampling in Cayley coordinates. Furthermore, to our knowledge such an approach to Problem 1 that avoids Problem 2 does not exist in the literature. One reason could be that although the covering map is quite well behaved, this cannot be said about the pseudoinverses of its Jacobian or Hessian.

Our contributions provide an optimal solution to Problem 2 and thereby a solution to Problem 1.

3 Contributions

(1) The first contribution is an algorithm *Uniform Cartesian* that solves Problem 2 when the graph G is in the above-mentioned class $\mathcal{C}_d$ for $d \leq 3$ (as characterized in [4][5]) in time linear in the output size, i.e. in the number of ϵ-cubes that intersect R. This is optimal (and nontrivial) since R is a topologically complex, effectively lower dimensional subset of the ambient space. In addition to leveraging the known efficient method (*) for sampling the base space of R in Cayley coordinates [10][5], our algorithm is inspired by a slicing algorithm for 3D printing very large objects filled with mapped (curved) microstructures [31].

(2) The second contribution is of independent interest: an on-demand grid traversal method that empirically (and intuitively) takes sublinear space in the number of grid cubes visited. This indicates sublinear space complexity (in terms of output size) for a modified Problems 1 and 2 that requires *at least* (instead of exactly) one point per grid hypercube that intersects R.

(3) The third contribution is an opensource software implementation of the above algorithm that is used to compare the performance of our method for Problem 1 using variants of the obvious method (*) described above, i.e., Cayley sampling according to 3 different distributions together with pre-image computations of the covering map. The implementation relies on efficient grid datastructures that could be of independent interest: they speed up the extraction of arbitrary dimensional facets and simplices and their intersection with the configuration space R.

3.1 Sketch of the Algorithm and Main Result

Input: a set of rigid bodies S, and constraints as in (C) with the constraint graph G in the class $\mathcal{C}_d$, $d \leq 3$ together with bounds l and h. These define the configuration space R. The required accuracy ϵ for Problems 1 and 2. For reasons of exposition and the current software implementation, we further assume $d = 3$ and $|S| = 2$ whereby the ambient dimension $m = 6$. Smaller d are subsumed and standard algorithmic extensions to $|S| > 2$ follow the description in [5]. We further assume the modified Problems 1 and 2 that require *at least* (instead of exactly) one point per grid hypercube that intersects R. A straightforward output datastructure with a hash map solves the original problems efficiently.

The algorithm has 4 parts.

1. using the covering map π_G given in [4], to *sample the base space* $\pi_G(R)$ in Cayley coordinates using the method in [5] that determines a Cayley step-size based on ϵ and finds boundaries and extremal configurations; further compute the corresponding pre-image Cartesian configurations s in R.

2. Using the Cartesian ϵ-grid hypercube containing s as a starting point, *generate hypercubes p on-demand, and traverse using a key frontier hypercube datastructure.*

3. Use the covering map π_G to generate the vertices of the Cayley cuboid $\pi_G(p)$, followed by its *linearization and intersection of its facets (decomposed into disjoint simplices) of dimension $|E(G)|$ with the convex base space* $\pi_G(R_2)$, of dimension $m - |E(G)|$; here R_2 is the set of configurations satisfying the constraints $(C2)$ and $R \subseteq R_2$; this generates partly feasible Cayley configurations c.

4. Compute the pre-image configurations $\pi_G^{-1}(c)$, retain only if fully feasible, i.e. only if (C1) is satisfied, and *find and count the corresponding Cartesian grid cube* p'. Note that due to linearization error, p may may differ from p'. This solves the modified Problems 1 and 2. A straightforward output datastructure stores the cubes p' and locates them with a hash map to avoid doublecounting. This solves the original unmodified problems.

