Graph Realization and Low-Rank Matrix Completion

Mihai Cucuringu

A Dissertation

Presented to the Faculty

of Princeton University

in Candidacy for the Degree

of Doctor of Philosophy

Recommended for Acceptance

by the Program in

Applied and Computational Mathematics

Adviser: Amit Singer

June 2012

Abstract

This thesis consists of five chapters, and focuses on two main problems: the graph realization problem with its applications to localization of sensor network and structural biology, and the low-rank matrix completion problem. Chapter 1 is a brief introduction to rigidity theory and supplies the background needed for the subsequent chapters. Chapter 2 introduces the graph realization problem in dimension two, and its application to sensor network localization. Chapter 3 considers the three dimensional graph realization problem and its application to the molecule problem from structural biology. Chapter 4 focuses on the group synchronization problem, and provides a more in-depth analysis of the synchronization methods used in our algorithms for the graph realization problem in $\mathbb{R}^2$ and $\mathbb{R}^3$. Finally, Chapter 5 investigates the problem of uniqueness of low-rank matrix completion, building on tools from rigidity theory.

Rigidity theory tries to answer if a given partial set of distances $d_{ij} = \|p_i - p_j\|$ between n points in $\mathbb{R}^d$ uniquely determines the coordinates of the points $p_1, \ldots, p_n$ up to rigid transformations (translations, rotations, reflections). Chapter 1 is a self contained but extremely selective and incomplete collection of basic definitions and results from the rigidity theory literature.

The graph realization problem has received a great deal of attention in recent years, due to its importance in applications such as wireless sensor networks and structural biology. In Chapter 2, we present a new approach to localization of sensors from noisy measurements of a subset of their Euclidean distances. Our algorithm starts by finding, embedding and aligning uniquely realizable subsets of neighboring sensors called patches. In the noise-free case, each patch agrees with its global positioning up to an unknown rigid motion of translation, rotation and possibly reflection. The reflections and rotations are estimated using the recently developed eigenvector synchronization algorithm, while the translations are estimated by solving an overdetermined linear system. The algorithm is scalable as the number of nodes increases, and can be implemented in a distributed fashion. Extensive numerical experiments show that it compares favorably to other existing algorithms in terms of robustness to noise, sparse connectivity and running time.

In Chapter 3, we extend on previous work and propose the 3D-ASAP algorithm, for the graph realization problem in $\mathbb{R}^3$, given a sparse and noisy set of distance measurements. 3D-ASAP is a divide and conquer, non-incremental and non-iterative algorithm, which integrates local distance information into a global structure determination. Our approach starts with identifying, for every node, a subgraph of its 1-hop neighborhood graph, which can be accurately embedded in its own coordinate system. In the noise-free case, the computed coordinates of the sensors in each patch must agree with their global positioning up to some unknown rigid motion, that is, up to translation, rotation and possibly reflection. In other words, to every patch there corresponds an element of the Euclidean group Euc(3) of rigid transformations in $\mathbb{R}^3$, and the goal is to estimate the group elements that will properly align all the patches in a globally consistent way. Furthermore, 3D-ASAP successfully incorporates information specific to the molecule problem in structural biology, in particular information on known substructures and their orientation. In addition, we also propose 3D-SP-ASAP, a faster version of 3D-ASAP, which uses a spectral partitioning algorithm

as a preprocessing step for dividing the initial graph into smaller subgraphs. Our extensive numerical simulations show that 3D-ASAP and 3D-SP-ASAP are very robust to high levels of noise in the measured distances and to sparse connectivity in the measurement graph, and compare favorably to similar state-of-the art localization algorithms.

Chapter 4 is a self contained analysis of the group synchronization problem and its variations. Finding group elements from noisy measurements of their ratios is also known as the synchronization problem. The eigenvector method was originally introduced for solving the synchronization problem over the group SO(2) of planar rotations, which is one of the building blocks for our ASAP algorithms. In this chapter we motivate the robustness to noise of the eigenvector synchronization method for the groups $\mathbb{Z}_2$ and SO(2) using tools from spectral graph theory. In the case of synchronization over SO(3) we show that, in the noise free case, the top three eigenvectors of the incomplete matrix of pairwise group measurements perfectly recover the unknown group elements. In addition, we give an analysis of different approaches to the synchronization problem over $\mathbb{Z}_2$ with anchor information, which is useful for incorporating molecular fragment information into the molecule problem.

Finally, Chapter 5 investigates the problem of uniqueness of low-rank matrix completion. This problem, of completing a low-rank matrix from a subset of its entries, is often encountered in the analysis of incomplete data sets exhibiting an underlying factor model with applications in collaborative filtering, computer vision and control. Most recent work had been focused on constructing efficient algorithms for exact or approximate recovery of the missing matrix entries and proving lower bounds for the number of known entries that guarantee a successful recovery with high probability. A related problem from both the mathematical and algorithmic point of view is the distance geometry problem of realizing points in a Euclidean space from a given subset of their pairwise distances. Rigidity theory answers basic questions regarding the uniqueness of the realization satisfying a given partial set of distances. We observe that basic ideas and tools of rigidity theory can be adapted to determine uniqueness of low-rank matrix completion, where inner products play the role that distances play in rigidity theory. This observation leads to efficient randomized algorithms for testing necessary and sufficient conditions for local completion and for testing sufficient conditions for global completion. Crucial to our analysis is a new matrix, which we call the *completion matrix*, that serves as the analogue of the rigidity matrix.

Acknowledgements

First and foremost, I would like to thank my graduate adviser, Amit Singer, for his academic guidance and personal support over the years. It is hard to imagine my work at Princeton had I not had the chance to work with Amit. I thank him for suggesting interesting and challenging problems, and for his confidence in my abilities. His patience, valuable insights and guidance helped me improve my work over the years, and made this thesis possible. A great deal of what I learned at Princeton, I learned from Amit. I feel very fortunate and proud to have been his student, and I hope that one day I will be able to provide my own students the same guidance and support I have received from Amit.

I would also like to thank a number of other people whose help and support made this thesis possible, directly or indirectly. I would like to thank my thesis readers and committee members, Robert Calderbank, Ioannis Kevrekidis and Paul Seymour. I am very thankful to all of them for their support and guidance, and the many useful discussions we had throughout the years. I thank David Cowburn and Yaron Lipman for their support and collaboration on the sensor networks and the molecule problem projects. I thank Vincent Blondel and Paul Van Dooren for their support while visiting UCL in Belgium, and Michael Mahoney and the SAMSI institute for the collaborations initiated there.

I also take this opportunity to express my gratitude to my professors at Hiram and Princeton for the numerous classes I attended. Going to class was so enjoyable, I loved it and I have learned so much from you. I am very grateful to Ingrid Daubechies, Robert Calderbank and Robert Strichartz for giving an undergraduate from a small liberal arts college in the middle of Ohio the opportunity to study at Princeton. I am also equally grateful to my undergraduate advisers Ugur Aker, Edward Smerek and Ellen Walker, for their warm guidance and for encouraging me to pursue a Ph.D., and my REU mentors Francine Blanchet-Sadri and Robert Strichartz for guiding my first steps in research.

I am very thankful to the graduate students in PACM and the Mathematics Department; they all made a difference from the day I visited Princeton deciding on a graduate school, until the very end of my stay here. Special thanks go to my roommates and friends at Princeton for making my stay there fun and enjoyable. I am also equally grateful to my friends from Hiram and Romania for their continued support and friendship. Thanks to you all, I felt like home no matter how far away from home I was.

Finally, I wish to thank my family members, in particular my parents, my grandparents and my brother. Their love, patience, support and sacrifice inspire me every single day.

To the memory of my grandparents.

Pentru maia si papu.

Contents

Chapter 1

Rigidity theory: basic definitions and results

Rigidity theory tries to answer if a given partial set of distances $d_{ij} = ||p_i - p_j||$ between n points in $\mathbb{R}^d$ uniquely determines the coordinates of the points $p_1, \ldots, p_n$ up to rigid transformations (translations, rotations, reflections). This section is a self contained but extremely selective and incomplete collection of basic definitions and results in rigidity theory from the literature (e.g., [28, 44, 55, 83, and references therein]). Readers who are unfamiliar with rigidity theory may wish to skip this section at first reading and use it as a glossary.

A *bar and joint framework* in $\mathbb{R}^d$ is an undirected graph $G = (V, E)$ ($|V| = n, |E| = m$) and a *configuration* p which assigns a point p_i in $\mathbb{R}^d$ to each vertex i of the graph. The edges of the graph correspond to distance constraints, that is, $(i, j) \in E$ iff there is a bar of length d_{ij} between p_i and p_j. Consider a motion of the configuration with $p_i(t)$ being the displacement vector of vertex i at time t. Any smooth motion that instantaneously preserves the distance d_{ij} must satisfy $\frac{d}{dt}\|p_i - p_j\|^2 = 0$ for all $(i, j) \in E$. Denoting the instantaneous velocity of the i-th point by $\dot{p}_i$, it follows that

$$(p_i - p_j)^T(\dot{p}_i - \dot{p}_j) = 0, \text{ for all } (i, j) \in E. \tag{1.1}$$

Given a framework $G(p)$ in $\mathbb{R}^d$, a solution $\dot{p} = [\dot{p}_1^T\ \dot{p}_2^T\ \cdots\ \dot{p}_n^T]^T$ with $\dot{p}_i$ in $\mathbb{R}^d$ to the system of linear equations (1.1) is called an *infinitesimal motion*. This linear system consisting of m equations in dn unknowns can be brought together as $R_G(p)\dot{p} = 0$, where $R_G(p)$ is the so called $m \times dn$ *rigidity matrix*.

Note that for every skew-symmetric $d \times d$ matrix A (with $A^T = -A$) and for every $b \in \mathbb{R}^d$ we have that $\dot{p}_i = Ap_i + b$ is an infinitesimal motion, where A accounts for some orthogonal transformation and b accounts for some translation. Such infinitesimal motions are called *trivial* because they are the derivative of rigid transformations. A framework $G(p)$ is *infinitesimally rigid* if all infinitesimal motions are trivial, and *infinitesimally flexible* otherwise. Observe that the trivial infinitesimal motions span a $d(d+1)/2$ dimensional

subspace of $\mathbb{R}^{dn}$, combining the d degrees of freedom of translations with the $d(d-1)/2$ degrees of freedom of orthogonal transformations. Therefore, the dimensionality of the null space of the rigidity matrix of a given framework determines if it is infinitesimally rigid or infinitesimally flexible: if $\dim \operatorname{null}(R_G(p)) > d(d+1)/2$ then $G(p)$ is infinitesimally flexible, otherwise $\dim \operatorname{null}(R_G(p)) = d(d+1)/2$ and $G(p)$ is infinitesimally rigid.

1.1 Local rigidity

A framework $G(p)$ is said to be *locally rigid* if there exists a neighborhood U of $G(p)$ such that $G(p)$ is the only framework in U with the same set of edge lengths, up to rigid transformations. In other words, there is no continuous deformation that preserves the edge lengths.

A configuration is *generic* if the coordinates do not satisfy any non-zero polynomial equation with integer coefficients (or equivalently algebraic coefficients). Generic configurations are not an open set of configurations in $\mathbb{R}^{dn}$, but they do form a dense set of full measure.

The following rigidity predictor was introduced by Gluck [42] and extensively used in Asimow and Roth [5]:

Theorem 1.1.1 (Gluck [42], Asimow and Roth [5])**.** *If a generic framework $G(p)$ of a graph G with $d+1$ or more vertices is locally rigid in $\mathbb{R}^d$, then it is infinitesimally rigid; otherwise* $\dim \operatorname{null}(R_G(p)) > d(d+1)/2$.

Since the dimension of the null space of the rigidity matrix is the same at every generic point, local rigidity in $\mathbb{R}^d$ is a generic property. That is, either all generic frameworks of the graph G are locally rigid, or none of them are. This is a condition for *generic local rigidity* in $\mathbb{R}^d$ which can be considered as a property of the graph G.

Hendrickson [55] observed that generic local rigidity can therefore be tested efficiently in any dimension using a randomized algorithm: simply randomize the displacement vectors $p_1, \ldots, p_n$ while ignoring the specific distance constraints that they have to satisfy, construct the rigidity matrix corresponding to the framework of the original graph with the randomized points and check its rank. With probability one, the rank of the rigidity matrix that corresponds to the unknown true displacement vectors equals the rank of the randomized rigidity matrix. A similar randomized algorithm for generic local rigidity was described in [44, Algorithm 3.2].

Since generic local rigidity is a combinatorial property of the graph, it is natural to search for a combinatorial characterization of such graphs. Such a combinatorial characterization exists for rigidity in the plane ($d = 2$). The total number of degrees of freedom for n points in the plane is $2n$. How many distance constraints are necessary to limit a framework to having only the trivial motions? Or equivalently, how many edges are necessary for a graph to be rigid? Each edge can remove a single degree of freedom. Rotations and translations will always be possible, so at least $2n-3$ edges are necessary for a graph

to be rigid. For example, a graph with $n = 3$ and $|E| = 2n - 3 = 3$ edges is the triangle which is rigid. Similarly, a graph with $n = 4$ and $|E| = 2n - 3 = 5$ is K_4 minus one edge which is also locally rigid. However, the graph on $n = 5$ vertices consisting of K_4 plus one dangling node has $2n - 3 = 7$ edges but it is not rigid. Such edge counting considerations had already been made by Maxwell [75] in the 19th century. Laman [72] was the first to prove the precise combinatorial characterization of rigid frameworks in the plane:

A framework is *minimally rigid*, if it is infinitesimally flexible once an edge is removed. A framework is *redundantly rigid*, or equivalently *edge-2-rigid* if it is infinitesimally rigid upon the removal of any single edge.

Theorem 1.1.2 (Laman [72])**.** *A graph with n vertices is generically minimally rigid in 2D if and only if it has $2n - 3$ edges and no subgraph of n' vertices has more than $2n' - 3$ edges. A graph is generically rigid if it contains a Laman graph with n vertices.*

In other words, Laman condition for minimally rigid graphs says that the graph needs to have at least $2n - 3$ "well-distributed" edges. Generic rigidity is a property of the graph connectivity, not the geometry. The *pebble game* algorithm of Hendrickson and Jacobs [63] applies Laman's theorem to determine generic local rigidity of a given graph in at most $O(n^2)$ steps. Laman graphs (i.e., generic minimally rigid graphs) are an instance of tight sparse graphs. A graph with n vertices and m edges is said to be (k, l)-*sparse* if every subset of $n' \leq n$ vertices spans at most $kn' - l$ edges. If, furthermore, $m = kn - l$, the graph is called *tight* (see, e.g, [73]). Thus the $(2, 3)$-sparse tight graphs are the Laman graphs. Unfortunately, an exact combinatorial characterization of locally generic rigid graphs is currently not available in higher dimensions ($d \geq 3$). The "edge-counting" condition is necessary but not sufficient.

1.2 Global rigidity

Local generic rigidity does not imply unique realization of the framework. For example, consider the $2D$-rigid graph with $n = 4$ vertices and $m = 5$ edges consisting of two triangles that can be folded with respect to their joint edge. A framework $G(p)$ is *globally rigid* in $\mathbb{R}^d$ if all frameworks $G(q)$ in $\mathbb{R}^d$ which are $G(p)$-equivalent (have all bars the same length as $G(p)$) are congruent to $G(p)$ (that is, they are related by a rigid transformation).

Hendrickson proved two key necessary conditions for global rigidity of a framework at a generic configuration:

Theorem 1.2.1 (Hendrickson [55])**.** *If a framework $G(p)$, other than a simplex, is globally rigid for a generic configuration p in $\mathbb{R}^d$ then:*

- *The graph G is vertex $(d + 1)$-connected;*

- *The framework $G(p)$ is edge-2-rigid (or, redundantly rigid), in the sense that removing any one edge leaves a graph which is infinitesimally rigid.*

A graph G is *generically globally rigid* in $\mathbb{R}^d$ if $G(p)$ is globally rigid at all generic configurations p [26, 27]. Only recently it was demonstrated that global rigidity is a generic property in this sense for graphs in each dimension [27, 44]. The conditions of Hendrickson as stated in Theorem 1.2.1 are necessary for generic global rigidity. They are also sufficient on the line, and in the plane [60]. However, by a result of Connelly [26], $K_{5,5}$ in 3-space is generically edge-2-rigid and 5-connected but is not generically globally rigid.

The critical technique used for proving global rigidity of frameworks uses stress matrices. A *stress* is an assignment of scalars w_{ij} to the edges such that for each $i \in V$

$$\sum_{j:\,(i,j)\in E} \omega_{ij}(p_i - p_j) = 0. \tag{1.2}$$

Alternatively, a stress is a vector w in the left null space of the rigidity matrix: $R_G(p)^T w = 0$. A stress vector can be rearranged into an $n \times n$ symmetric matrix Ω, known as the *stress matrix*, such that for $i \neq j$, the (i, j) entry of Ω is $\Omega_{ij} = -\omega_{ij}$, and the diagonal entries for (i, i) are $\Omega_{ii} = \sum_{j:\,j\neq i} \omega_{ij}$. Note that all row and column sums are now zero from which it follows that the all-ones vector $(1\ 1\ \cdots\ 1)^T$ is in the null space of Ω as well as each of the coordinate vectors of the configuration p. Therefore, for generic configurations the rank of the stress matrix is at most $n - (d + 1)$. The key connection for global rigidity is the following pair of results:

Theorem 1.2.2 (Connelly [27])**.** *If p is a generic configuration in $\mathbb{R}^d$, such that there is a stress, where the rank of the associated stress matrix Ω is $n-(d+1)$, then $G(p)$ is globally rigid in $\mathbb{R}^d$.*

Theorem 1.2.3 (Gortler, Healy, and Thurston [44])**.** *Suppose that p is a generic configuration in $\mathbb{R}^d$, such that $G(p)$ is globally rigid in $\mathbb{R}^d$. Then either $G(p)$ is a simplex or there is a stress where the rank of the associated stress matrix Ω is $n - (d + 1)$.*

Based on their theorem, Gortler, Healy and Thurston also provided a randomized polynomial algorithm for checking generic global rigidity of a graph [44, Algorithm 3.3]. If the graph is generically locally rigid then their algorithm picks a random stress vector of the left null space of the rigidity matrix, converts it into a stress matrix and computes the rank of the stress matrix which is compared with $n - (d + 1)$ to determine generic global rigidity.

Rigidity properties of random Erdős-Rényi $G(n, \beta)$ graphs, where each edge is chosen with probability β have been recently analyzed for the two dimensional case:

Theorem 1.2.4 (Jackson, Servatius, and Servatius [61])**.** *Let $G = G(n, \beta)$, where $\beta = (\log n + k \log\log n + w(n))/n$, and $\lim_{n\to\infty} w(n) = +\infty$.*

- *If $k = 2$ then G is asymptotically almost surely (a.a.s) generically locally rigid.*

- *If $k = 3$ then G is a.a.s. generically globally rigid.*

The bounds on β given in Theorem 1.2.4 are best possible in the sense that if $G = G(n, \beta)$ and $\beta = (\log n + k \log \log n + c)/n$ for any constant c, then G does not a.a.s. have minimum degree at least k. The emergence of large rigid components was also studied by Theran in [100].