3.1.1 The Frontier Hypercube Graph Datastructure of Step 2

We describe the key Step 2 above: this is achieved by a frontier m-dimensional hypercube graph datastructure that

maintains a graph whose vertices represent those inspected-but-unprocessed hypercubes and whose neighbors are some of their $(m-1)$-dimensional face neighbors as described below. The graph's hypercubes/vertices are partitioned into two sets: P for promising inspected hypercubes yet to be processed (i.e. intersections and pre-images yet to be computed as in Steps 3 and 4), and Q for similarly inspected and unprocessed candidate frontier hypercubes that are not yet promising. Processed hypercubes are not stored, but all of their face-neighbor hypercubes are guaranteed to be in P or Q or already processed. An unprocessed hypercube is in P if its shared face with one of its processed neighbors contains a $|E(G)| \leq m$ dimensional facet with a valid intersection and pre-image in R. An unprocessed hypercube is in Q if at least one of its face neighbors has been processed, but it is not in P.

Each face of a hypercube in P and Q is given one of 3 labels. Those it shares with processed cubes, those it shares with uninspected cubes (neither of these are edges in the frontier hypercube graph datastructure), and those it shares with unprocessed cubes, which are edges of the frontier graph datastructure.

Note that P and Q could contain disconnected components and even singleton hypercube/vertices (all of whose face neighbors have either been processed or have not been inspected). Furthermore, faces corresponding to edges between two hypercubes in P could already contain $E(G)$-dimensional facets that have yielded points in R: this is because such facets could additionally belong to faces shared with already processed cubes. However, faces corresponding to edges incident on any hypercube in Q cannot contain such a facet.

The main operations of Step 2 are the following. A hypercube c with the maximum number of processed face neighbors is selected from P. As mentioned above, it is possible that at the time c is chosen from P to be processed, in fact all of c's faces were shared with previously processed hypercubes, in which case, there is nothing further to be done and *c is removed from the datastructure.*

It is also possible that although some of c's faces were shared with unprocessed hypercubes in P or Q, or uninspected hypercubes, all of c's $E(G)$ dimensional facets could have already been processed. I.e. there is no Step 3 or 4 to be done at the time c is chosen to be processed. In any case, faces corresponding to c's uninspected or unprocessed face neighbor hypercubes are processed one face at a time. Effectively any of c's $E(G)$-dimensional facets not shared with processed cubes are now processed. The faces corresponding to c's unprocessed neighbors in P or Q are processed and the neighbors' shared faces with c are appropriately relabeled. Some hypercubes could move from Q to P. Any of c's uninspected face neighbors that were previously not in the frontier hypercube datastructure are now added to P or Q appropriately labeling those shared faces. Then *c and its edges are removed from the frontier graph datastructure.*

The algorithm ends when P is empty. The key property of this datastructure is that a hypercube c can be removed as soon as it is processed without compromising the traversal.

3.1.2 Main Result: Key Observations

The use of the frontier hypercube datastructure as described above ensures that the algorithm inspects *all* the Cartesian hypercubes neighboring those hypercubes that intersect R and starts with an intersection point in R. If R in the above sentence were replaced by R_2, i.e., without the (C1) constraints, the convexity of the base space $\pi_G^{-1}(R_2)$ would ensure that the algorithm does not miss any Cartesian hypercubes that intersect R_2 and therefore R. Step 1 deals with any discontinuity or other violation of convexity in $\pi_G^{-1}R$ arising from the constraints (C1), by including as starting points of the traversal at least one (boundary) hypercube in every component of R (in a minimal decomposition of R into convex regions).

Further, since *only* neighbors of R-intersecting hypercubes are inspected, the number of inspected hypercubes that do *not* intersect R is bounded by a constant $(2d)$ factor of the number of intersecting hypercubes.

The above observations ensure Contribution 1. As noted in the previous section, a further optimization in the frontier hypercube datastructure ensures that inspected hypercubes - which are now in the "interior" of the traversal region - are immediately - and safely - deleted from the frontier datastructure, ensuring Contribution 2.

The next section describes Contribution 3.

3.2 Computational Experiments and Results

3.2.1 Setup

The software for the new algorithm *UC (uniform Cartesian)* was implemented atop an existing curated opensource suite EASAL (Efficient Atlasing and Search of Assembly Landscapes, and hence denoted *EASAL-UC*. software available at `http://bitbucket.org/geoplexity/easal_dev`; see also video `https://cise.ufl.edu/~sitharam/EASALvideo.mpeg`, and user guide `https://bitbucket.org/geoplexity/easal/src/master/CompleteUserGuide.pdf`). Although EASAL is suited to full-fledged parallel processing, the experiments presented here are merely for proof-of-concept and were run on a single core of an Intel Core(TM) i7-8700K CPU@ 3.70 GHz CPU with a total estimated memory usage of 100MB.