Coning is a method for preserving global rigidity between dimensions. The process of coning a graph G adds a new vertex v, and adds edges from v to all original vertices in G, creating the cone graph $G * v$. A recent result of [28] states the following:

Theorem 1.2.5 (Connelly [28])**.** *A graph G is generically globally rigid in $\mathbb{R}^d$ if and only if the cone graph $G * v$ is generically globally rigid in $\mathbb{R}^{d+1}$.*

This result allows us to reduce the notion of global rigidity in $\mathbb{R}^3$ to global rigidity in $\mathbb{R}^2$, after removing the center node, and is crucial to our 2D-ASAP algorithm as detailed in Section 2.4.

1.3 Unique Localizability and Universal Rigidity

We start with some more notation needed for this section and the following chapters. We consider a sensor network in $\mathbb{R}^D$ with k anchors denoted by $\mathcal{A}$, and n sensors denoted by $\mathcal{S}$. An anchor is a node whose location $a_i \in \mathbb{R}^d$ is readily available, $i = 1, \ldots, k$, and a sensor is a node whose location x_j is to be determined, $j = 1, \ldots, n$.

We denote by d_{ij} the Euclidean distance between a pair of nodes, $(i, j) \in \mathcal{A} \cup \mathcal{S}$. In most applications, not all pairwise distance measurements are available, therefore we denote by $E(\mathcal{S}, \mathcal{S})$ and $E(\mathcal{S}, \mathcal{A})$ the set of edges denoting the measured sensor-sensor and sensor-anchor distances. We represent the available distance measurements in an undirected graph $G = (V, E)$ with vertex set $V = \mathcal{S} \cup \mathcal{A}$ of size $|V| = n + k$, and edge set of size $|E| = m$. An edge of the graph corresponds to a distance constraint, that is $(i, j) \in E$ iff the distance between nodes i and j is available and equals $d_{ij} = d_{ji}$, where $i, j \in \mathcal{A} \cup \mathcal{S}$. We denote the partial distance measurements matrix by $D = \{d_{ij} : (i, j) \in E(\mathcal{S}, \mathcal{S}) \cup E(\mathcal{S}, \mathcal{A})\}$. A solution x together with the anchor set a comprise a *localization* or *realization* $p = (x, a)$ of G. As defined at the beginning of this chapter, a *framework* in $\mathbb{R}^d$ is the ensemble (G, p), i.e., the graph G together with the realization p which assigns a point p_i in $\mathbb{R}^d$ to each vertex i of the graph.

Given a partial set of noiseless distances and the anchor set a, the graph realization problem can be formulated as the following system

$$\begin{aligned} \|x_i - x_j\|_2^2 &= d_{ij}^2 \quad \text{for } (i, j) \in E(\mathcal{S}, \mathcal{S}) \\ \|a_i - x_j\|_2^2 &= d_{ij}^2 \quad \text{for } (i, j) \in E(\mathcal{S}, \mathcal{A}) \\ x_i &\in \mathbb{R}^d \quad \text{for } i = 1, \ldots, n \end{aligned} \tag{1.3}$$

Unless the above system has enough constraints (i.e., the graph G has sufficiently many edges), then G is not globally rigid and there could be multiple solutions. However, if the graph G is known to be (generically) globally rigid in $\mathbb{R}^3$, and there are at least four anchors (i.e., $k \geq 4$), and G admits a generic realization[1], then (3.16) admits a unique solution. Due to recent results on the characterization of generic global rigidity, there now exists a randomized efficient algorithm that verifies if a given graph is generically globally rigid in $\mathbb{R}^d$ [45]. However, this efficient algorithm does not translate into an efficient method for actually computing a realization of G. Knowing that a graph is generically globally rigid in $\mathbb{R}^d$ still leaves the graph realization problem intractable, as shown in [7]. Motivated by this gap between deciding if a graph is generically globally rigid and computing its realization (if it exists), So and Ye introduced the following notion of unique d-localizability [96]. An instance (G, p) of the graph localization problem is said to be *uniquely d-localizable* if

1. the system (3.16) has a unique solution $\tilde{x} = (\tilde{x}_1; \ldots; \tilde{x}_n) \in \mathbb{R}^{nd}$, and
2. for any $l > d$, $\tilde{x} = ((\tilde{x}_1; \mathbf{0}), \ldots, (\tilde{x}_n; \mathbf{0})) \in \mathbb{R}^{nl}$ is the unique solution to the following system:

$$\begin{aligned} \|x_i - x_j\|_2^2 &= d_{ij}^2 \quad \text{for } (i,j) \in E(\mathcal{S}, \mathcal{S}) \\ \|(a_i; \mathbf{0}) - x_j\|_2^2 &= d_{ij}^2 \quad \text{for } (i,j) \in E(\mathcal{S}, \mathcal{A}) \\ x_i &\in \mathbb{R}^l \quad \text{for } i = 1, \ldots, n \end{aligned} \tag{1.4}$$

where $(v; \mathbf{0})$ denotes the concatenation of a vector v of size d with the all-zeros vector $\mathbf{0}$ of size $l - d$. The second condition states that the problem cannot have a non-trivial localization in some higher dimensional space $\mathbb{R}^l$ (i.e, a localization different from the one obtained by setting $x_j = (\tilde{x}_j; \mathbf{0})$ for $j = 1, \ldots, n$), where anchor points are trivially augmented to $(a_i; \mathbf{0})$, for $i = 1, \ldots, k$. A crucial observation should now be made: unlike global rigidity, which is a generic property of the graph G, the notion of *unique localizability* depends not only on the underlying graph G but also on the particular realization p, i.e., it depends on the framework (G, p). In Section 3.5.2 we introduce the notion of a weakly uniquely localizable graph, essential for the preprocessing step of the 3D-ASAP algorithm.

To establish a precise connection between the notion of unique localizability and the framework of rigidity theory, Zhu et al. introduce in [114] the rigidity theoretic notion of universal rigidity, whose definition is as follows. Let $d, n \geq 1$ be integers. Let $G = (V, E)$ be a graph on n nodes, and let $p = (p_1, \ldots, p_n) \in \mathbb{R}^{dn}$ be its realization in $\mathbb{R}^d$. We say that (G, p) is *universally rigid* in $\mathbb{R}^d$ if for any realization $q = (q_1, \ldots, q_n) \in \mathbb{R}^{hn}$ of G in $\mathbb{R}^h$ (where $h \geq 1$ is arbitrary), we have:

$$[||p_i - p_j||_2 = ||q_i - q_j||_2 \text{for} (i,j) \in E] \Longrightarrow [||p_i - p_j||_2 = ||q_i - q_j||_2 \text{for} 1 \leq i < j \leq n]$$

In other words, p is the unique (up to congruence) realization of G in any Euclidean space. Furthermore, if (G, p) is universally rigid in $\mathbb{R}^d$ for all generic realizations $p \in \mathbb{R}^{dn}$, then we say that the graph G is *generically universally rigid* in $\mathbb{R}^d$. Note that the notion of

[1] A realization is *generic* if the coordinates do not satisfy any non-zero polynomial equation with integer coefficients.

global rigidity defined in Section 1.2 is less restrictive than that of universal rigidity, since the former only requires that p is the unique (up to congruence) realization of G in $\mathbb{R}^d$.

The following theorem of Zhu et al. establishes the equivalence conditions between the notions of universal rigidity and unique d-localizability [114]. We denote by $\mathcal{D}$ the set of all available sensor-sensor $E(\mathcal{S},\mathcal{S})$ and sensor-anchor distances $E(\mathcal{S},\mathcal{A})$. Recall that a denotes the (known) location of all anchors, $\mathcal{S}$ the set of all sensors ($|\mathcal{S}| = n$), $\mathcal{A}$ the set of all anchors ($|\mathcal{A}| = k$), and we let $N = n + k$.

Theorem 1.3.1. *Suppose that $(G, \mathcal{D}, a, d)$ is a uniquely d-localizable instance of the graph realization problem, with $p = (x, a) \in \mathbb{R}^{d(n+k)}$ being its unique localization in $\mathbb{R}^l$ for all $l \geq d$. Then (G, p) is universally rigid in $\mathbb{R}^d$. Conversely, let $G = (V, E)$ be a graph with $N \geq d + 1$ nodes, and let $p \in \mathbb{R}^{dN}$ be a realization of G in $\mathbb{R}^d$. Suppose that (G, p) is universally rigid in $\mathbb{R}^d$. Then, whenever the number of anchors $k \geq d + 1$ and there exist $d + 1$ affinely independent vectors in the family $\{p_i\}_{i\in\mathcal{A}}$, the instance $(G, \mathcal{D}, p_{\mathcal{A}}, d)$ is uniquely d-localizable, where $p_{\mathcal{A}} = (p_i)_{i\in\mathcal{A}}$.*

The above result combined with Theorem 3.5.2 that appears in a later chapter, have important algorithmic consequences as they allow one to test efficiently whether a framework (G, p) is universally rigid in $\mathbb{R}^d$ using SDP. In particular, if (G, p) is universally rigid in $\mathbb{R}^d$, then the unique realization p can also be found efficiently by the same SDP formulation. This highly contrasts with the results for global rigidity, since it is NP-hard to check whether (G, p) is globally rigid in $\mathbb{R}^d$, and there is currently no efficient algorithm for finding the unique realization p of a globally rigid framework (G, p).

Chapter 2

Graph realization in $\mathbb{R}^2$ and localization of sensor networks

2.1 Introduction

Consider a graph $G = (V, E)$ consisting of a set of $|V| = n$ nodes and $|E| = m$ edges, together with a distance measurement associated with each edge. The graph realization problem is to assign to each vertex coordinates in $\mathbb{R}^d$ so that the Euclidean distance between any two neighboring nodes matches the distance associated to that edge. In other words, for any edge $(i, j) \in E$ we are given the distance $d_{ij} = d_{ji}$ between nodes i and j, and the goal is to find a d-dimensional embedding $p_1, p_2, \ldots, p_n \in \mathbb{R}^d$ such that $\|p_i - p_j\| = d_{ij},$ for all $(i, j) \in E$.

Due to its practical significance, the graph realization problem has attracted a lot of attention in recent years, across many communities. The problem and its variants come up naturally in a variety of settings such as wireless sensor networks [13, 104], structural biology [56], dimensionality reduction, Euclidean ball packing and multidimensional scaling (MDS) [30]. In such real world applications, the given distances d_{ij} between adjacent nodes are not accurate, $d_{ij} = \|p_i - p_j\| + \varepsilon_{ij}$ where ε_{ij} represents the added noise, and the goal is to find an embedding that realizes all known distances d_{ij} as best as possible.

When all $n(n-1)/2$ pairwise distances are known, a d-dimensional embedding of the complete graph can be computed using classical MDS. However, when many of the distance constraints are missing, the problem becomes significantly more challenging because the rank-d constraint on the solution is not convex. Applying a rigid transformation (composition of rotation, translation and possibly reflection) to a graph realization results in another graph realization, because rigid transformations preserve distances. Whenever an embedding exists, it is unique (up to rigid transformations) only if there are enough distance constraints, in which case the graph is said to be globally rigid or uniquely real-

izable (see, e.g., [55]). The graph realization problem is known to be difficult; Saxe has shown it is strongly NP-complete in one dimension, and strongly NP-hard for higher dimensions [86, 112]. Despite its difficulty, the graph realization problem has received a great deal of attention in the networking and distributed computation communities, and numerous heuristic algorithms exist that approximate its solution. In the context of sensor networks [65, 7, 6, 2], there are many algorithms that solve the graph realization problem, and they include methods such as global optimization [19], semidefinite programming (SDP) [17, 13, 14, 95, 96, 114] and local to global approaches [77, 89, 71, 91, 113].

In this chapter, we focus on the problem of localization of sensor networks in the plane ($\mathbb{R}^2$), although the approach is applicable to higher dimensions $d > 2$ as well. In the next chapter we consider the three dimensional version of this problem, and its application to a problem from structural biology. From here onwards we will refer to the two dimensional graph realization problem as the *sensor network localization* problem, or simply the *SNL* problem. Sensor networks are a collection of autonomous miniature devices distributed over a geographical area that cooperate to monitor various physical or environmental conditions. Each sensor is capable of limited computing power and wireless communication capabilities. While the initial development was motivated mainly by military applications, the current range of applications of sensor networks includes video surveillance, medical devices, monitoring of weather conditions and traffic control [104]. Since each sensor typically communicates with a small number of dynamic neighboring nodes, information flows through the network by means of ad-hoc routing algorithms. Traditional routing algorithms were based only on the connectivity of the measurement graph, but location-aware sensors lead to more efficient geographic routing. Such algorithms for geographically informed routing assume that nodes are located at precise coordinate locations or that the sensors are equipped with a GPS or similar localization systems. However, for certain applications, GPS devices may be too expensive, have high power consumptions, or may not be available as in indoors applications. Sensors that are aware of their location are often referred to as anchors, and anchor-based algorithms make use of their existence when computing the coordinates of the remaining sensors. Since in some applications the presence of anchors is not a realistic assumption, it is important to have efficient and robust to noise anchor-free algorithms, that can also incorporate the location of anchors if provided.

A popular model for the SNL problem is that of a disc graph model, where two sensors communicate with each other if and only if they are within sensing radius ρ of each other, i.e., $(i, j) \in E \iff d_{ij} \leq \rho$. The SNL problem is NP-hard also under the disc graph model [6]. Measuring inter-sensor distances is usually achieved by either Received Signal Strength Indicator (RSSI), where the strength of the signal decays with the increase of the distance, or the Time of Arrival (ToA) technique that uses the difference in the arrival times of the radio signal. Figure 3.1 shows an example of a measurement graph for a dataset of $n = 1090$ cities in the United States, with sensing radius $\rho = 0.032$, for which each node knows, on average, the distance to its $deg = 19$ closest neighbors.

Solutions to the SNL problem are often measured by three criteria: (1) sensitivity to noise in the distance measurements and sparse connectivity; (2) scalability to large networks; and (3) the property of being fully distributable. The third criterion means that local computations at each sensor should be based only on information available at that sensor and its neighbors. Most of the SNL algorithms, while allowing for distributable implemen-

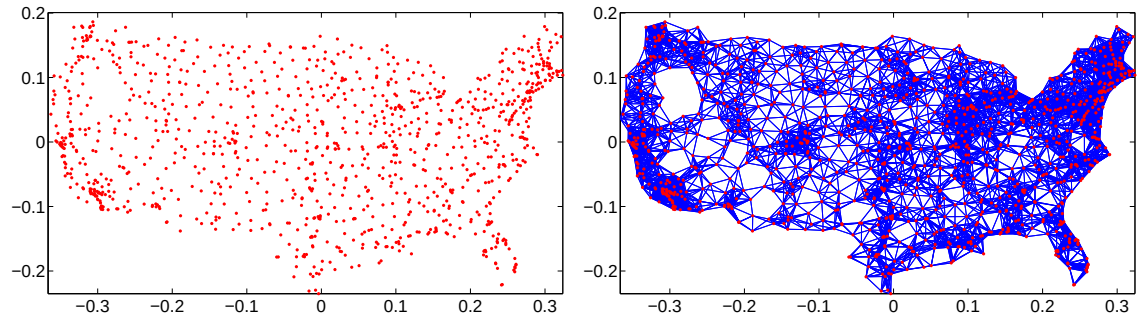


Figure 2.1: Original US map with $n = 1090$ cities (left) and the measurement graph with sensing radius $\rho = 0.032$ (right).

tations, are sensitive to noise and do not scale well as the size of the network increases.

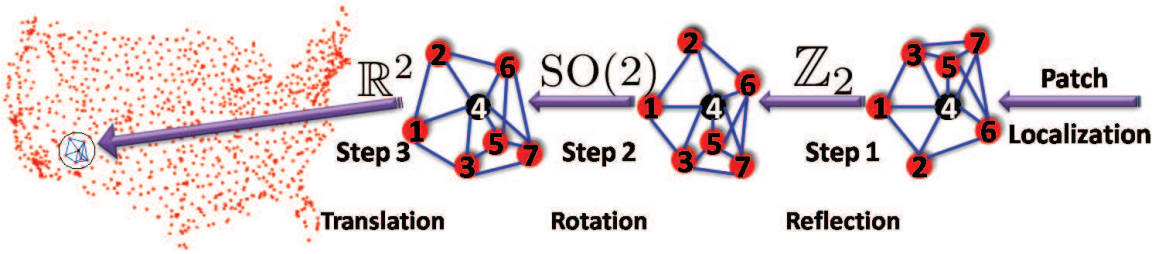


Figure 2.2: The 2D-ASAP recovery process for a patch in the US cities graph. The rightmost subgraph is the embedding of the patch in its own local frame, using a localization algorithm, such as stress minimization or SDP. To every patch, like the one shown here, there corresponds an elements of Euc(2) that we try to estimate. Using the pair alignments, in Step 1 we estimate the reflection from an eigenvector synchronization computation over $\mathbb{Z}_2$, in Step 2 we estimate the rotation angle by the same eigenvector synchronization method applied to SO(2), while in Step 3 we find the estimated coordinates by solving an overdetermined system of linear equations.

The algorithm we are about to describe belongs to the group of algorithms that integrate local distance information into a global structure determination. Our approach starts with identifying, for every sensor, globally rigid subgraphs of its 1-hop neighborhood that we call patches. Each patch is then separately localized in a coordinate system of its own using either the stress minimization approach of [46] or by SDP. In the noise-free case, the computed coordinates of the sensors in each patch must agree with their global positioning up to some unknown rigid motion, that is, up to translation, rotation and possibly reflection. To every patch there corresponds an element of the Euclidean group Euc(2) of rigid transformations in the plane, and the goal is to estimate the group elements that will properly align all the patches in a globally consistent way. By finding the optimal alignment

of all pairs of patches whose intersection is large enough, we obtain measurements for the ratios of the unknown group elements. Finding group elements from noisy measurements of their ratios is also known as the synchronization problem [66, 41]. For example, the synchronization of clocks in a distributed network from noisy measurements of their time offsets is a particular example of synchronization over $\mathbb{R}$. [92] introduced an eigenvector method for solving the synchronization problem over the group SO(2) of planar rotations. This algorithm will serve as the basic building block for our SNL algorithm. Namely, we reduce the SNL problem to three consecutive synchronization problems that overall solve the synchronization problem over Euc(2). Intuitively, we use the eigenvector method for the compact part of the group (reflections and rotations), and use the least-squares method for the non-compact part (translations). In the first step, we solve a synchronization problem over $\mathbb{Z}_2$ for the possible reflections of the patches using the eigenvector method. In the second step, we solve a synchronization problem over SO(2) for the rotations also using the eigenvector method. And, in the third step, we solve a synchronization problem over $\mathbb{R}^2$ for the translations by solving an overdetermined linear system of equations using the method of least squares. This solution yields the estimated coordinates of all the sensors up to a global rigid transformation. Figure 3.2 shows a schematic overview of our algorithm, which we call 2D-As-Synchronized-As-Possible (2D-ASAP).

From the computational point of view, all steps of the algorithm can be implemented in a distributed fashion and scale linearly in the size of the network, except for the eigenvector computation, which is nearly-linear[1]. We give a complexity analysis of the 2D-ASAP algorithm in Section 2.6, and demonstrate its scalability by localizing a network with 100,000 nodes. We conducted numerous numerical experiments that demonstrate the robustness of our algorithm to noise and to sparse connectivity of the measurement graph.

Our work in this chapter is organized as follows:

1. Section 2.2 contains a survey of existing methods for solving the SNL problem, although some of these methods also apply to the graph realization problem in dimensions $d > 2$.

2. Section 2.3 gives an overview of the 2D-ASAP algorithm we propose.

3. In Section 2.4, we explain the procedure for breaking up the initial large network into many smaller globally rigid subgraphs.

4. In Section 2.5, we describe several methods for aligning pairs of overlapping patches that have enough nodes in common.

5. Section 2.6 is a complexity analysis of each step of 2D-ASAP, and shows that the algorithm scales almost linearly in the size of the network.

6. In Section 2.7, we detail the results of numerical simulations in which we tested the performance of our algorithm in comparison to existing state-of-the-art algorithms.

7. Finally, Section 2.8 is a summary and a discussion of possible extensions of the algorithm and its usefulness in other applications.

[1]Every iteration of the power method is linear in the number of edges of the graph, but the number of iterations is greater than $O(1)$ as it depends on the spectral gap.

2.2 Related work

An approximate solution to the SNL problem is a two-dimensional embedding $p_1, \ldots, p_n \in \mathbb{R}^2$ that realizes all measured distances $d_{ij}, (i,j) \in E$ as best as possible. A popular approach to solving the SNL problem is based on SDP, and has attracted considerable attention in recent years [17, 12, 13, 14, 114]. One possible way of solving the SNL problem is to find the embedding $p_1, ..., p_n$ that minimizes the following error function

$$\min_{p_1,\ldots,p_n \in \mathbb{R}^2} \sum_{(i,j)\in E} \left(\|p_i - p_j\|^2 - d_{ij}^2 \right)^2 . \tag{2.1}$$

While the above objective function is not convex over the constraint set, it can be relaxed into an SDP [13]. Although SDP can be generally solved (up to a given accuracy) in polynomial time, it was pointed out in [14] that the objective function (2.1) leads to a rather expensive SDP, because it involves fourth order polynomials of the coordinates. Additionally, this approach is rather sensitive to noise, because large errors are amplified by the objective function in (2.1), compared to the objective functions in (2.2) and (2.3) that are discussed below.

Instead of using the objective function in (2.1), [14] consider the SDP relaxation of the following penalty function

$$\min_{p_1,\ldots,p_n \in \mathbb{R}^2} \sum_{(i,j)\in E} \left| \| p_i - p_j \|^2 - d_{ij}^2 \right| . \tag{2.2}$$

In fact, they also allow for possible non-equal weighting of the summands in (2.2) and for possible anchor points. The SDP relaxation of (2.2) is faster to solve than the relaxation of (2.1) and it is usually more robust to noise. Constraining the solution to be in $\mathbb{R}^2$ is non-convex, and its relaxation by the SDP often leads to solutions that belong to a higher dimensional Euclidean space that are projected to the plane. This projection often results in large errors for the estimation of the coordinates. A regularization term for the objective function of the SDP was suggested in [14] to assist it in finding solutions of lower dimensionality and preventing nodes from crowding together towards the center of the configuration. To improve the overall localization result, the SDP solution is used as a starting point for a gradient-descent method. The gradient descent method generally fails to compute the global optimal solution of the non-convex problem, unless a good initialization is provided.

In our simulations, we find that in the absence of anchor points, the SDP approach works well when the sensing radius is large enough and for relatively low levels of noise. However, as the size of the network grows, solving one large SDP problem can become too expensive. The numerical simulations in Figures 2.20 and 2.21 show that the SDP approach is sensitive to high levels of noise, to sparse connectivity of the graph, and to the number and the locations of the anchors.

Another approach to solving the SNL problem is by minimizing the following stress function

$$Stress(p_1, \ldots p_n) = \sum_{(i,j)\in E} \left(\| p_i - p_j \| - d_{ij} \right)^2 \tag{2.3}$$

over all possible configurations $p_1, \dots p_n \in \mathbb{R}^2$. One of the more recent iterative algorithms that was observed to perform well in practice compared to other traditional optimization methods is a variant of the gradient descent approach called the stress majorization algorithm, also known as SMACOF [19], originally introduced by [34]. The main drawback of this approach is that the stress function in (2.3) is not convex and the search for the global minimum is prone to getting stuck at local minima. This often makes the initial guess for gradient descent based algorithms important for obtaining satisfactory results. What usually happens at a local but not global minimum is a phenomenon known as foldovers, where large pieces of the graph realization fold on top of others. Long range distance measurements help to prevent foldovers in the recovered solution, but such measurements are rarely available in applications, and are completely absent in the disc graph model.

In [77] the authors proposed an incremental algorithm which first tries to localize small subsets of the network. Each such local subset consists of four sensors forming a rigid graph, i.e., the complete graph K_4 on four vertices where all six pairwise distances are known (a quad). The procedure for embedding such quads is called *trilateration*, and the method is incremental in the sense that once a quad has been localized, another one is found which has a common triangle with the first one, and the alignment is performed by applying the best possible rigid transformation between the two. Using breadth-first-search, all the existing quads of the graph are localized, with the intermediate embedding being improved at each step by running stress minimization. One drawback of this method, besides being incremental, is that it localizes only sensors contained in trilateralizable components of the network, but not all globally rigid graphs are trilateralizable.

The authors of [89] describe a similar algorithm that first localizes small subsets of nodes, and then stitches them together sequentially. The initial embedding of a patch is obtained by first computing all pairwise shortest paths in the weighted connectivity graph of the patch in order to estimate all missing distances, followed by MDS to obtain an initial estimate which is then improved by running the stress minimization algorithm. The patches are glued together incrementally in a greedy fashion by finding the best affine transformation between a new patch and the current global layout. Finally, the complete network obtained in this manner is improved using stress minimization. The main drawback of such incremental algorithms is their sensitivity to noise due to accumulation of the errors.

In an attempt to depart from the noise sensitive incremental methods, [71] proposed PATCHWORK, an algorithm that avoids stitching patches together in a greedy manner. Their method embeds small local patches, that are later glued together using a distributed global optimization process. Patches are mapped to a global coordinate system using affine transformations, and the patch overlaps yield a linear least-squares problem enforcing that the transformations agree well on the common vertices.

Maximum variance unfolding (MVU) is a non-linear dimensionality reduction algorithm [109]. It produces a low-dimensional representation of the data by maximizing the variance of its embedding while preserving the original local distance constraints. MVU builds on the SDP approach and addresses the issue of the possibly high dimensional solution to the SDP problem. While rank constraints are non convex and cannot be directly imposed, it has been observed that low dimensional solutions emerge naturally maximizing the variance of the embedding (also known as the maximum trace heuristic). Their main observation is that the *x* and *y* coordinate vectors of the sensors are often well approximated

by just the first few (e.g., 10) low-oscillatory eigenvectors of the graph Laplacian. This observation allows to replace the original and possibly large scale SDP by a much smaller SDP which leads to a significant reduction in running time.

The Locally Rigid Embedding (LRE) algorithm [91] is reminiscent of the Locally Linear Embedding (LLE) [84] technique used in machine learning for dimensionality reduction. LRE tries to preserve, in a global coordinate system, the local affine relationships present within patches. Each sensor contributes with a linear equation relating its location to those of its neighboring nodes, thus altogether setting up a global linear system. LRE builds up a specially designed sparse matrix whose eigenvectors give an embedding of all sensors, from which a global affine transformation must be removed. The LRE algorithm is able to recover the global coordinates from local noisy measurements, under the assumption that every node together with its neighbors form a rigid subgraph that can be embedded uniquely, up to a rigid transformation.

The authors of [113] recently proposed an algorithm along the lines of PATCHWORK and LRE, called As-Rigid-As-Possible (ARAP). Their algorithm starts off by localizing small patches in a similar manner, but instead of finding a global embedding via affine mappings they use rigid mappings. Again, the patch overlaps impose constraints on the mappings, however the usage of rigid mappings has the advantage of better preserving the local relationships between patches. This comes at the price of resulting in a nonlinear optimization problem, which is solved efficiently using a two-phase alternating least squares method. The algorithm requires an initial guess for the nonlinear optimization, which is obtained by As-Affine-As-Possible (AAAP), an improved version of the LRE and PATCHWORK algorithms. The reported experimental results, confirmed also by our own experiments, show that ARAP is more robust to sparse connectivity and noise in the measurement graph compared to all other algorithms surveyed above.

DILAND, recently introduced in [70], is a distributed algorithm for localization with noisy distance measurements. Under appropriate conditions on the connectivity and triangulation of the network, DILAND was shown to converge almost surely to the true solution.

2.3 The 2D-ASAP Algorithm

The gist of our algorithm is to break up the large graph into many smaller overlapping subgraphs, that we call patches, and "stitch" them together consistently in a global coordinate system with the purpose of localizing the entire measurement graph. To avoid foldovers in the final solution, each such patch needs to be globally rigid and the entire measurement graphs needs to be globally rigid as well[2].

The patches are determined in the following way. For every node i we denote by $V(i) = \{j : (i,j) \in E\} \cup \{i\}$ the set of its neighbors together with the node itself, and by $G(i) = (V(i), E(i))$ its subgraph of 1-hop neighbors. If $G(i)$ is globally rigid, then we embed it in

[2]We remark that in the disc graph model, the non-edges also provide distance information since $(i,j) \notin E$ implies $d_{ij} > \rho$. This information sometimes allows to uniquely localize networks that are not globally rigid to begin with. This information, however, is not used in the standard formulation of the 2D-ASAP algorithm, except for extremely sparse networks, as discussed later in Section 2.5.

$\mathbb{R}^2$. If $G(i)$ is not globally rigid, we break it into maximally globally rigid subgraphs that we call patches, and embed each patch in $\mathbb{R}^2$. The embedding of every patch in $\mathbb{R}^2$ is given in its own local frame. The exact way we break up the 1-hop neighborhood subgraphs into smaller maximally globally rigid subgraphs is detailed in Section 2.4. We denote by N the number of patches obtained in the above decomposition of the measurement graph, and note that it may be different from n, the number of nodes in G, since the neighborhood graph of a node may contribute several patches or none.

For the embedding of local patches we usually use the Stress majorization algorithm as described in [46]. Once each patch is embedded in its own coordinate system, one must find the reflections, rotations and translations that will stitch all patches together in a consistent manner, a process to which we refer as *synchronization*.

To every patch P_i there corresponds an element $e_i \in$ Euc(2), where Euc(2) is the Euclidean group of rigid motions in the plane. The rigid motion e_i moves patch P_i to its correct position with respect to the global coordinate system. Our goal is to estimate the rigid motions $e_1, \ldots, e_N$ (up to a global rigid motion) that will properly align all the patches in a globally consistent way. To achieve this goal, we first estimate the alignment between any pair of patches P_i and P_j that have enough nodes in common (alignment methods are discussed in Section 2.5). The alignment of patches P_i and P_j provides a (perhaps noisy) measurement for the ratio $e_i e_j^{-1}$ in Euc(2). We solve the resulting synchronization problem in a globally consistent manner, such that information from local alignments propagates to pairs of non-overlapping patches. This is done by replacing the synchronization problem over Euc(2) by three different consecutive synchronization problems. In the first synchronization problem, we find the reflections of all the patches using the eigenvector synchronization algorithm over the group $\mathbb{Z}_2$. Once the reflections are estimated, we use the eigenvector synchronization method over SO(2) to estimate the rotations of all patches. Once both reflections and rotations are estimated, we estimate the translations by solving an overdetermined linear system. In other words, we integrate all the available local information into a global coordinate system over several steps by using the eigenvector synchronization algorithm and least squares over the isometries of the Euclidean plane. The main advantage of the eigenvector method is that it can recover the reflections and rotations even when many of the alignments are incorrect. The algorithm is summarized in Table 2.1.

2.3.1 Step 1: Synchronization over $\mathbb{Z}_2$ to estimate reflections

As mentioned earlier, for every patch P_i that was already embedded in its local frame, we need to estimate whether or not it needs to be reflected with respect to the global coordinate system. We denote the reflection of patch P_i by $z_i \in \{-1, 1\}$. These are defined up to a global reflection (global sign). The alignment of every pair of patches P_i and P_j whose intersection is sufficiently large, provides a measurement z_{ij} for the ratio $z_i z_j^{-1}$. However, some ratio measurements can be corrupted because of errors in the embedding of the patches due to noise in the measured distances. We denote by $G^P = (V^P, E^P)$ the patch graph whose vertices V^P are the patches $P_1, \ldots, P_N$, and two patches P_i and P_j are

INPUT	$G=(V,E),\ \|V\|=n,\ \|E\|=m,\ d_{ij}$ for $(i,j)\in E$
Pre-processing Step	1. Break the measurement graph G into N globally rigid patches $P_1,\ldots,P_N$. 2. Embed each patch P_i separately using the embedding method of choice (e.g., stress majorization or SDP).
Step 1 Estimating Reflections	1. Align all pairs of patches (P_i,P_j) that have enough nodes in common. 2. Estimate their relative reflection $z_{ij}\in\{-1,+1\}$. 3. Build a sparse $N\times N$ symmetric matrix $Z=(z_{ij})$ as defined in (4.3). 4. Define $\mathcal{Z}=D^{-1}Z$, where D is a diagonal matrix with $D_{ii}=deg(i)$. 5. Compute the top eigenvector $v_1^{\mathcal{Z}}$ of $\mathcal{Z}$ which satisfies $\mathcal{Z}v_1^{\mathcal{Z}}=\lambda_1^{\mathcal{Z}}v_1^{\mathcal{Z}}$. 6. Estimate the global reflection of patch P_i by $\hat{z}_i=\mathrm{sign}(v_1^{\mathcal{Z}}(i))=\frac{v_1^{\mathcal{Z}}(i)}{\|v_1^{\mathcal{Z}}(i)\|}$. 7. Replace the embedding patch P_i with its mirrored image whenever $\hat{z}_i=-1$.
Step 2 Estimating Rotations	1. Align all pairs of patches (P_i,P_j) that have enough nodes in common. 2. Estimate their relative rotation angle $\theta_{ij}\in[0,2\pi)$ and set $r_{ij}=e^{\imath\theta_{ij}}$. 3. Build a sparse $N\times N$ Hermitian matrix $R=(r_{ij})$ as defined in (4.1). 4. Define $\mathcal{R}=D^{-1}R$. 5. Compute the top eigenvector $v_1^{\mathcal{R}}$ of $\mathcal{R}$ corresponding to $\mathcal{R}v_1^{\mathcal{R}}=\lambda_1^{\mathcal{R}}v_1^{\mathcal{R}}$. 6. Estimate the global rotation angle $\hat{\theta}_i$ of patch P_i using $e^{\imath\hat{\theta}_i}=\frac{v_1^{\mathcal{R}}(i)}{\|v_1^{\mathcal{R}}(i)\|}$. 7. Rotate the embedding of patch P_i by the angle θ_i.
Step 3 Estimating Translations	1. Build the $m\times n$ overdetermined system of linear equations given in (3.13). 2. Include the anchors information (if available) into the linear system. 3. Compute the least squares solution for the x-axis and y-axis coordinates.
OUTPUT	Estimated coordinates $\hat{p}_1,\ldots,\hat{p}_n$

Table 2.1: Overview of the 2D-ASAP Algorithm

adjacent, $(P_i,P_j)\in E^P$, iff they have enough[3] vertices in common to be aligned such that the ratio $z_iz_j^{-1}$ can be estimated.

The first step of the 2D-ASAP algorithm is to estimate the appropriate reflections of all patches. To that end, we use the eigenvector synchronization method as it was shown to perform well even in the presence of a large number of errors. The eigenvector method starts off by building the following $N\times N$ sparse symmetric matrix $Z=(z_{ij})$

$$z_{ij}=\begin{cases}1 & \text{aligning } P_i \text{ with } P_j \text{ did not require reflection}\\ -1 & \text{aligning } P_i \text{ with } P_j \text{ required reflection of one of them}\\ 0 & (i,j)\notin E^P \text{ } (P_i \text{ and } P_j \text{ cannot be aligned})\end{cases} \tag{2.4}$$

We explain in more detail in Section 2.5 the procedures by which we align pairs of patches, if such an alignment is at all possible.

Prior to computing the top eigenvector of the matrix Z, as done in [92], we choose to normalize it as follows. Let D be an $N\times N$ diagonal matrix,[4] whose entries are given by $D_{ii}=\sum_{j=1}^{N}|z_{ij}|$. In other words,

$$D_{ii}=deg(i), \tag{2.5}$$

[3] E.g., three common vertices, although the precise definition of "enough" will be given later

[4] The diagonal matrix D should not be confused with the partial distance matrix.

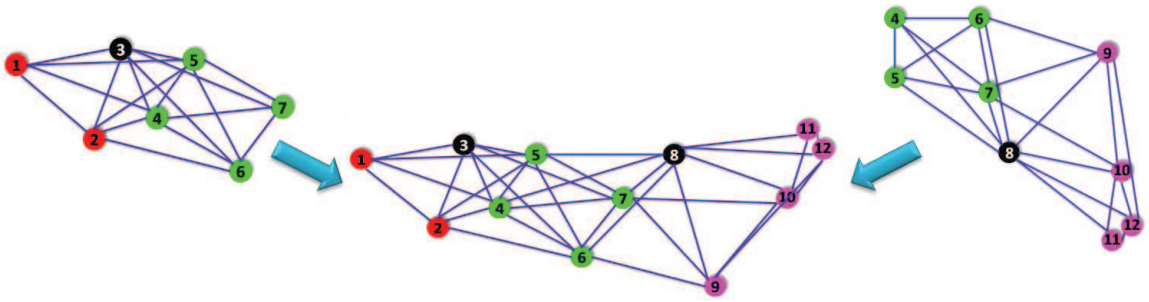


Figure 2.3: Optimal alignment of two patches that overlap in four nodes. The alignment provides a measurement for the ratio of the two group elements in Euc(2). In this example we see that a reflection was required to properly align the patches.

where $deg(i)$ is the node degree of patch P_i in G^P, that is, the number of other patches that can be aligned with it. We define the matrix $\mathcal{Z}$ as

$$\mathcal{Z} = D^{-1}Z, \tag{2.6}$$

and note that, although not necessarily symmetric, it is similar to the symmetric matrix $D^{-1/2}ZD^{-1/2}$ through

$$\mathcal{Z} = D^{-1/2}(D^{-1/2}ZD^{-1/2})D^{1/2}.$$

Therefore, the matrix $\mathcal{Z}$ has N real eigenvalues $\lambda_1^{\mathcal{Z}} > \lambda_2^{\mathcal{Z}} \geq \cdots \geq \lambda_N^{\mathcal{Z}}$ and N orthonormal eigenvectors $v_1^{\mathcal{Z}}, ..., v_N^{\mathcal{Z}}$, satisfying $\mathcal{Z}v_i^{\mathcal{Z}} = \lambda_i^{\mathcal{Z}} v_i^{\mathcal{Z}}$. In the eigenvector method, we compute the top eigenvector $v_1^{\mathcal{Z}} \in \mathbb{R}^N$ of $\mathcal{Z}$, which satisfies

$$\mathcal{Z}v_1^{\mathcal{Z}} = \lambda_1^{\mathcal{Z}} v_1^{\mathcal{Z}}, \tag{2.7}$$

and use it to obtain estimators $\hat{z}_1, \ldots, \hat{z}_N$ for the reflections of the patches, in the following way:

$$\hat{z}_i = \text{sign}(v_1^{\mathcal{Z}}(i)) = \frac{v_1^{\mathcal{Z}}(i)}{|v_1^{\mathcal{Z}}(i)|}, \quad i = 1, 2, \ldots, N. \tag{2.8}$$

The top eigenvector recovers the reflection of all patches up to a global sign, since if $v_1^{\mathcal{Z}}$ is the top eigenvector of $\mathcal{Z}$ then so is $-v_1^{\mathcal{Z}}$. After estimating the reflection of all patches, we replace the embedding of patch P_i by its mirrored image whenever $\hat{z}_i = -1$.