The experiments compare EASAL-UC with comparator methods w.r.t. their performance on Problems 1 and 2 on benchmark Cartesian configuration spaces R defined as follows. The set S consisted of $|S| = k = 2$ rigid bodies A and B in $d = 3$ with 6 points each. Thus the ambient dimension $m = \binom{4}{2} = 6$. Our distance constraint systems (C) are defined by first assigning every point p a "radius" r_p. The distance interval lower bound $l(a,b)$ in (C1) and (C2) is specified to be $0.75(r_a + r_b)$, and the distance interval upper bound $h(a,b)$ in (C2) is $(r_a + r_b)$. For the constraints in (C2), two different graphs G (both independent, containing 4 edges each belonging to the "nice" class $\mathcal{C}_3$ of [4][5]) were chosen. These give two different constraint systems (C) and correspondingly two different 2-dimensional feasible configuration spaces R (referred to as "configuration space 1" and "configuration space 2"), with appropriate covering maps and convex base spaces.

The branched covering space of Configuration space 1 has 7 available flips indexed 0 through 6, while configuration space 2 has 4 flips indexed 2, 3, 4, and 5. Choosing configuration space 1, flip 0 as control group, the Cartesian volume of all flips are calculated using EASAL-UC. The comparators are the known efficient methods EASAL-1/2/3 that perform the straightforward approach (*) described in Section 2.3. The ϵ for Problem 1 and 2 also determines the EASAL-UC Cartesian hypercubes, i.e. translational and rotational step sizes are set to 0.5 and $\pi/8$, respectively. EASAL-1 performs uniform sampling in Cayley coordinates, and its Cayley step size is set to 0.8. This ensured that EASAL-UC (194) and EASAL-1(191) generate similar number of samples for the control flip. For the sampling distributions chosen by EASAL-2 and EASAL-3, initial step size is set to 2 and 4 respectively. In addition a higher resolution (0.2 and $\pi/16$) ultra-fine grid is used as baseline/ground truth to compare

the other methods.

3.2.2 Key Measurements

We describe the key measurements used to compare EASAL-UC with the EASAL variants 1/2/3 for their performance on Problem 1 and 2.

Problem 1: Volume Computation Accuracy and Efficiency EASAL-UC computes volume of each flip by counting Cartesian hypercubes with at least one feasible configuration in them. EASAL 1/2/3 simply follow the efficient method (*) of Section 2.3, sample the base space in Cayley coordinates according to different distributions and compute the pre-image Cartesian configurations and count them to give a rough Cartesian volume approximation. The volume ratio against the control flip is calculated and compared. As a measurement of efficiency, time cost per feasible cube(in UC)/point(in EASAL) is calculated. The results are given in Tables 1 and 2 in the Appendix. Clearly, EASAL-UC vastly outperforms EASAL 1/2/3 however at at a significant expense w.r.t. practical efficiency although the formal complexity analysis indicates no difference, i.e. linear time complexity in the output size.

Problem 2: Uniform Coverage Accuracy and Efficiency Accuracy of coverage of a method M of a fine Cartesian grid is measured using the γ-coverage. An γ-cube is a cube with a baseline grid point as centre and 2γ as range in all 6 Cartesian dimensions.

To normalize methods with different number of samples, γ-coverage is defined as the percentage of baseline grid points *covered* by at least one sample point, i.e.that lies within a γ-cube of a sample point of Method M. Further, the value of γ is set to

$$\gamma := \lceil \tfrac{(\Gamma/\sigma)^{1/6}}{2} \rceil$$

where Γ is grid point count and σ is sample point count for Method M. Clearly, a more accurate solution to Problem 2 will have higher γ-coverage and EASAL-UC outperforms the comparators despite the fact that it is disadvantaged by the ceiling in the computation of γ because of which the γ-cubes were the same for all of the methods although EASAL 2/3 had many more samples as shown in Table 3 in the Appendix.