Both the success of the eigenvector method in estimating the correct reflections and the importance of the normalization (2.6) are demonstrated in Figures 2.4 and 2.5 that correspond to the US cities graph with sensing radius $\rho = 0.032$ and average degree $deg = 19$. The percentages of patches for which the top eigenvector $v_1^{\mathcal{Z}}$ of $\mathcal{Z}$ failed to estimate the reflection correctly are only $\tau = 0\%$ and $\tau = 0.1\%$ corresponding to distance measurement errors of $\eta = 0\%$ and $\eta = 20\%$, respectively (η is defined in (2.28)). That is, even when the measured distances are off by as much as 20% from their correct values, only 0.1% of the patches were assigned the wrong reflection. Without the normalization, however, extracting the signs of the top eigenvector v_1^Z of Z leads to significantly larger error rates

($\tau = 24.6\%$ and $\tau = 40.2\%$). We defer to Section 4.2 of the chapter on synchronization, the theoretical explanation for this behavior, but the numerical evidence in Figures 2.4 and 2.5 already provides some intuition. For example, Figure 2.4 shows that most entries of v_1^Z are close to zero (even in the noise free case) and therefore sign confusion is probable, while only a few entries have large magnitude. On the other hand, for the normalized matrix $\mathcal{Z}$, its top eigenvector in the noise-free case has only two possible values (1 and -1), and even in the noisy case, where entries have different magnitudes, their signs rarely get confused. Another difference between the two cases can be realized from Figure 2.5 that shows the eigenvalue histograms and bar plots for the matrices Z and $I - \mathcal{Z}$. While the eigenvalues of Z are both negative and positive, all eigenvalues of $I - \mathcal{Z}$ seem to be non-negative, as the latter matrix is related to the Laplacian of the patch graph, a fact that will be later explored in Section 4.2.

2.3.2 Step 2: Synchronization over SO(2) to estimate rotations

After estimating the reflections, we turn in Step 2 to estimate the rotations of all patches, that will properly align them with respect to the global coordinate system, up to translations and a global rotation. To each patch we associate an element $r_i \in$ SO(2), $i = 1, \ldots, N$ that we represent as a point on the unit circle in the complex plane $r_i = e^{\imath\theta_i} = \cos\theta_i + \imath \sin\theta_i$. We repeat the alignment process from Step 1 to estimate the angle θ_{ij} between two overlapping patches, i.e., the angle by which one needs to rotate patch P_i to align it with patch P_j. When the aligned patches contain corrupted distance measurements, θ_{ij} is a noisy measurement of their offset $\theta_i - \theta_j \mod 2\pi$. Following a similar approach to Step 1, we build the $N \times N$ sparse symmetric matrix $R = (r_{ij})$ whose elements are either 0 or points on the unit circle in the complex plane:

$$r_{ij} = \begin{cases} e^{\imath\theta_{ij}} & \text{if } (i,j) \in E^P \\ 0 & \text{if } (i,j) \notin E^P \end{cases}. \tag{2.9}$$

Since $\theta_{ij} = -\theta_{ji} \mod 2\pi$, it follows that R is a Hermitian matrix, that is, $R_{ij} = \bar{R}_{ji}$, where for any complex number $w = a + \imath b$ we denote by $\bar{w} = a - \imath b$ its complex conjugate. Note that the patch graph G^P may change (have extra edges) from Step 1 to Step 2, because the registration method needs at least three nodes in the intersection of patches P_i and P_j in order to compute the relative reflection, but only two such points to compute the rotation angle. However, for simplicity, we assume that the patch graph G^P is the same for both Steps 1 and 2.

As in Step 1, we choose to normalize R using the diagonal matrix D, whose diagonal elements are also given by $D_{ii} = \sum_{j=1}^{N} |r_{ij}|$. We define the matrix

$$\mathcal{R} = D^{-1}R, \tag{2.10}$$

which is similar to the Hermitian matrix $D^{-1/2}RD^{-1/2}$ through

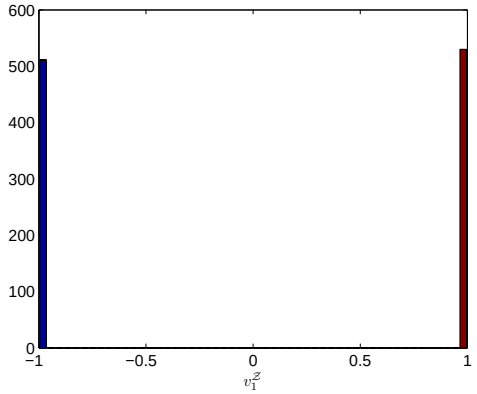
$$\mathcal{R} = D^{-1/2}(D^{-1/2}RD^{-1/2})D^{1/2}.$$

Therefore, $\mathcal{R}$ has N real eigenvalues $\lambda_1^{\mathcal{R}} > \lambda_2^{\mathcal{R}} \geq \cdots \geq \lambda_N^{\mathcal{R}}$ with corresponding N orthogonal (complex valued) eigenvectors $v_1^{\mathcal{R}}, \ldots, v_N^{\mathcal{R}}$, satisfying $\mathcal{R}v_i^{\mathcal{R}} = \lambda_i^R v_i^{\mathcal{R}}$. We define the estimated rotation angles $\hat{\theta}_1, ..., \hat{\theta}_N$ and their corresponding elements in SO(2), $\hat{r}_1, ..., \hat{r}_N$ using the top eigenvector $v_1^{\mathcal{R}}$ as

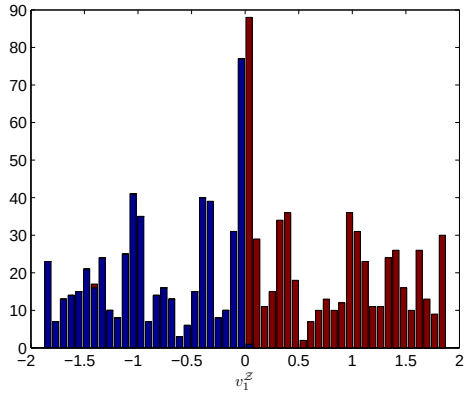
$$\hat{r}_i = e^{i\hat{\theta}_i} = \frac{v_1^R(i)}{|v_1^R(i)|}, \quad i = 1, 2, \ldots, N. \tag{2.11}$$

The estimation of the rotation angles is up to an additive phase since $e^{i\alpha}v_1^R$ is also an eigenvector of $\mathcal{R}$ for any $\alpha \in \mathbb{R}$.

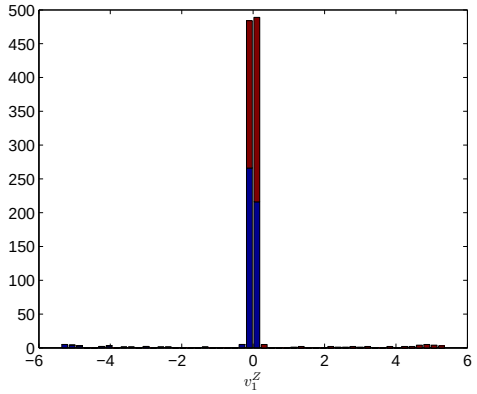
Note that the only difference between Step 2 and the angular synchronization algorithm in [92] is the normalization of the matrix prior to the computation of the top eigenvector. The usefulness of the normalization and the success of (2.11) in estimating the rotation angles are demonstrated in Figures 2.6, 2.7 and 2.8.



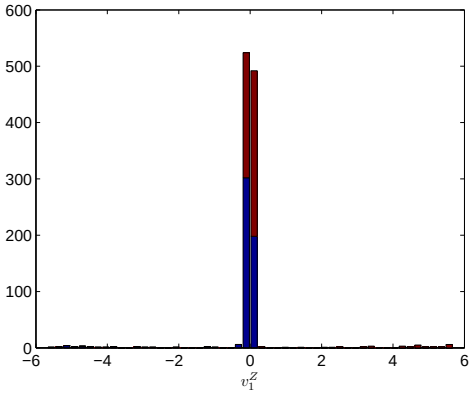
(a) $\mathcal{Z},\ \eta = 0\%,\ \tau = 0\%$



(b) $\mathcal{Z},\ \eta = 20\%,\ \tau = 0.1\%$

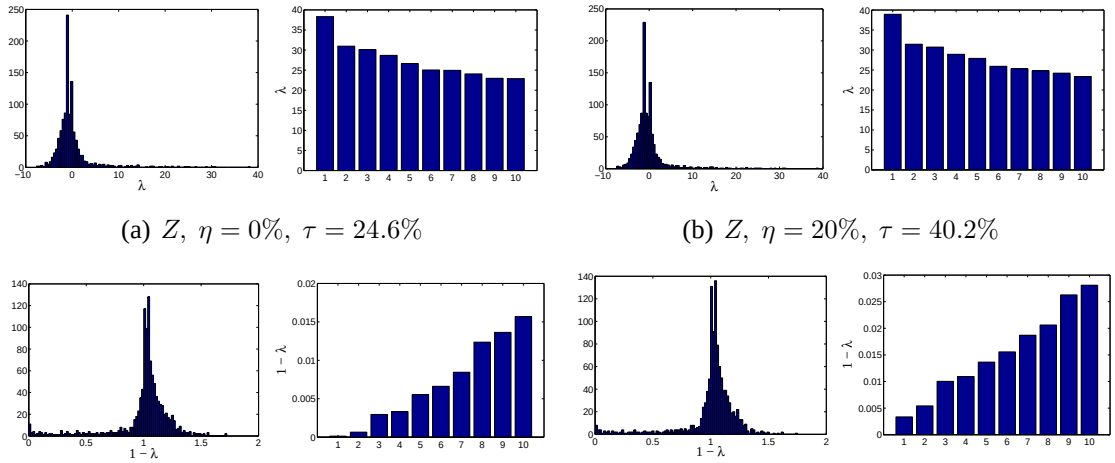


(c) $Z,\ \eta = 0\%,\ \tau = 24.6\%$



(d) $Z,\ \eta = 20\%,\ \tau = 40.2\%$

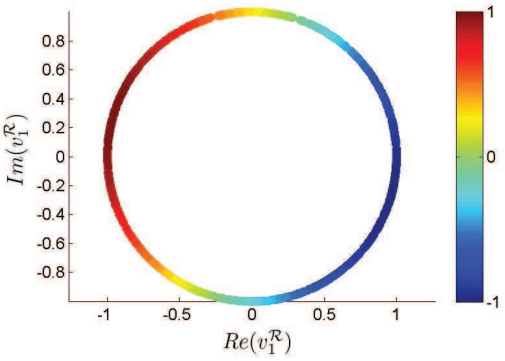
Figure 2.4: Histogram of the entries of the top eigenvectors $v_1^{\mathcal{Z}}$ and v_1^{Z} (scaled such that $\|v_1^{\mathcal{Z}}\|^2 = \|v_1^{Z}\|^2 = N$) for various noise levels for the US cities graph with sensing radius $\rho = 0.032$. Patches P_i for which $z_1 = -1$ are colored blue, while patches for which $z_i = 1$ are marked in red. Note that the top eigenvector $v_1^{\mathcal{Z}}$ is a good classifier between red and blue, while v_1^{Z} results in many misclassifications.



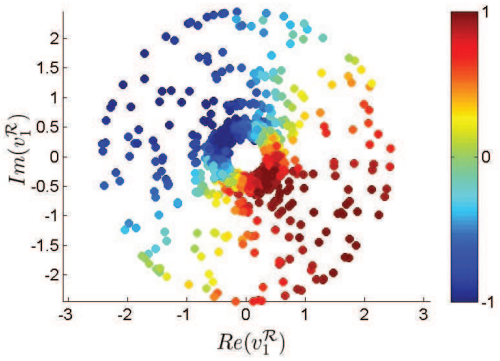
(a) $Z,\ \eta = 0\%,\ \tau = 24.6\%$ (b) $Z,\ \eta = 20\%,\ \tau = 40.2\%$

(c) $\mathcal{Z},\ \eta = 0\%,\ \tau = 0\%$ (d) $\mathcal{Z},\ \eta = 20\%,\ \tau = 0\%$

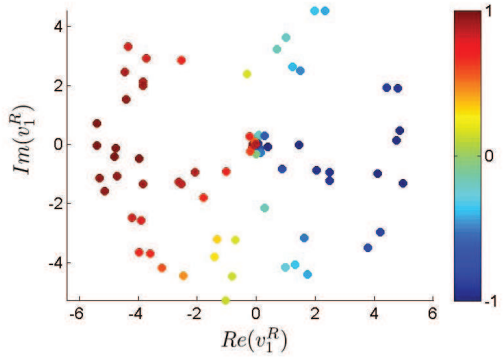
Figure 2.5: Histogram of all eigenvalues and bar-plot of the top 10 eigenvalues of Z and $\mathcal{Z}$ for the US cities graph with $\rho = 0.032$ ($deg = 19$) and various noise levels η. The resulting error rate τ is the percentage of patches whose reflection was incorrectly estimated. To ease the visualization of the eigenvalues of $\mathcal{Z}$, we choose to plot $1 - \lambda^{\mathcal{Z}}$ because the top eigenvalues of $\mathcal{Z}$ tend to pile up near 1, so it is difficult to differentiate between them by looking at the bar plot of $\lambda^{\mathcal{Z}}$.



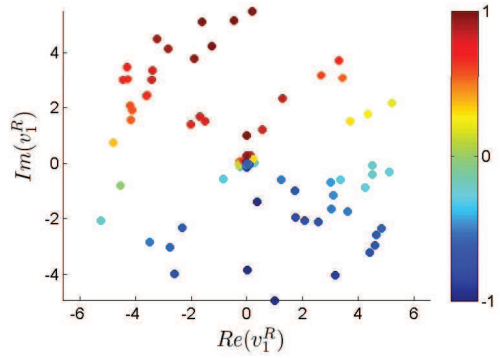
(a) $\mathcal{R},\ \eta = 0\%$



(b) $\mathcal{R},\ \eta = 20\%$

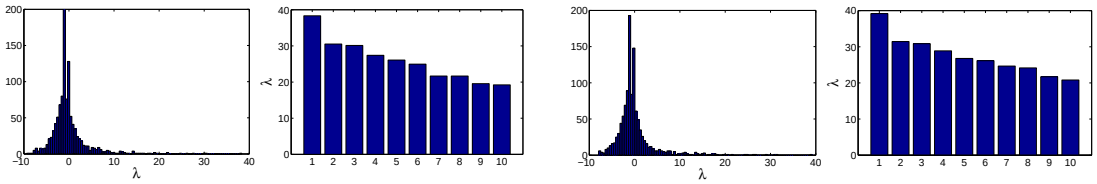


(c) $R,\ \eta = 0\%$



(d) $R,\ \eta = 20\%$

Figure 2.6: Scatter plots in the complex plane of the entries of the top eigenvectors $v_1^{\mathcal{R}}$ and v_1^R for the US cities graph with $\rho = 0.032$ ($deg = 19$) and various noise levels η. The color of the points correspond to $\cos\theta_i = \mathrm{Re}(r_i)$.



(a) $R,\ \eta = 0\%$ (b) $R,\ \eta = 20\%$

(c) $\mathcal{R},\ \eta = 0\%$ (d) $\mathcal{R},\ \eta = 20\%$

Figure 2.7: Histogram of all eigenvalues and bar-plot of the top 10 eigenvalues of R and $\mathcal{R}$ for the US cities graph with $\rho = 0.032$ and various noise levels η. Note that, as we did with $\mathcal{Z}$, also for $\mathcal{R}$ we plot the histogram and bar plots of $1 - \lambda^{\mathcal{R}}$.

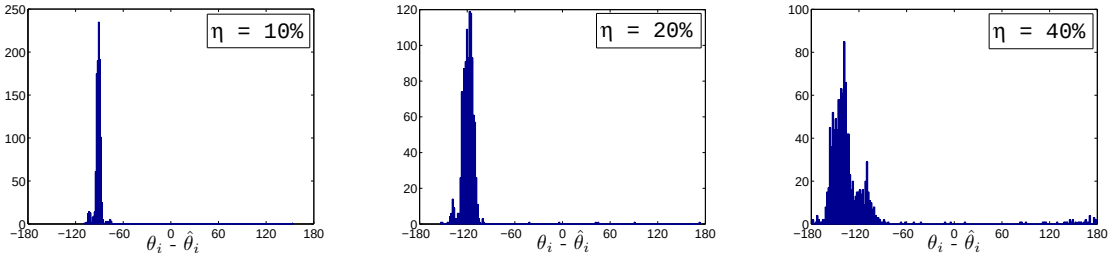


Figure 2.8: Histogram of the angle estimation error $\theta_i - \hat{\theta}_i$ (in degrees) for the US cities graph with $\rho = 0.032$ and various noise levels η. Note that angles are estimated up to an arbitrary phase and we have not mean shifted the histograms.

2.3.3 Step 3: Synchronization over $\mathbb{R}^d$ to estimate translations

The final Step of the 2D-ASAP algorithm is computing the global translations of all patches and recovering the true coordinates. For each patch P_k, we denote by $G_k = (V_k, E_k)$ [5] the graph associated to patch P_k, where V_k is the set of nodes in P_k, and E_k is the set of edges induced by V_k in the measurement graph $G = (V, E)$. We denote by $p_i^{(k)} = (x_i^{(k)}, y_i^{(k)})^T$

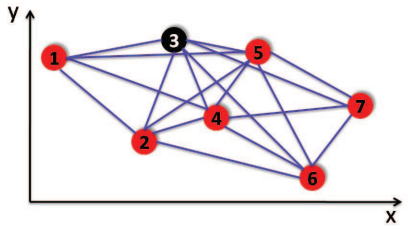


Figure 2.9: An embedding of a patch P_k in its local coordinate system (frame) after it was appropriately reflected and rotated. In the noise-free case, the coordinates $p_i^{(k)} = (x_i^{(k)}, y_i^{(k)})^T$ agree with the global positioning $p_i = (x_i, y_i)^T$ up to some translation $t^{(k)}$ (unique to all i in V_k).

the known local frame coordinates of node $i \in V_k$ in the embedding of patch P_k (see Figure 2.9).

At this stage of the algorithm, each patch P_k has been properly reflected and rotated so that the local frame coordinates are consistent with the global coordinates, up to a translation $t^{(k)} \in \mathbb{R}^2$. In the noise-free case we should therefore have

$$p_i = p_i^{(k)} + t^{(k)}, \quad i \in V_k, \quad k = 1, \ldots, N. \tag{2.12}$$

We can estimate the global coordinates $p_1, .., p_n$ as the least squares solution to the overdetermined system of linear equations (2.12), while ignoring the by-product translations $t^{(1)}, ..., t^{(N)}$. In practice, we write a linear system for the displacement vectors $p_i - p_j$ for which the translations have been eliminated. Indeed, from (2.12) it follows that each edge $(i, j) \in E_k$ contributes a linear equation of the form [6]

$$p_i - p_j = p_i^{(k)} - p_j^{(k)}, \quad (i, j) \in E_k, \quad k = 1, \ldots, N. \tag{2.13}$$

In terms of the x and y global coordinates of nodes i and j, (2.13) is equivalent to

$$x_i - x_j = x_i^{(k)} - x_j^{(k)}, \quad (i, j) \in E_k, \quad k = 1, \ldots, N, \tag{2.14}$$

$$y_i - y_j = y_i^{(k)} - y_j^{(k)}, \quad (i, j) \in E_k, \quad k = 1, \ldots, N. \tag{2.15}$$

We solve these two linear systems separately, once for $x_1, \ldots, x_n$ and once for $y_1, \ldots, y_n$. Let T be the least squares matrix associated with the overdetermined linear system in

[5] Not to be confused with $G(i) = (V(i), E(i))$ defined in the beginning of this section.

[6] In fact, we can write such equations for every $i, j \in V_k$ but choose to do so only for edges of the original measurement graph.

(2.14), x be the $n \times 1$ vector representing the x-coordinates of all nodes, and b^x be the vector with entries given by the right-hand side of (2.14). Using this notation, the system of equations given by (2.14) can be written as

$$Tx = b^x, \tag{2.16}$$

and similarly (2.15) can be written as

$$Ty = b^y. \tag{2.17}$$

Note that the matrix T is sparse with only two non-zero entries per row and that the all-ones vector $\mathbf{1} = (1, 1, \ldots, 1)^T$ is in the null space of T, i.e., $T\mathbf{1} = 0$, so we can find the coordinates only up to a global translation.

To avoid building a very large least squares matrix, we combine the information provided by the same edges across different patches in only one equation, as opposed to having one equation per patch. This is achieved by adding up all equations of the form (2.14) corresponding to the same edge (i, j) from different patches, into a single equation, i.e.,

$$\sum_{k \in \{1,\ldots,N\}\ s.t. (i,j) \in E_k} x_i - x_j = \sum_{k \in \{1,\ldots,N\}\ s.t. (i,j) \in E_k} x_i^{(k)} - x_j^{(k)}, \quad (i,j) \in E, \tag{2.18}$$

and similarly for the y-coordinates using (2.15). We denote the resulting $m \times n$ matrix by $\tilde{T}$, and its $m \times 1$ right-hand-side vector by $\tilde{b}^x$. Note that $\tilde{T}$ has only two nonzero entries per row [7]. The least squares solution $\hat{p}_1, \ldots, \hat{p}_n$ to

$$\tilde{T}x = \tilde{b}^x, \text{ and } \tilde{T}y = \tilde{b}^y, \tag{2.19}$$

is our estimate for the coordinates $p_1, \ldots, p_n$ up to a global rigid transformation. Figure 2.10 shows the original and estimated embedding (after rigid alignment), and the histogram of errors in the coordinates, where the error associated with node i is given by $\| p_i - \hat{p}_i \|$.

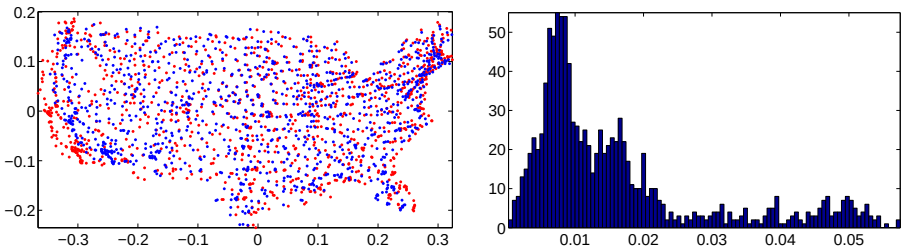
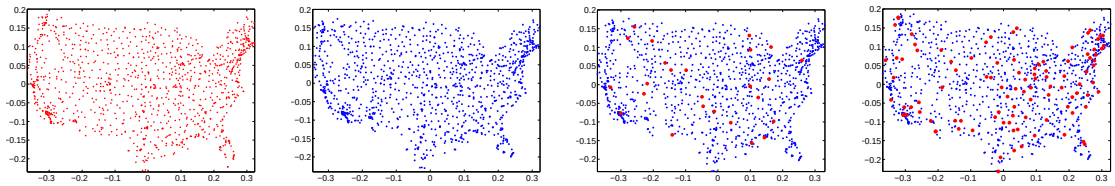


Figure 2.10: Left: Estimated embedding (blue) after alignment with the true positions (red) for the US cities graph with $\rho = 0.032$ and noise level $\eta = 20\%$. Right: Histogram of the errors $\|p_i - \hat{p}_i\|$.

[7] Note that some edges in E may not be contained in any patch P_k, in which case the corresponding row in $\tilde{T}$ has only zero entries.

The 2D-ASAP algorithm can easily integrate the information provided by anchors, if those exist. First, in the pre-processing step, if two or more anchors are contained in a patch, then this information can be used in localizing that patch. In Step 1, the relative reflection is solely determined for pairs of patches that have three or more anchor points in their intersection. Similarly, in Step 2, the relative rotation is determined for pairs of patches that have two or more anchors points in their intersection. In Step 3 we incorporate such information in the least squares method. Suppose we have obtained a reconstruction without using the anchor information. Since the anchors take their coordinates from the original embedding (which is a rigid transformation of our reconstruction) we first need to properly align the anchors with respect to our reconstruction. Then, for every node i that is an anchor, we simply substitute the unknowns $x_i^{(k)}$ in equation (3.13) with their true known value, and solve for the remaining unknowns (and similarly for the y-coordinates). Figure 2.11 shows reconstructions of the US cities map with noise $\eta = 0.2$ and different number of anchors.



Original graph | No anchors $ANE = 0.091$ | 25 anchors $ANE = 0.028$ | 100 anchors $ANE = 0.021$

Figure 2.11: Reconstruction of the US cities graph with $\rho = 0.032$ and noise level $\eta = 20\%$ for different number of anchors points. The average normalized error (ANE) is defined in (2.29).

Another way of including the anchor information in Step 3 is to substitute the anchor points p_i in equation (2.13) by $Op_i + t$, where p_i is known, while O is an unknown 2×2 matrix, accounting for the possible rotation and reflection of the anchors with respect to the reconstruction (although in our solution we cannot restrict O to be an orthogonal matrix), and t is an unknown 2×1 vector for the possible translation. Upon this substitution, (2.13) becomes a linear system of equations for the coordinates of the non-anchor points, for the entries of the matrix O and for the vector t. Note that this linear system can still be solved for the x-coordinates and the y-coordinates separately.

2.4 Finding and localizing globally rigid patches

In this section we turn to the problem of finding and localizing patches, which is a crucial preprocessing step of our algorithm. Most localization algorithms that use a local to global approach, such as PATCHWORK, LRE and ARAP, simply define patches by associating with every node i its entire 1-hop neighborhood $G(i)$. It is possible, however, that the subgraph $G(i)$ of 1-hop neighbors of vertex i is not globally rigid. In such a case, $G(i)$ has more than one possible realization in the plane. Therefore, whenever $G(i)$ is not globally

rigid, we find its maximally globally rigid components, which we call patches. The number of resulting patches can be 0, 1, or greater than 1. We note that [56] also suggested a method for breaking up networks into maximally globally rigid components. However, as we show below, breaking up the 1-hop neighborhood subgraph $G(i)$ is easier than breaking up a general graph, by utilizing recent results of [28] about the global rigidity property of cone graphs.

Star graph. We call a *star graph* a graph which contains at least one vertex that is connected to all remaining nodes. Note that in our definition, unlike perhaps more conventional definitions of star graphs, we allow edges between non-central nodes to exist. Note that for each node i, the local graph $G(i)$ composed of the central node i and all its neighbors takes the form of a star graph.

k-connectivity. A graph is *k-vertex-connected* iff it remains connected even after the removal of any $k-1$ vertices. Alternatively, a graph is k-vertex-connected iff every pair of vertices is connected by at least k disjoint paths. In a planar network, a necessary condition for global rigidity is 3-vertex-connectivity [55], meaning that the graph should remain connected after the removal of any two vertices. Note that 3-vertex-connectivity implies that the minimum degree of the graph is three, since any vertex of lower degree can be disconnected from the graph by removing its neighbors. An alternative characterization is in terms of cuts, also known as splitting pairs. A graph that is not 3-vertex-connected has a vertex cut of size two, i.e., a pair of vertices whose removal disconnects the graph into two separated components. A graph with a cut of size two is not globally rigid since one of the two components can be flipped across the line determined by the splitting pair. A similar definition holds for k-edge-connectivity, where a graph is said to be *k-edge-connected* if there is no set of $k-1$ edges whose removal disconnects the graph, and the smallest such k denotes the edge-connectivity of the graph. Note that if a graph is k-vertex-connected then it is also q-edge-connected for $q \leq k$.

Proposition 2.4.1. *A star graph is generically globally rigid in $\mathbb{R}^2$ iff it is 3-vertex-connected.*

Proof. The process of coning a graph G adds a new vertex v, and adds edges from v to all original vertices in G, creating the cone graph $G * v$. A recent result of [28] states that a graph is generically globally rigid in $\mathbb{R}^{d-1}$ iff the cone graph is generically globally rigid in $\mathbb{R}^d$.

Let H be a 3-vertex-connected star graph, v be its center node, and H^* the graph obtained by removing node v, $H^* = H \backslash v$. Since H is 3-vertex-connected then H^* must be 2-vertex-connected, since otherwise, if u is a cut-vertex in H^*, then $\{v, u\}$ is a vertex-cut of size 2 in H, a contradiction. Since the vertex connectivity of a graph cannot exceed its edge-connectivity, it follows that H^* is at least 2-edge-connected, which is a necessary and

sufficient condition for generic global rigidity on the line. Using the coning theorem, the generic globally rigidity of H^* in $\mathbb{R}^1$ implies that H is generically globally rigid in $\mathbb{R}^2$. On the other hand, as mentioned before, if H is generically globally rigid, then it must be 3-vertex-connected. □

Using Proposition 2.4.1, we propose the following simple algorithm for breaking up a star graph into maximally globally rigid components. We first remove all vertices of degree one, since no globally rigid subgraph can contain such a vertex. Note that a vertex of degree two can be only be contained in a triangle, provided its two neighbors are connected. Next, we search for the (maximal) 3-connected components in the graph, taking advantage of its structure as a star graph. In other words, we are looking for a decomposition of the graph into a union of 3-vertex-connected subgraphs of maximal size. For the case of star graphs, the following approach leads to a simple and efficient algorithm. We look for a cut set (of size one or two) containing the center node that separates the graph into two or more components, and recurse on each one of them. In order to check for the 3-connectivity of a given 1-hop neighborhood star graph $G(i)$, it suffices to remove the center node i and check if the remaining graph $G(i) \setminus \{i\}$ is 2-connected, which can be done in $O(m')$ time, where m' is the number of edges in $G(i) \setminus \{i\}$.

Figure 2.12 shows an example of a 1-hop neighborhood graph where the center node is connected to all its neighbors in the measurement graph. The neighborhood graph has four 3-connected components that share edges and vertices, each component being a star graph and hence globally rigid by the above result. Note that a globally rigid patch is allowed to be as small as a triangle.

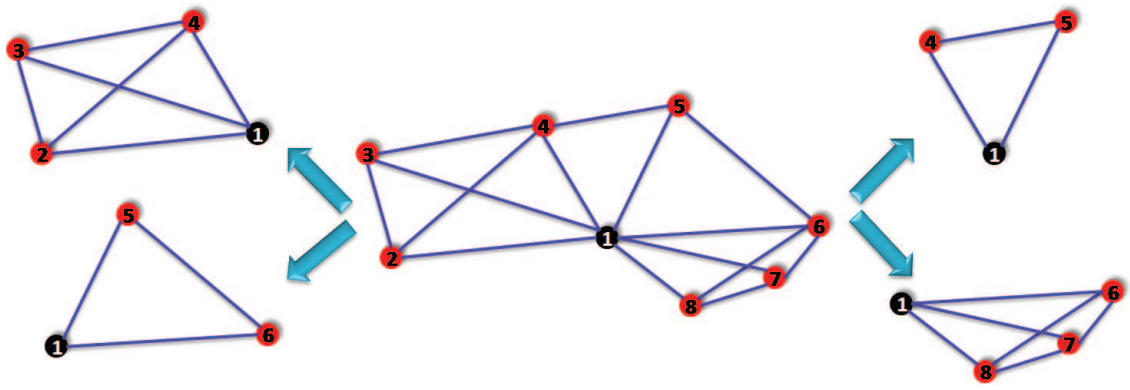


Figure 2.12: The neighborhood graph of center node 1 is split into four maximally 3-connected-components (patches): $\{1, 2, 3, 4\}, \{1, 4, 5\}, \{1, 5, 6\}, \{1, 6, 7, 8\}$.

After finding the patches, it still remains to localize each of them separately in the plane. Localizing a small globally rigid subgraph is significantly easier in terms of speed and accuracy than localizing the whole measurement graph. First, the size of a patch is significantly smaller than the size of the whole network. For example, the typical patch

size for the US cities graph with $n = 1090$ and sensing radius $\rho = 0.032$ is between 10 to 30 nodes, as shown in Figure 2.13 (left panel). Also, when embedding locally, we are no longer constrained to a distributed computation that can impose additional challenges due to inter-sensor communication. Since each node in the patch is connected to a central node, all the information can be passed on to this node which will perform the computation in a centralized manner. Finally, under the assumptions of the disc graph model, it is likely that 1-hop neighbors of the central node will also be interconnected, rendering a relatively high density of edges for the patches, as indicated by Figure 2.13 (right panel). This means that locally, the partial distance matrix of a typical patch usually has only a small number of missing entries, which makes the embedding of the patch more robust to noise and more efficient to compute. We have also observed in our experimental simulations that SDP localization algorithms tend to run considerably faster when the partial distance matrix is denser.

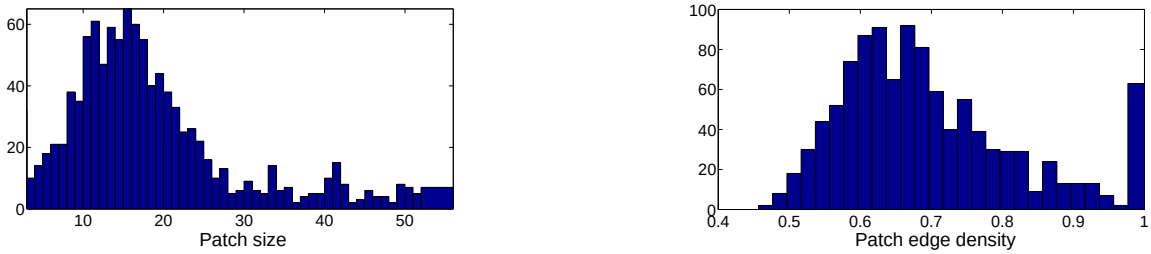


Figure 2.13: Histogram of patch sizes (left) and edge density (right). US cities map, $n = 1090$ and noise $\eta = 20\%$ ($deg = 20$).

After experimenting with the different localization methods, our method of choice for embedding the patches was the three-stage procedure described in [46], due to its relatively low running time and its robustness to noise for small patches. When used for small patches (e.g. of size 20-30) rather than the entire network, the stress minimization is more reliable and less sensitive to local minima. Compared to an anchor-free SDP localization algorithm like SNL-SDP[8], it produces similar results in terms of the localization error, but with lower running times (see Figure 2.14). To the best of our knowledge, the SDP based approaches (in particular those in [17, 13, 14, 95, 96, 114]) have not been analyzed in the context of the disc graph model, and the SDP localization theory is built only on the known distances, without any additional lower and upper bounds that can be inferred from the disc graph assumption. However, experimental results reported by the same authors (personal communication) reveal that when adding such additional constraints into the SDP formulation, the localizations become more accurate at the cost of increased running time.

The three-stage algorithm of [46] first estimates the missing distances d'_{ij} for $(i, j) \notin E_k$ by making use of the disc graph assumption (for the lower bound) and the triangle inequality (for the upper bound). Second, the coordinates are computed by running the classical MDS on the complete set of pairwise distances. Third, the embedding is improved

[8]We used the SNL-SDP code of [101].

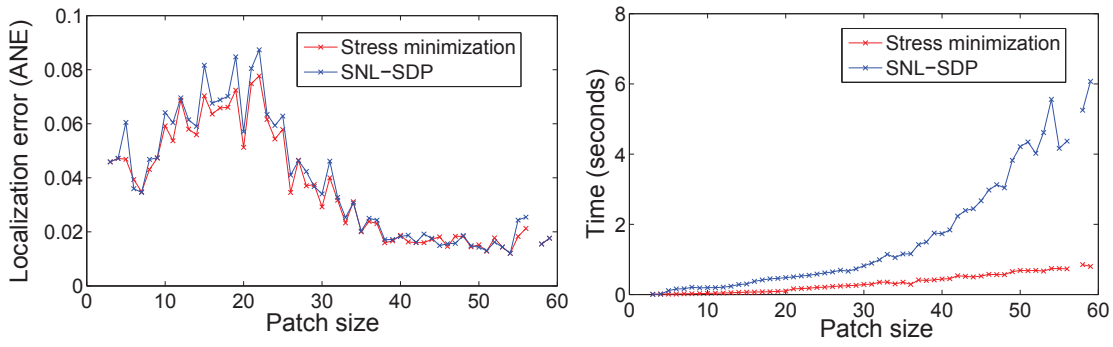


Figure 2.14: Comparison of the three-stage algorithm labeled Stress minimization and SNL-SDP for the US cities graph with $\rho = 0.032$ and noise level $\eta = 20\%$.

by running the stress minimization algorithm based only on the initial distances, but not on the estimated missing distances:

Stage 1. Estimating missing distances. For each missing distance d'_{ij} with $(i,j) \notin E_k$, we denote its lower bound estimate (respectively upper bound) by $\underline{d_{ij}}$ (respectively $\overline{d_{ij}}$). Using the triangle inequality on all pairs of existing edges $(i,k),(j,k) \in E_k$, an upper bound on d'_{ij} is given by

$$\overline{d_{ij}} = \min_{k:(i,k),(j,k)\in E_k} d_{ik} + d_{jk}. \tag{2.20}$$

Using the disc graph model assumption, a lower bound $\underline{d_{ij}}$ is given by

$$\underline{d_{ij}} = \max\{\max_{k:(i,k)\in E_k}\{d_{ik}\}, \max_{k:(j,k)\in E_k}\{d_{jk}\}\}. \tag{2.21}$$

The missing distances are estimated as $d'_{ij} = \frac{\underline{d_{ij}}+\overline{d_{ij}}}{2}$, for $(i,j) \notin E_k$.

Stage 2. Classical MDS. After estimating all missing distances, classical MDS algorithm [30] is used on the complete set of pairwise distances to compute local coordinates of all nodes of the patch.

Stage 3. Stress minimization. The embedding obtained from classical MDS is refined using the stress majorization algorithm (mentioned in Section 2.2). The stress function in (2.3) is minimized by running the iterative majorization technique described in [46]. At each iteration, the coordinates of each node are updated according to the following rule

$$p_i \longleftarrow \frac{1}{deg_i(P_k)} \sum_{j\in V_k,(i,j)\in E_k} [p_j + d_{ij}(p_i - p_j)\,\mathrm{inv}(\| p_i - p_j \|)] \tag{2.22}$$

where $deg_i(G_k)$ denotes the degree of node i in patch P_k, and

$$\mathrm{inv}(x) = \begin{cases} 1/x & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases}. \tag{2.23}$$

We remark that we use classical MDS for patches that have no missing edges. Note that some patches can be much larger than others, rendering their embedding less accurate. We therefore restrict the size of the patches to some maximal prescribed size.