The efficiency of coverage is shown in the Appendix Figure 2 with the number of sample points μ that lie in an γ-cube against number of γ-cubes with μ sampled points in them. A more efficient method should have fewer points mapped to the same γ-cube. As is seen in the plot, all 4 methods have relatively good coverage efficiency.

References

[1] Jack E Graver, Brigitte Servatius, and Herman Servatius. *Combinatorial rigidity*. 2. American Mathematical Soc., 1993.

[2] Meera Sitharam, Audrey St John, and Jessica Sidman. *Handbook of geometric constraint systems principles*. Chapman and Hall/CRC, 2018.

[3] Sangjae Seo and Wataru Shinoda. "Molecular Dynamics Simulations". In: *Reference Module in Chemistry, Molecular Sciences and Chemical Engineering*. Elsevier, 2018. ISBN: 978-0-12-409547-2. DOI: 10.1016/B978-0-12-409547-2.14274-X.

[4] Meera Sitharam and Heping Gao. "Characterizing graphs with convex and connected cayley configuration spaces". In: *Discrete and Computational Geometry* 43 (3 2010), pp. 594–625. ISSN: 01795376. DOI: 10.1007/s00454-009-9160-8.

[5] Rahul Prabhu et al. "Atlasing of Assembly Landscapes using Distance Geometry and Graph Rigidity". In: *Journal of Chemical Information and Modeling* 60 (10 Oct. 2020), pp. 4924–4957. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.0c00763.

[6] Meera Sitharam and Menghan Wang. "How the Beast really moves: Cayley analysis of mechanism realization spaces using CayMos". In: *Computer-Aided Design* 46 (2014). 2013 SIAM Conference on Geometric and Physical Modeling, pp. 205–210. ISSN: 0010-4485. DOI: doi.org/10.1016/j.cad.2013.08.033.

[7] Menghan Wang and Meera Sitharam. "Algorithm 951: Cayley Analysis of Mechanism Configuration Spaces using CayMos: Software Functionalities and Architecture". In: *ACM Trans. Math. Softw.* 41.4 (2015), 27:1–27:8. DOI: 10.1145/2699462.

[8] Meera Sitharam, Menghan Wang, and Heping Gao. "Cayley configuration spaces of 2D mechanisms, Part I: extreme points, continuous motion paths and minimal representations". In: (Dec. 2011). URL: http://arxiv.org/abs/1112.6008.

[9] Meera Sitharam, Menghan Wang, and Heping Gao. "Cayley Configuration Spaces of 1-dof Tree-decomposable Linkages, Part II: Combinatorial Characterization of Complexity". In: (Dec. 2011). URL: http://arxiv.org/abs/1112.6009.

[10] Aysegul Ozkan et al. "Algorithm 990: Efficient atlasing and search of configuration spaces of point-sets constrained by distance intervals". In: *ACM Transactions on Mathematical Software* 44 (4 June 2018). ISSN: 15577295. DOI: 10.1145/3204472.

[11] I J Schoenberg. *Metric Spaces and Positive Definite Functions*. 1938, pp. 522–536.

[12] Keith Ball. "Isometric embedding in lp-spaces". In: *European Journal of Combinatorics* 11.4 (1990), pp. 305–311.

[13] Meera Sitharam and Joel Willoughby. "On Flattenability of Graphs". In: (Mar. 2015). URL: http://arxiv.org/abs/1503.01489.

[14] Maria Belk and Robert Connelly. "Realizability of graphs". In: *Discrete and Computational Geometry* 37 (2 2007), pp. 125–137. ISSN: 14320444. DOI: 10.1007/s00454-006-1284-5.

[15] Maria Belk. "Realizability of graphs in three dimensions". In: *Discrete & Computational Geometry* 37.2 (2007), pp. 139–162.

[16] Troy Baker et al. "Optimal Decomposition and Recombination of Isostatic Geometric Constraint Systems for Designing Layered Materials". In: *Computer Aided Geometric Design* 40 (July 2015). DOI: 10.1016/j.cagd.2015.07.001.