2.5 Methods for aligning patches in Steps 1 and 2

In this section we describe several methods for aligning patches and for computing their relative reflections and rotations. Successful alignment of patches is important, since in order for the eigenvector method to succeed, the Z and R matrices from Steps 1 and 2 of 2D-ASAP need to have enough correct, or approximately correct, entries. Given two patches P_i and P_j, each embedded in its own coordinate system, we are first interested in estimating their relative reflection z_{ij}, where $z_{ij} = -1$ if P_i needs to be replaced by its mirrored image before being aligned with P_j, and $z_{ij} = 1$ if the two patches can be aligned via an angular rotation and translation without a reflection. Second, we are interested in estimating the offset angle $\theta_{ij} = \theta_i - \theta_j \mod 2\pi$ that aligns the two patches. Obviously, two patches that are far apart and have no common nodes cannot be aligned, and there must be enough overlapping nodes to make the alignment possible. Figure 2.13 shows a typical example of the sizes of the patches we consider, as well as their intersection sizes. As expected, in the case of the disc graph model, the overlap is often small. It is therefore crucial to have robust alignment methods even when the overlap size is small.

A closed form solution to the registration problem in any dimension was given by [59], where the best rigid transformation between two sets of points is obtained by various matrix manipulations and eigenvalue/eigenvector decomposition. In our approach described in the following paragraph, we choose to convert this non-linear regression problem to a linear complex least squares problem by using complex numbers to denote 2-by-2 rotation matrices.

Least squares registration. Given two patches P_k and P_l that have at least three nodes in common, the registration process finds the optimal 2D rigid motion of P_l that aligns the common points (as shown in Figure 2.3). We denote by $V_{k,l} = \{v_1, \ldots, v_s\}$ the nodes in the intersection of patches P_k and P_l, i.e., $V_{k,l} = V_k \cap V_l$. We let $p_1^{(k)}, \ldots, p_s^{(k)}$ be the coordinates of the set of nodes $V_{k,l}$ in the embedding of patch P_k, and similarly $p_1^{(l)}, \ldots, p_s^{(l)}$ be the coordinates of nodes $V_{k,l}$ in the embedding of patch P_l. For a point $p_i^{(k)} = (x_i^{(k)}, y_i^{(k)})$ we denote by $\bar{p}_i^{(k)} = (x_i^{(k)}, -y_i^{(k)})$ its mirrored image across the x-axis. For the purpose of the minimization problems we are about to describe, it is convenient to view the local frame of each patch as the complex plane $\mathbb{C}$ instead of the Euclidean space $\mathbb{R}^2$. We write the coordinates of a node $p_i^{(k)} = (x_i^{(k)}, y_i^{(k)})$ as $p_i^{(k)} = x_i^{(k)} + \imath y_i^{(k)}$, and represent its mirrored image by $\bar{p}_i^{(k)} = x_i^{(k)} - \imath y_i^{(k)}$.

Given two sets of planar labeled points $\{p_1^{(k)}, \ldots, p_s^{(k)}\}$ and $\{p_1^{(l)}, \ldots, p_s^{(l)}\}$ (viewed as elements of $\mathbb{C}$), the registration problem is to find a rotation $r_\theta = e^{\imath\theta}$ and a translation vector $t = x + \imath y$, that finds the optimal alignment of the two sets of points in the least squares sense. In other words, we are interested in finding r_θ and t that minimize the following

objective function

$$f(\theta, t) = \sum_{i=1}^{s} |p_i^{(k)} - (r_\theta p_i^{(l)} + t)|^2. \tag{2.24}$$

Since we do not know a priori the relative reflection z_{ij} of the pair of patches P_k and P_l, we use the registration method twice. We first register P_k and P_l by minimizing (2.24), and then register P_k and $\bar{P}_l$, the mirrored image of patch P_l, by minimizing a similar objective function

$$\tilde{f}(\theta, t) = \sum_{i=1}^{s} |p_i^{(k)} - (r_\theta \bar{p}_i^{(l)} + t)|^2. \tag{2.25}$$

If the residual in the minimization (2.24) is smaller than the residual in the minimization (2.25), then the two patches are properly oriented, otherwise one of the two patches needs to be replaced by its mirrored image. In other words, we define z_{ij} as

$$z_{ij} = \begin{cases} 1 & \text{if } \min_{\theta,t} f(\theta, t) \leq \min_{\theta,t} \tilde{f}(\theta, t) \\ -1 & \text{if } \min_{\theta,t} f(\theta, t) > \min_{\theta,t} \tilde{f}(\theta, t) \\ 0 & \text{if } P_i \text{ and } P_j \text{ cannot be aligned (intersection is too small)} \end{cases} \tag{2.26}$$

We rewrite (2.24) (and similarly for (2.25)) as $\|Ax - b\|^2$, where

$$A^T = \begin{pmatrix} p_1^{(l)} & \dots & p_i^{(l)} & \dots p_s^{(l)} \\ 1 & \dots & 1 & \dots 1 \end{pmatrix}, \quad b^T = \begin{pmatrix} p_1^{(k)} & \dots & p_i^{(k)} & \dots p_s^{(k)} \end{pmatrix}, \quad x^T = [r_\theta \;\; t].$$

Therefore, we solve the minimization problem of (2.24) by the method of least squares and find r_θ and t. By solving the registration problem using complex least squares, we are guaranteed to recover the optimal solution (best 2D rigid transformation) up to scaling [87]. For noisy distance measurements, r_θ does not necessarily lie on the unit circle, in which case we extract its phase θ (but ignore its amplitude). For noisy data, the registration method becomes significantly more robust if the pair of patches have a large overlap (e.g. at least 6 or 7 nodes). Also, note that for computing the relative reflection, the two patches must overlap in at least $s \geq 3$ nodes, while for estimating the rotation (after finding the proper rotation) it suffices to have $s \geq 2$.

Combinatorial Score. The second alignment method we consider makes use of the underlying assumptions of the disc graph model. Specifically, we exploit the information in the non-edges that correspond to distances larger than the sensing radius ρ. The resulting method can be used to estimate both the relative reflection and rotation for a pair of patches that overlap in just two nodes (or more).

Consider two overlapping patches P_k and P_l that intersect at only two nodes $\{a, b\} \in V_k \cap V_l$. We would like to decide whether the two patches have the same orientation with respect to the original complete network, or rather P_l needs to be replaced by its mirrored image. As illustrated in Figure 2.15, there are two possible ways to align the two patches using the common edge ab, in terms of their relative orientation. One with P_k and P_l as they appear on the left-hand side of the figure, and one where the P_l patch is reflected across edge ab, shown on the right-hand side of the figure. Only one of the two scenarios is feasible, and to decide which one, we make use of the disc graph model assumption that

two nodes are connected iff their distance does not exceed ρ. For each of the two scenarios, we count the number of violations of the disc graph assumption. There are two types of violations: distances that are predicted by the patch alignment to be smaller than ρ but are missing from the original measurement graph, and distances that are predicted by the patch alignment to be greater than ρ but also appear in the original measurement graph. One of the two scenarios will correspond to a foldover in the graph, causing nodes that were far apart in the original graph to become within sensing radius of each other (causing false edges), and nodes that were close in the original graph to become far apart (thus leading to missing edges). Of the two scenarios, we choose the one with the smaller number of violations.

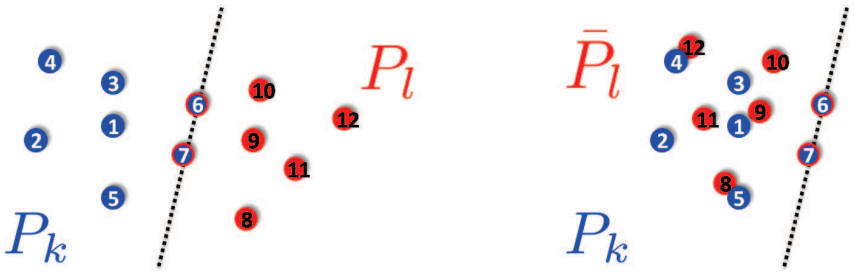


Figure 2.15: Using the combinatorial method to decide on the relative reflection of two patches: aligning P_k and P_l (left), and P_k and $\bar{P}_l$ (right).

Link edges. The last alignment method we consider is useful whenever two patches have a small overlap, but there exist many cross edges in the measurement graph that connect the two patches. Suppose the two patches P_k and P_l overlap in at least one vertex, and call a *link edge* an edge $(u, v) \in E$ that connects a vertex u in patch P_k (but not in P_l) with a vertex v in patch P_l (but not in P_k). We denote the number of link edges by q. Figure 2.16 shows two patches overlapping in only one vertex that have $q = 3$ such link edges.

First, in order to factor out the translation, we align the center of mass of the intersection points. The next step is to find the rotation that optimally aligns the two patches. Of course, if there are enough common nodes (at least three) one can use the registration method to obtain the rotation angle. However, when the overlap size is small (up to four or five nodes), the results of the registration method are not very accurate in the presence of large noise. We want to be able to align patches robustly, even when they have only one or two common nodes, because there are many pairs with small overlap size, as the right panel of Figure 2.13 indicates. Each link edge adds a constraint between the two patches, and we would like to compute the optimal rotation angle that satisfies the link edge constraints as best as possible. If we denote the coordinates of node u_i in patch P_k by $p_{u_i}^{(k)}$, the penalty function we minimize is:

$$F(\theta) = \sum_{i=1}^{q} (|p_{u_i}^{(k)} - r_\theta p_{v_i}^{(l)}|^2 - d_{u_i,v_i}^2)^2 \tag{2.27}$$

where $r_\theta = e^{\imath\theta}$. Setting the derivative $F'(\theta) = 0$, we arrive at a cubic for r_θ, and we pick the root that gives the minimum value for $F(\theta)$. To decide on the relative reflection for a

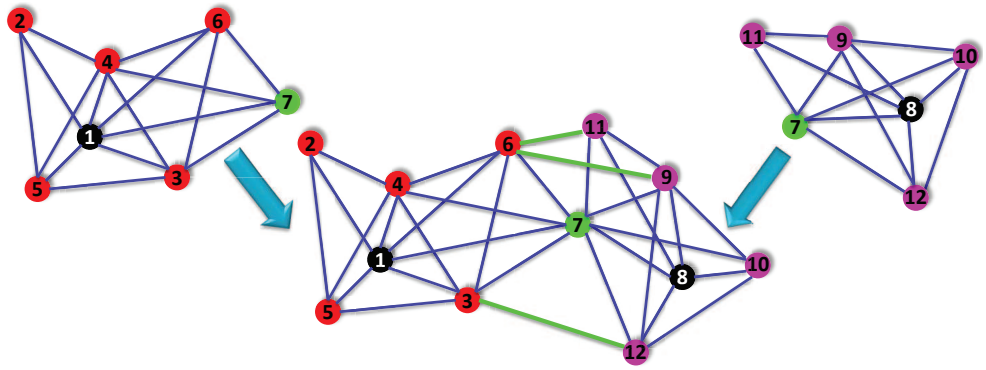


Figure 2.16: Alignment of two patches P_i and P_j overlapping in just one node using link edges (green).

pair of patches, we run this minimization twice, once for patches P_k and P_l, and a second time for patches P_k and $\bar{P}_l$. Whichever setup gives a smaller global minimum, indicates the correct relative reflection of the two patches.

The three registration methods considered above are useful in different scenarios. In practice, we only use the least squares registration method, which turns out to be the most robust to noise whenever the overlap between patches is large enough (e.g., 6 overlapping nodes or more). However, in some cases, that we report in the proceeding section, we also use the combinatorial method that is useful when the overlap is small (e.g., 2 nodes or more). Although the link edges method is useful when the node overlap is just 1, we did not use it in practice, as in our experiments all patch graphs were already connected without using it. The link edges method is important to maintain connectivity of the patch graph when the input measurement graph is very sparse. Figure 2.17 shows a histogram of the intersection sizes between pairs of patches in the US cities graph.

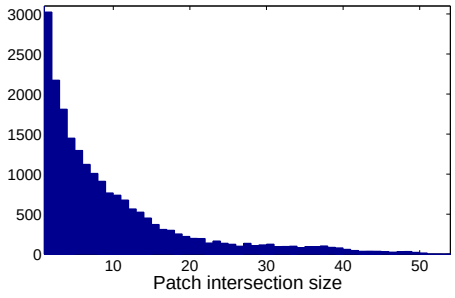


Figure 2.17: Histogram of the intersection size of patches in the US cities graph $\rho = 0.032$ and $\eta = 20\%$.

2.6 Complexity Analysis

In this section we give a complexity analysis of each step of the 2D-ASAP algorithm, showing that the time complexity scales almost linearly in the size of the network (number of nodes n and edges m), and augment this theoretical analysis with the running times of numerical simulations for the localization of networks of increasing sizes ($n = 10^3, 10^4, 10^5$) as detailed in Table 2.11. Tables 2.2 and 2.3 summarize the notation used throughout this section, respectively the complexity of each step of the 2D-ASAP algorithm.

n	# of nodes in G, $\|V\| = n$	
m	# of edges in G, $\|E\| = m$	
k	upper bound on the size of a patch (user input)	
N	# of patches (nodes in the patch graph G^P), $\|V^P\| = N$	$N \leq n(k-1)$
M	# of pairs of overlapping patches (edges in G^P), $\|E^P\| = M$	$M \leq Nk^2/2$
d^P	maximum degree in G^P	$d^P \leq k^2$
m'	maximum # of edges in a patch	$m' \leq k^2/2$

Table 2.2: Summary of notation used to describe the graph of sensors $G = (V, E)$ and the patch graph $G^P = (V^P, E^P)$.

Stage	Complexity	# of calls
Break 1-hop neighborhood into patches	$O(k + m')$	n
Patch embedding by SMACOF	$O(k^3 + k^{3/2}t)$	N
Patch embedding by FULL-SDP	$O(k^2 m'^{2.5} + m'^{3.5})$	N
Patch intersection	$O(k)$	M
Patch alignment	$O(k)$	M
Top eigenvector computation	$O(M\zeta)$	2
Linear least squares	$O(m\sqrt{\kappa})$	2
Total	$O(n\ poly(k, m', t, \zeta) + m\sqrt{\kappa})$	

Table 2.3: Summary of the complexity of each step of the 2D-ASAP algorithm. t denotes the number of iterations of the SMACOF algorithm, ζ the number of iterations of the power method, and κ the condition number of the matrix $T^T T$ (where T is the least squares matrix from Step 3).

Preprocessing Step: Finding and localizing globally rigid patches. Breaking up the graph into maximally globally rigid components was presented in detail in Section 2.4, and represents the first computationally expensive task in 2D-ASAP. In light of Proposition 2.4.1, to check for the 3-connectivity of a given 1-hop neighborhood star graph $G(i)$, it suffices to remove the center node i and check if the remaining graph $G(i) \setminus \{i\}$ is 2-connected, otherwise extract its 2-connected components. Partitioning a graph into 2-connected components can be done in time linear in the number of nodes and number of edges of the graph. The (worst-case) complexity of this step is therefore $O(k + m')$. For convenience, we did not use an $O(m')$ implementation, but rather the $O(m'^2)$ naïve algorithm that looks for cuts in the graph by examining all possible pairs of nodes. Despite the

expected linear scaling of this step of the algorithm, the running times reported in Table 2.11 do not seem to scale linearly, but we are able to explain this discrepancy as follows. In our experiments, the average patch size remains approximatively the same (e.g. ≈ 13) for $n = \{10^3, 10^4, 10^5\}$, and the maximum patch sizes are $k = \{24, 28, 31\}$. Since the number of patches M is bounded by nk, we attribute the slow running times to MATLAB's added overhead when working with arrays of structures. A similar behavior can be observed in the *Patch intersections* preprocessing step, where we compute and store the intersection of pairs of overlapping patches. As shown in the following paragraphs, each patch overlaps with a constant number of other patches, and thus the number of patch intersections to compute and store scales linearly in the number of patches. We expect that an efficient implementation in C of these steps of the algorithm will scale linearly.

The next question we address is whether the resulting number of patches N is linear in the number of nodes n. We answer this question in the affirmative, and show in the following analysis that $N \leq n(k-1)$ where k is the user chosen upper bound on the size of a patch. Denote by $P_1, P_2, \ldots P_r$ the maximally globally rigid components in the 1-hop neighborhood graph $G(i)$ of a given node i, with $|P_l| \geq 3$ and $|P_l| \leq k \; \forall l = 1, \ldots, r$, since we restrict the size of the 1-hop neighborhood to be at most k. Note that the union of two globally rigid graphs P_i and P_j that intersect in at least $d+1 = 3$ nodes is itself a globally rigid graph. This observation together with the maximality condition on the patch sizes imply that any two patches intersect in at most two vertices $|P_i \cap P_j| \leq 2$, as otherwise their union is a globally rigid component in $G(i)$, and neither P_i nor P_j would be maximal. In addition, whenever a pair of patches overlap in 2 vertices it must be the case that one of the two vertices is the center node i. If one were to draw an imaginary line through all edges originating at i (there are $k-1$ such edges), and think of the resulting sectors (slices) as building blocks for the patches, then it becomes clear that a patch is comprised of adjacent sectors and has a left-end and a right-end edge. This also means that a patch overlaps with at most 2 other patches, and the intersection is given by the left-end and right-edges. Since there are $k-1$ edges originating at i, it means that there are at most $k-1$ patches contributed by the 1-hop neighborhood of any node.

The running time for localizing the patches depends on the embedding method of choice, SMACOF or FULSDP. In terms of complexity, [4] show that the SMACOF algorithm runs in $O(k^3 + k^{3/2}t)$ time and $O(k)$ space, where t is the number of iterations required to minimize the stress energy function introduced earlier in (2.3). In our experiments, we limit the maximum size of a patch to a constant $k \approx 30 - 50$ by including in the 1-hop neighborhood of node i only the $k-1$ nearest neighbors of i (if i has more than $k-1$ neighbors). However, if we choose to use the SDP approach for embedding the patches, this task is more expensive since the computation complexity of SeDuMi (the SDP solver used here) is $O(k^2 m'^{2.5} + m'^{3.5})$, since there are k number of decision variables (nodes of a patch) and m' linear matrix (in)equality constraints (edges of a patch) [81]. In either scenario, the embedding of a single patch remains polynomial in k, and the complexity of the preprocessing step adds up to $O(N \; poly(k))$, since there are N patches for which we check biconnectivity and compute their embedding.

Steps 1 and 2: Computing Reflections and Rotations The computationally expensive tasks in Steps 1 and 2 are the registration of pairs of overlapping patches, and the computation of the top eigenvectors of sparse N-by-N matrices. The registration method introduced

at the beginning of Section 2.5 amounts to solving a complex linear least squares problem, of the form $Ax = b$, where A is a matrix of size $s \times 3$, and s is the number of points in the intersection of the two patches. The least squares solution is given by $x = (A^T A)^{-1} A^T b$, which can be computed in $O(s)$ time. Since $s \leq k$, the overall complexity of aligning two patches using least squares is $O(k)$. Concerning the eigenvector computation, we note that every iteration of the power method is linear in the number of edges M of the patch graph G^P, but the number of iterations is greater than $O(1)$ as it depends on the spectral gap. Note that a pair of patches P_i and P_j overlap if and only if j is either a 1-hop or 2-hop neighbor of i. Since we limit the number of 1-hop neighbors of a node to k, it follows that the number of 2-hop neighbors is at most k^2. In other words, k^2 is an upper bound for the maximum degree in the patch graph G^P, and we conclude that the number of edges M does not exceed $Nk^2/2$, where N grows linearly in n. Note that in the analysis above we assumed each node contributes with one patch, but the result $M = O(N)$ still remains valid if $G(i)$ generates multiple patches (a constant depending on k).

Step 3: Least Squares. To estimate the x and y-axis translations and compute the final coordinates of the reconstruction, we solve the linear least square problems in (3.15). One possible approach for solving such linear least squares problems of the form $Tx = b$ is to use conjugate gradient iterations applied to the normal equations $T^T Tx = T^T b$ (which can be done without explicitly doing the expensive computation of the matrix $T^T T$). The rate of convergence of the gradient iterations is determined by the condition number κ of the matrix $T^T T$, and the number of iterations required for convergence is $O(\sqrt{\kappa})$ [103]. For matrices that are sparse or have exploitable structure, each conjugate gradient iteration has complexity as low as $O(m)$. Recall that in our case T is a sparse matrix with only two nonzero entries per row. Overall, the complexity of the linear least squares in Step 3 in our case is $O(m\sqrt{\kappa})$.

Adding up the complexity of all steps of the algorithm, we get a running time of $O(n\ poly(k, m', t, \zeta) + m\sqrt{\kappa})$, which is almost linear in the size of the network.

2.7 Experimental results

We have implemented our 2D-ASAP algorithm and compared its performance with other methods across a variety of measurement graphs, varying parameters such as the number of nodes, average degree (sensing radius) and level of noise.

In our experiments, the noise is multiplicative and uniform, meaning that to each true distance measurement $l_{ij} = \| p_i - p_j \|$, we add random independent noise ϵ_{ij} in the range $[-\eta l_{ij}, \eta l_{ij}]$, i.e.,

$$\begin{aligned} d_{ij} &= l_{ij} + \epsilon_{ij} \\ \epsilon_{ij} &\sim Uniform([-\eta l_{ij}, \eta l_{ij}]) \end{aligned} \tag{2.28}$$

The percentage noise added is 100η, (e.g., $\eta = 0.1$ corresponds to 10% noise).

The sizes of the graphs we experimented with range from 200 to 10^5 nodes taking different shapes, with average degrees as low as 6.8, and noise levels up to 70%. Across all

our simulations, we consider the disc graph model, meaning that all pairs of sensors within range ρ are connected. We denote the true coordinates of all sensors by the $2\times n$ matrix $P = (p_1 \ \cdots \ p_n)$, and the estimated coordinates by the matrix $\hat{P} = (\hat{p}_1 \ \cdots \ \hat{p}_n)$. To measure the localization error of our algorithm we first factor out the optimal rigid transformation between the true embedding P and our reconstruction $\hat{P}$ (using the registration method), and then compute the following average normalized error (ANE)

$$ANE = \frac{\sqrt{\sum_{i=1}^{n} \parallel p_i - \hat{p}_i \parallel^2}}{\sqrt{\sum_{i=1}^{n} \parallel p_i - p_0 \parallel^2}} = \frac{\parallel P - \hat{P} \parallel_F}{\parallel P - p_0 \mathbf{1}^T \parallel_F}, \tag{2.29}$$

where $p_0 = \frac{1}{n}\sum_{i=1}^{n} p_i$ is the center of mass of the true coordinates and the Frobenius norm of an $n_1 \times n_2$ matrix H is $\parallel H \parallel_F = \sqrt{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} |H_{ij}|^2}$. The normalization factor in the denominator of (2.29) ensures that the ANE is not only rigid invariant, but it is also scale free, i.e., it is invariant to scaling all the distances by a constant factor.

Figure 2.19 shows reconstructions of the US cities map at different levels of noise $\eta = 0\%, 10\%, 20\%, 30\%, 40\%, 50\%$. The number of nodes is $n = 1090$ and the sensing radius $\rho = 0.032$ leads to an average degree between 19 and 23 depending on the noise level η. A high average degree improves significantly the accuracy of the reconstruction even at high levels of noise. Table 2.4 shows various measurements of the errors in Steps 1 and 2, and indicates that the eigenvector method is able to correct many of the input errors: the errors $\mathcal{E}^{\mathcal{Z}}_{eig}$ and $\mathcal{E}^{\mathcal{R}}_{eig}$ are smaller than the input errors $\mathcal{E}^{\mathcal{Z}}_{inp}$ and $\mathcal{E}^{\mathcal{R}}_{inp}$. Table 2.5 compares the final reconstruction errors of 2D-ASAP, ARAP and MVU, from which we conclude that 2D-ASAP and ARAP usually give very similar reconstruction errors. While FAST-MVU performs reasonably well for the US cities graph, its performance deteriorates significantly for the other graphs considered below, despite the fact that unlike with the other tested algorithms, we allowed FAST-MVU to do the gradient descent final step for minimizing the stress function. We remark that ARAP performs several alternating least squares iterations (about 40) that refine the reconstruction until convergence, while 2D-ASAP is non-iterative and its resulting structure can be further improved with any refinement method. The ANEs reported in Table 2.5 for the US cities graph and for all other graphs are averaged over 10 runs with independent realizations of noise for the distances. Table 2.6 shows the running time of the various steps of the 2D-ASAP algorithm corresponding to our not particularly optimized MATLAB implementation on a PC machine equipped with an Intel(R) Core(TM)2 Duo CPU E8500 @ 3.16GHz 4 GB RAM. Notice that all steps are amenable to a distributed implementation, thus a parallelized implementation would significantly reduce the running times. Building $\mathcal{Z}$ takes more time than building $\mathcal{R}$, since in Step 1 the registration method is performed twice for every pair of overlapping patches, while in Step 2 it is performed only once. We also remark that for $\eta = 50\%$ we used the combinatorial method for aligning patches that overlap in 2,3,4 or 5 nodes, which resulted in a little improvement for the ANE from 0.57 to 0.54.

The C-shape graphs in Figure 2.20 have $n = 200$ nodes, sensing radius $\rho = 0.17$ leading to an average degree between 9 and 10, and were tested at noise levels $\eta = 0\%, 10\%, 20\%, 30\%, 40\%$. The similar C-shape graphs in Figure 2.21 have the same number of nodes, but $\rho = 0.28$, the average degrees are between 20 and 28 and the noise levels are $\eta = 35\%, 40\%, 50\%, 60\%, 70\%$. The simulation results in the two scenarios clearly

			Step 1 (Reflections)			Step 2 (Rotations)			
η	deg	M	τ	$\mathcal{E}^{\mathcal{Z}}_{inp}$	$\mathcal{E}^{\mathcal{Z}}_{eig}$	$\mathcal{E}^{\mathcal{R}}_{inp}$	$\mathcal{E}^{\mathcal{R}}_{eig}$	$Outliers^{\mathcal{R}}_{inp}$	$Outliers^{\mathcal{R}}_{eig}$
0%	19	10228	0%	0.1%	0%	0.2	0.2	0.3%	0.1%
10%	19	10589	0%	0.2%	0%	1.7	1.2	0.8%	0.3%
20%	20	11299	0.1%	0.7%	0.1%	4	2.6	7.2%	2.5%
30%	20	11730	2.4%	2%	0.4%	7.7	4.2	21.1%	8.9%
35%	21	12165	7.1%	3%	1.3%	17.4	7.7	36.8%	18.8%
40%	22	12479	7.1%	5.5%	2.9%	18.3	8.3	44.2%	23.1%
45%	23	14305	8.8%	10.4%	6.9%	22.6	11.1	53.8%	32.0%
50%	23	15295	10.5%	16.1%	10.8%	26.8	12.2	61.7%	39.1%

Table 2.4: Error rates for 2D-ASAP in Steps 1 and 2 for the US map with $n = 1090$ cities, $\rho = 0.032$ and number of patches $N \approx 1200$ depending on the noise level. M is the number of pairs of patches aligned (number of edges in the patch graph). τ denotes the percentage of patches incorrectly oriented, that is, the percentage of disagreements between $\hat{z}_i$ and z_i. $\mathcal{E}^{\mathcal{Z}}_{inp}$ is the percentage of errors in the matrix $\mathcal{Z}$, that is, the percentage of disagreements between z_{ij} and $z_i z_j^{-1}$ restricted to $(i,j) \in E^P$. $\mathcal{E}^{\mathcal{Z}}_{eig}$ is the percentage of disagreements between $\hat{z}_i \hat{z}_j^{-1}$ and $z_i z_j^{-1}$ also restricted to $(i,j) \in E^P$. $\mathcal{E}^{\mathcal{R}}_{inp}$ and $Outliers^{\mathcal{R}}_{inp}$ denote the average angle error (in degrees) and the number of outlier angles (errors above 10°) in $\theta_{ij} - (\theta_i - \theta_j) \mod 2\pi$ for $(i,j) \in E^P$. $\mathcal{E}^{\mathcal{R}}_{eig}$ and $Outliers^{\mathcal{R}}_{eig}$ denote the average angle error (in degrees) and the number of outlier angles (errors above 10°) in $\hat{\theta}_i - \hat{\theta}_j - (\theta_i - \theta_j) \mod 2\pi$ for $(i,j) \in E^P$.

illustrate a significant improvement in robustness to noise for the case of denser graphs. In addition to ARAP and FAST-MVU, we compare our results against the FULL-SDP algorithm [17], in three different scenarios. In the first two, we run FULL-SDP on the same measurement graph used by the other algorithms, but provide FULL-SDP with additional 3 and 10 anchors placed at random, that are not provided to the other algorithms. We choose the anchors at random from the set of all sensors. In the third scenario, we use a measurement graph of (approximately) the same average degree deg as the one used by the other algorithms, but allow FULL-SDP to use a much larger sensing radius $\rho = 1$. Each node has a large number of neighbors within reach, but we select on average deg nodes uniformly at random from the set of all nodes within the sensing radius. These experiments show that the FULL-SDP algorithm is somewhat sensitive to the sensing radius and the number of anchors used. As shown at the top of Figure 2.20, the recovery given by the FULL-SDP algorithm with 10 random anchors and no noise ($\eta = 0\%$) is rather poor compared to 2D-ASAP, ARAP and FAST-MVU, that are not using any anchor points whatsoever. However, the usage of long range distances significantly improves the solution of the FULL-SDP algorithm, as shown by the top right plot in Figure 2.20. Tables 2.7 and 2.8 provide the reconstruction errors for the C graphs. While 2D-ASAP and ARAP give comparable results, the errors of MVU are significantly larger.

For the PACM graphs in Figure 2.22, the sensor network takes the shape of the letters P, A, C, M that form a connected graph on $n = 378$ vertices. The sensing radius is $\rho = 0.9$ and the average degree $deg \approx 12$. This graph was particularly useful in testing the

η	deg	2D-ASAP	ARAP	MVU
0%	19	0.01	0.02	0.10
10%	19	0.05	0.04	0.09
20%	20	0.09	0.08	0.09
30%	20	0.18	0.16	0.15
40%	22	0.32	0.40	0.25
45%	22	0.41	0.53	0.32
50%	23	0.54	0.62	0.38

Table 2.5: Reconstruction errors (measured in ANE) for the US cities graph with $n = 1090$ cities and sensing radius $\rho = 0.032$.

Stage	Time (sec)
Break G into patches	41
Embedding patches	52
Build $\mathcal{Z}$	4.5
Compute $v_1^{\mathcal{Z}}$	0.3
Build $\mathcal{R}$	1.7
Compute $v_1^{\mathcal{R}}$	0.3
Step 3 (Least squares)	3.9
Total	103.6

Table 2.6: Running times of the 2D-ASAP algorithm. US map with $n = 1090$ cities, $\eta = 20\%$, $deg = 20$, $N = 1151$ patches, average patch size=15.6.

sensitivity of the algorithm to the topology of the network. In Table 2.9, we show the reconstruction errors for various levels of noise $\eta = 0\%, 10\%, 20\%, 30\%, 40\%$.

Another graph that we tested is the GRID graph shown in Figure 2.23. It has $n = 272$ nodes, sensing radius $\rho = 0.7$ and average degree $deg \approx 10$. Table 2.10 shows the reconstruction errors for various levels of noise $\eta = 0\%, 10\%, 20\%, 30\%, 40\%, 50\%$. To test the robustness to noise in the case of a very sparse graph, we experimented with the SQUARE graph with $n = 250$ nodes and average degree $deg = 6.8$. The 2D-ASAP and ARAP algorithm can handle well up to 40% noise, with the latter one being slightly more accurate. Figure 2.24 and Table 2.13 show the reconstruction errors for the SQUARE graph at noise levels $\eta = 0\%, 10\%, 20\%, 30\%, 40\%, 50\%$.

η	2D-ASAP	AAAP	ARAP	MVU	FULLSDP$_3$	FULLSDP$_{10}$	FULLSDP$^*_{10}$
0%	0	0.03	0.01	0.19	0.30	0.18	0
10%	0.02	0.15	0.03	0.20	0.29	0.17	0.10
20%	0.04	0.30	0.05	0.26	0.25	0.17	0.15
30%	0.07	0.37	0.08	0.29	0.25	0.17	0.23
40%	0.14	0.42	0.11	0.32	0.23	0.17	0.30
50%	0.23	0.46	0.20	0.34	0.23	0.18	0.37

Table 2.7: Reconstruction errors (measured in ANE) for the C graph with $n = 200$ nodes, sensing radius $\rho = 0.17$ and average degree $deg = 9$. FULLSDP$_k$ denotes the FULLSDP algorithm with k anchors. FULLSDP$^*_{10}$ denotes the same FULLSDP algorithm, but for a sensing radius $\rho = 1$ and a degree limit on each nodes that preserves the average degree deg.

The second to last graph that we tested is the SPIRAL graph shown in Figure 2.18(a). This graph is made of $n = 2259$ nodes, that are spread near a spiral curve that starts at the origin and once it gets to its outermost loop it traces back towards the origin. The perturba-

η	deg	2D-ASAP	ARAP	MVU	FULLSDP$_3$	FULLSDP$_{10}$	FULLSDP$^*_{10}$
0%	20	0	0.01	0.45	0	0	0
35%	22	0.15	0.18	0.61	0.25	0.24	0.28
40%	22	0.19	0.23	0.57	0.28	0.24	0.33
45%	23	0.23	0.28	0.72	0.28	0.24	0.37
50%	24	0.32	0.32	0.72	0.30	0.25	0.47
55%	25	0.40	0.38	0.82	0.34	0.28	0.49
60%	26	0.47	0.47	0.76	0.37	0.29	0.54
65%	27	0.58	0.56	0.90	0.42	0.32	0.58
70%	28	0.67	0.62	0.81	0.47	0.35	0.58

Table 2.8: Reconstruction errors (measured in ANE) for the C graph with $n = 200$ nodes and sensing radius $\rho = 0.28$.

η	2D-ASAP	AAAP	ARAP	MVU
0%	0	0	0	0.20
10%	0.06	0.11	0.02	0.22
20%	0.20	0.24	0.03	0.22
30%	0.23	0.30	0.06	0.23
40%	0.32	0.34	0.13	0.24

Table 2.9: Reconstruction errors (measured in ANE) for the PACM graph with $n = 425$ vertices, sensing radius $\rho = 0.9$ and average degree $deg = 12$.

tion of the sensors from the curve ensures that the 1-hop neighborhoods are not too close to being collinear. The sensing radius for this graph is $\rho = 0.47$. Despite the fact that the measured distances are noise-free, the localizations obtained by both 2D-ASAP, AAAP, ARAP and MVU (Figures 2.18(c), 2.18(e) and 2.18(f)) deviate from the true positioning. The failure of 2D-ASAP to find the original embedding in this noise-free case is due to a failure of the SMACOF procedure to localize a small number of patches. Although there is no noise in the distance measurements, the stress minimization algorithm sometimes converges to a local minimum, resulting in patches that are incorrectly localized. Since the topology of this graph is that of a closed curve, such bad patches lead to incorrect twists and turns in our computed embedding. Although 2D-ASAP and ARAP are using the same algorithm to localize the patches, it is clear that the incorrectly localized patches are less harmful to 2D-ASAP as they are to ARAP. This is also indicated in the averaged normalized errors: $ANE(\text{ASAP}) = 0.47$, $ANE(\text{ARAP}) = 1.30$, $ANE(\text{MVU}) = 3.43$. Figure 2.18(b) shows the accurate embedding obtained by 2D-ASAP when SNL-SDP was used to localize the patches, denoted 2D-ASAP-SDP, for which $ANE(\text{ASAP-SDP}) = 0.002$. Although SNL-SDP requires more time to embed the patches (850 seconds compared to 250 seconds for SMACOF), this time was well spent in getting an improved reconstruction.

Another difference between 2D-ASAP and ARAP for this SPIRAL graph is in the running time. While 2D-ASAP runs here for about 500 seconds (and 1100 seconds with SNL-SDP), ARAP takes over 10000 seconds, with the majority of its running time being

η	2D-ASAP	AAAP	ARAP	MVU	FULL-SDP$_3$
0%	0	0.01	0	0.02	0
10%	0.01	0.04	0.01	0.06	0.17
20%	0.02	0.12	0.01	0.12	0.23
30%	0.04	0.19	0.02	0.18	0.24
40%	0.06	0.25	0.04	0.26	0.26
50%	0.11	0.29	0.06	0.30	0.26

Table 2.10: Reconstruction errors (measured in ANE) for the GRID graph with $n = 272$ nodes, sensing radius $\rho = 0.7$, and average degree $deg = 10$.

spent on localizing the patches. We believe that this difference in running time is caused by ARAP's attempt to localize the entire 1-hop neighborhood of every node, rather than breaking up the 1-hop neighborhoods into globally rigid patches like 2D-ASAP does. The example of the SPIRAL graph shows that the embedding of some graphs can be challenging even in the noise-free case, but 2D-ASAP is doing relatively well even for this difficult graph.

Finally, in order to illustrate the scaling behavior of 2D-ASAP and compare its running time to that of the other algorithms, we experimented with random graphs with $n = \{10^3, 10^4, 10^5\}$ nodes distributed uniformly at random in the unit square. For this experiment, we used a machine with 192 GB memory and 2.66GHz core. Since the sizes of the graphs are increasing in scale, we choose the radius ρ such that the average degree remains about the same $deg = \{12.8, 12.5, 12.8\}$. Table 2.11 details the running times of the various steps of the 2D-ASAP algorithm for three graphs, and Table 2.12 compares the running times of 2D-ASAP, AAAP, ARAP, FAST-MVU and FULLSDP$_{20}$ for a random graph on $n = 10^3$ nodes. FAST-MVU is by far the fastest method with only 2.7 seconds, but also the one least robust to noise. 2D-ASAP comes second with 477 seconds, 95% of which are spent in the preprocessing steps. ARAP[9] took 1201 seconds, and when ran on $n = 10^4$ nodes, it did not produce an outcome within 48 hours. We could also move this earlier in this section. The FULL-SDP method takes over 5000 seconds when ran on a graph with $n = 10^3$ nodes, and is unlikely to compare well to 2D-ASAP if we increase n to 10^4. We expect that an optimized implementation in C would further reduce the running time of 2D-ASAP, in particular the steps of breaking G into patches and computing the patch intersections. For $n = 10^5$, these two steps account for almost 70% of the current running time, and an implementation where they would scale linearly means reducing the overall running time by more than 60% to only about 60000 seconds.