[17] Aysegul Ozkan et al. "Baseline Comparisons of Complementary Sampling Methods for Assembly Driven by Short-Ranged Pair Potentials toward Fast and Flexible Hybridization". In: *Journal of Chemical Theory and Computation* 17 (3 Mar. 2021), pp. 1967–1987. ISSN: 15499626. DOI: 10.1021/acs.jctc.0c00945.

[18] Ruijin Wu et al. “Rapid prediction of crucial hotspot interactions for icosahedral viral capsid self-assembly by energy landscape atlasing validated by mutagenesis”. In: *PLoS Computational Biology* 16 (10 Oct. 2020). ISSN: 15537358. DOI: 10.1371/journal.pcbi.1008357.

[19] Simon Hikiri, Takashi Yoshidome, and Mitsunori Ikeguchi. “Computational Methods for Configurational Entropy Using Internal and Cartesian Coordinates”. In: *Journal of Chemical Theory and Computation* 12 (12 2016), pp. 5990–6000. ISSN: 15499626. DOI: 10.1021/acs.jctc.6b00563.

[20] Gergely Gyimesi, Péter Závodszky, and András Szilágyi. “Calculation of configurational entropy differences from conformational ensembles using Gaussian mixtures”. In: *Journal of Chemical Theory and Computation* 13 (1 2017), pp. 29–41. ISSN: 15499626. DOI: 10.1021/acs.jctc.6b00837.

[21] Federico Fogolari et al. “Distance-based configurational entropy of proteins from molecular dynamics simulations”. In: *PLoS ONE* 10 (7 2015), pp. 1–26. ISSN: 19326203. DOI: 10.1371/journal.pone.0132356.

[22] Jon Baker, Don Kinghorn, and Peter Pulay. “Geometry optimization in delocalized internal coordinates: An efficient quadratically scaling algorithm for large molecules”. In: *The Journal of Chemical Physics* 110.11 (1999), pp. 4986–4991. DOI: 10.1063/1.478397.

[23] Vladimir V. Rybkin, Ulf Ekström, and Trygve Helgaker. “Internal-to-Cartesian back transformation of molecular geometry steps using high-order geometric derivatives”. In: *Journal of Computational Chemistry* 34.21 (2013), pp. 1842–1849. DOI: 10.1002/jcc.23327.

[24] Jie Li et al. *Learning Correlations between Internal Coordinates to improve 3D Cartesian Coordinates for Proteins*. 2022. DOI: 10.48550/ARXIV.2205.04676. URL: https://arxiv.org/abs/2205.04676.

[25] Aysegul Ozkan and Meera Sitharam. “Best of Both Worlds: Uniform sampling in Cartesian and Cayley Molecular Assembly Configuration Space”. In: (2014), pp. 7–10. URL: http://arxiv.org/abs/1409.0956.

[26] Martin Dyer, Alan Frieze, and Ravi Kannan. “A Random Polynomial-Time Algorithm for Approximating the Volume of Convex Bodies”. In: *J. ACM* 38.1 (Jan. 1991), pp. 1–17. ISSN: 0004-5411. DOI: 10.1145/102782.102783.

[27] David Applegate and Ravi Kannan. “Sampling and Integration of near Log-Concave Functions”. In: *Proceedings of the Twenty-Third Annual ACM Symposium on Theory of Computing*. STOC ’91. New Orleans, Louisiana, USA: Association for Computing Machinery, 1991, pp. 156–163. ISBN: 0897913973. DOI: 10.1145/103418.103439.

[28] Ravi Kannan, László Lovász, and Miklós Simonovits. “Random walks and an O*(n5) volume algorithm for convex bodies”. In: *Random Structures & Algorithms* 11.1 (1997), pp. 1–50. DOI: 10.1002/(SICI)1098-2418(199708)11:1<1::AID-RSA1>3.0.CO;2-X.

[29] László Lovász and Santosh Vempala. “Simulated annealing in convex bodies and an O*(n4) volume algorithm”. In: *Journal of Computer and System Sciences* 72.2 (2006). JCSS FOCS 2003 Special Issue, pp. 392–417. ISSN: 0022-0000. DOI: 10.1016/j.jcss.2005.08.004.

[30] Cunjing Ge and Feifei Ma. “A Fast and Practical Method to Estimate Volumes of Convex Polytopes”. In: *FAW*. 2015.

[31] Jeremy Youngquist, Meera Sitharam, and Jörg Peters. “A Slice-Traversal Algorithm for Very Large Mapped Volumetric Models”. In: *Computer-Aided Design* 141 (2021), p. 103102. ISSN: 0010-4485. DOI: 10.1016/j.cad.2021.103102.

Appendix

Table 1 shows result of relative volume calculation using aforementioned method. The best method should have relative volume close to the baseline result. As we can see EASAL-UC is the only method presenting a reasonable accuracy for Problem 1.

Table 1: Relative volume against configuration space 1, flip 0

Node name	Flip num	Baseline	UC	EASAL-1	EASAL-2	EASAL-3
configuration space 1	1	1.656	1.830	0.995	0.669	1.021
configuration space 1	2	0.772	0.747	0.545	0.215	1.541
configuration space 1	3	1.075	1.093	0.539	0.214	1.299
configuration space 1	4	0.931	1.036	0.749	0.278	1.480
configuration space 1	5	0.975	0.964	0.728	0.270	1.685
configuration space 1	6	1.240	1.438	0.843	0.645	0.922
configuration space 2	2	1.353	1.459	0.754	2.000	1.428
configuration space 2	3	1.681	1.768	0.749	1.977	1.489
configuration space 2	4	1.708	1.773	0.764	1.991	1.515
configuration space 2	5	1.417	1.500	0.696	2.050	1.339

Table 2 shows time spent on the sampling and calculation process. UC spends more than 1000x time per sample as a trade-off to achieve a precise result for Problem 1.

Table 2: Time consumption per sample (ms)

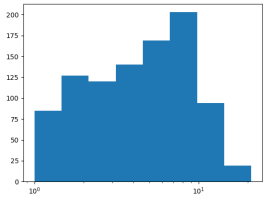
Node name	UC time per point	UC time per cube	EASAL-1	EASAL-2	EASAL-3
configuration space 1	158.348	887.476	0.568	0.529	0.507
configuration space 2	137.956	1451.229	1.037	0.303	0.341

Table 3 shows percentage of γ-cubes covered by different methods. With similar number of samples, fewer grid cubes are covered in EASAL-1; while EASAL-2/3 used more samples but achieve lower coverage rate. This clearly shows EASAL-UC's sampling accuracy for Problem 2.

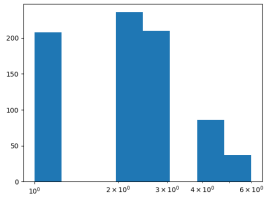
Table 3: Comparison of different methods on Uniform Coverage

Sampling method	Sample count	γ - value	Coverage rate
UC	194	$\lceil 0.659 \rceil$	94.8%
EASAL-1	191	$\lceil 0.661 \rceil$	76.3%
EASAL-2	659	$\lceil 0.537 \rceil$	73.1%
EASAL-3	425	$\lceil 0.578 \rceil$	90.6%

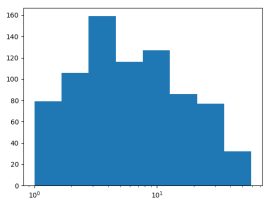
Figure 2 plots the number of sample points μ that lie in an γ-cube against number of γ-cubes with μ sampled points in them. A more efficient method should have fewer points mapped to the same γ-cube. As is seen in the plot, all 4 methods have relatively good coverage efficiency.



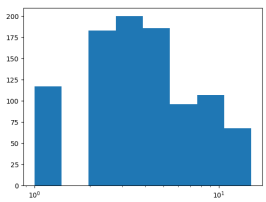
(a) Uniform Cartesian



(b) EASAL-1



(c) EASAL-2



(d) EASAL-3

Figure 2: Horizontal axis showing the number of sample points ν that lie in γ-cubes and vertical axis showing the number of γ-cubes having ν mapped points inside of them.