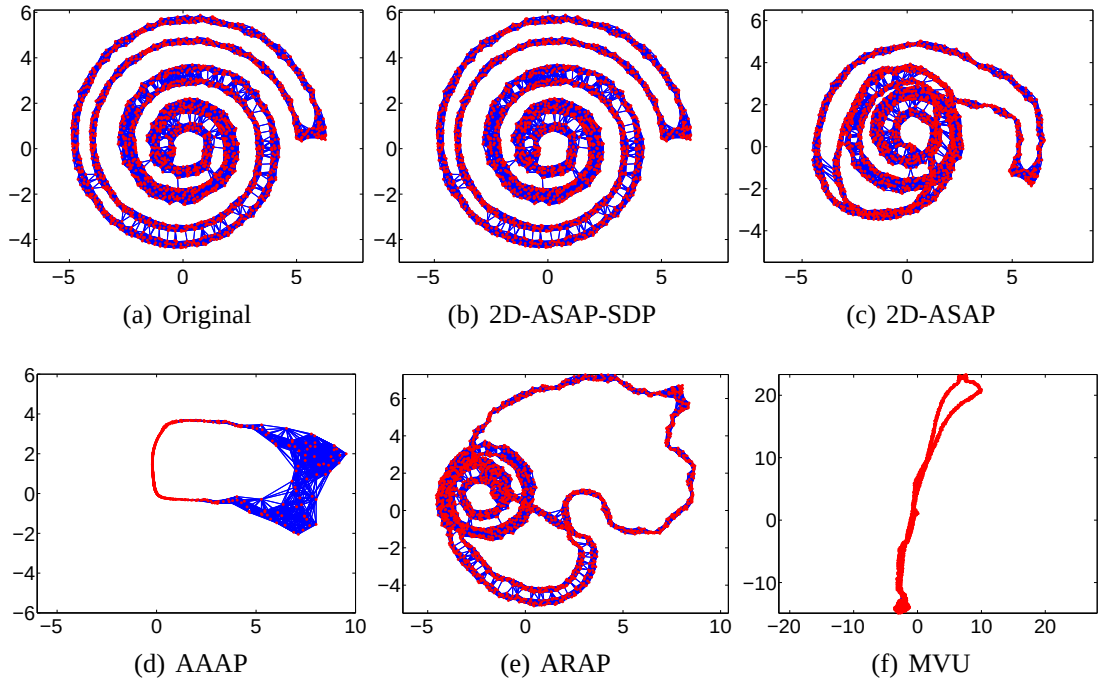
[9]Note that for testing ARAP, we used the MATLAB implementation kindly provided by its authors on November 2009.

Stage \# of nodes n	1000	10000	100000
Break G into patches	41	901	52,180
Embedding patches	414	4,325	37,140
Patch intersections	2	132	58,134
Build $\mathcal{Z}$	8.7	90	2,237
Compute $v_1^{\mathcal{Z}}$	0.8	13	926
Build $\mathcal{R}$	4.6	49	3,414
Compute $v_1^{\mathcal{R}}$	0.2	7	522
Step 3	6	88	4,772
Total Time (sec)	477	5,605	159,325

Table 2.11: Running times (in seconds) of the 2D-ASAP algorithm on the SQUARE graph with $n = \{10^3, 10^4, 10^5\}$ nodes inside the unit square, $\eta = 0\%$ and $deg \approx 12, 13$.

Algorithm	$n = 1000$	$n = 10000$
2D-ASAP	477	5605
AAAP	1170	> 48 hours
ARAP	1201	> 48 hours
FAST-MVU	2.7	10.8
FULLSDP$_{20}$	5250	-

Table 2.12: Comparison of the running times (in seconds) of different algorithms, for the SQUARE graph with $n = \{10^3, 10^4\}$ nodes and $\eta = 0\%$.



(a) Original (b) 2D-ASAP-SDP (c) 2D-ASAP

(d) AAAP (e) ARAP (f) MVU

Figure 2.18: Reconstructions of the SPIRAL graph with $n = 2259$ nodes, $\rho = 0.47$ and $\eta = 0\%$. 2D-ASAP-SDP is a version of 2D-ASAP where we used SDP for the localization of the patches, instead of SMACOF.

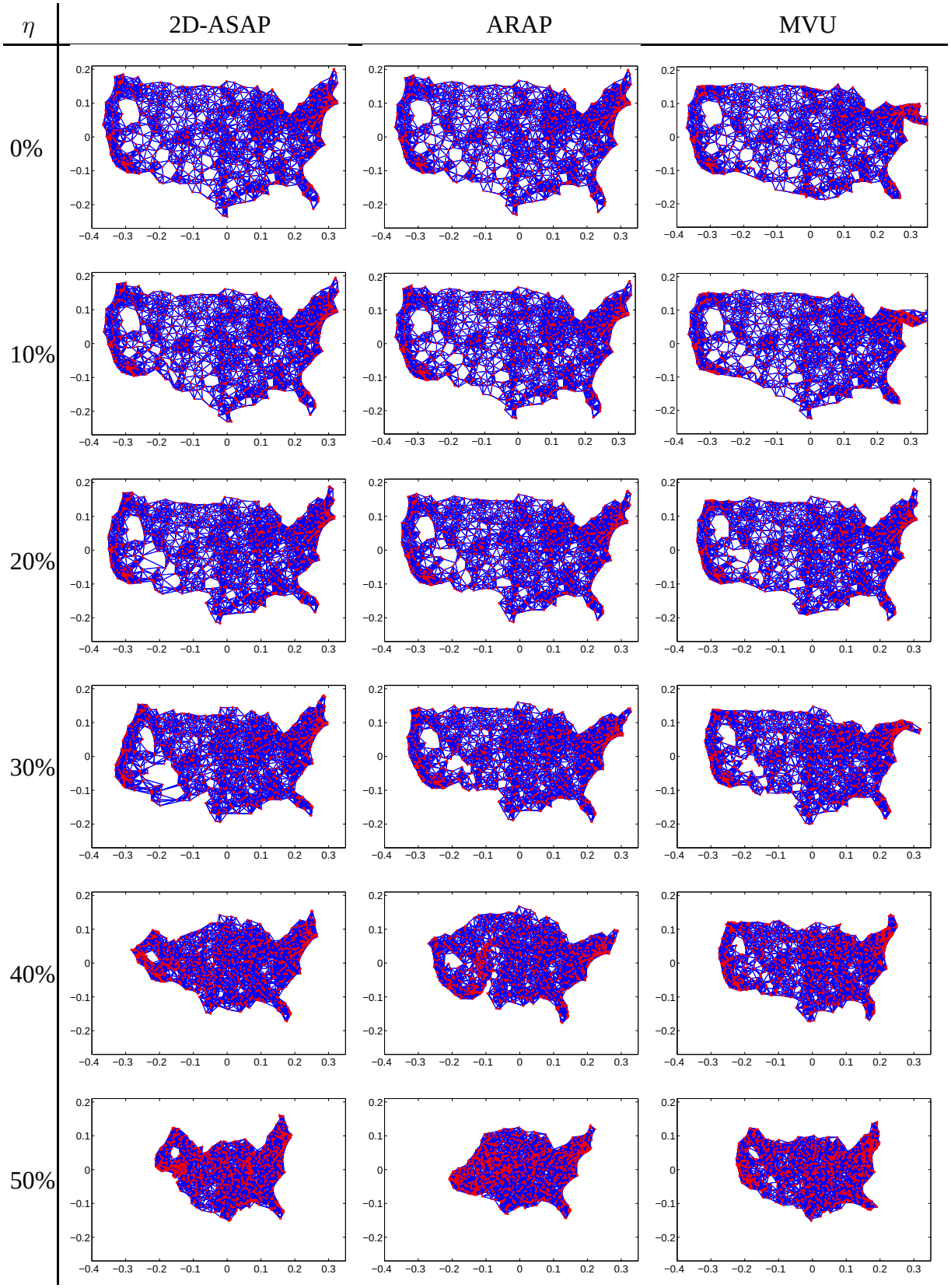


Figure 2.19: Reconstructions of the US cities graph with $n = 1090$ nodes, sensing radius $\rho = 0.32$ and $\eta = 0\%, 10\%, 20\%, 30\%, 40\%, 50\%$.

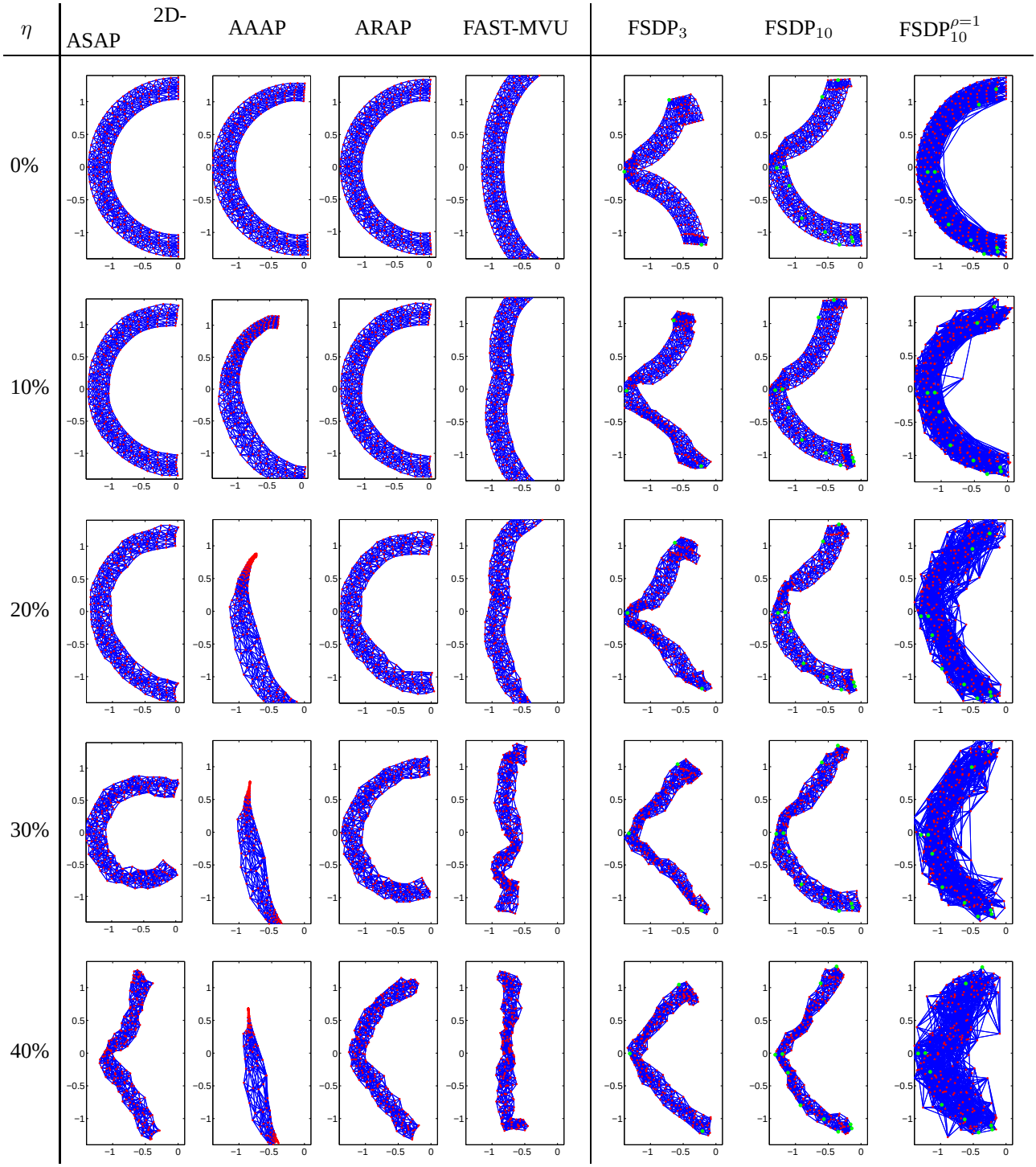


Figure 2.20: Reconstructions of the sparse C graph with $n = 200$ nodes, $\rho = 0.17$, and $\eta = 0\%, 10\%, 20\%, 30\%, 40\%$.

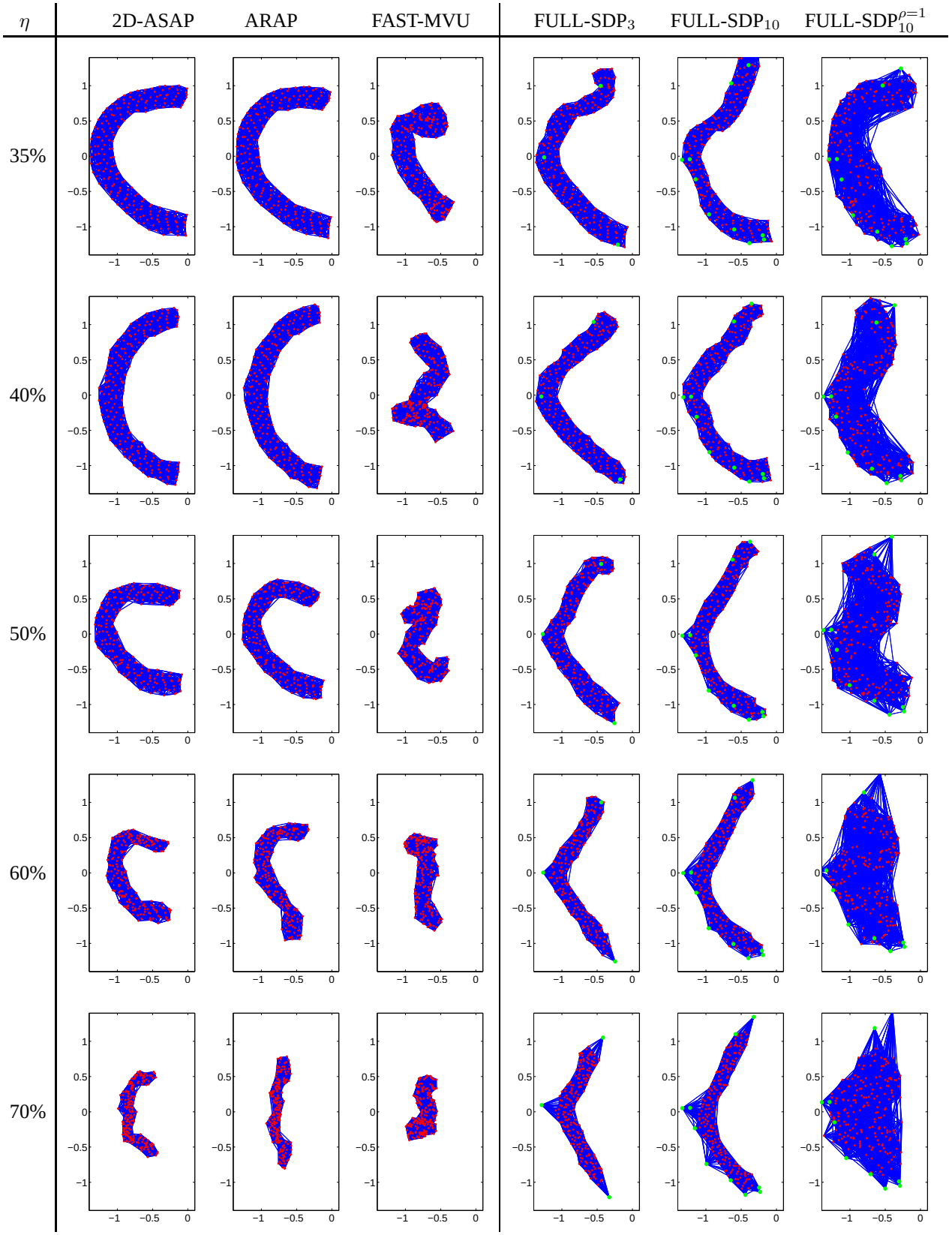


Figure 2.21: Reconstructions of the dense C graph with $n = 200$ nodes, $\rho = 0.28$, and $\eta = 35\%, 40\%, 50\%, 60\%, 70\%$.

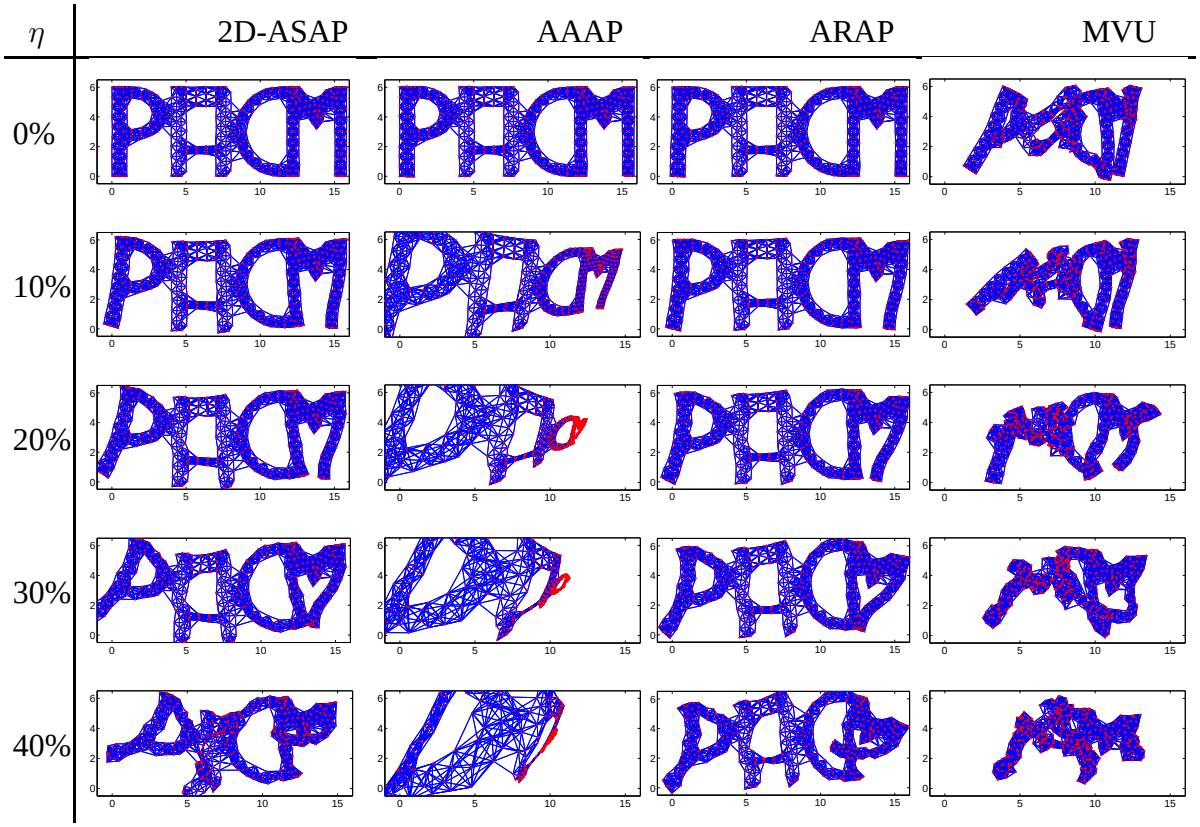


Figure 2.22: Reconstructions of the PACM graph with $n = 425$ nodes, $\rho = 0.9$, and noise levels $\eta = 0\%, 10\%, 20\%, 30\%, 40\%$.

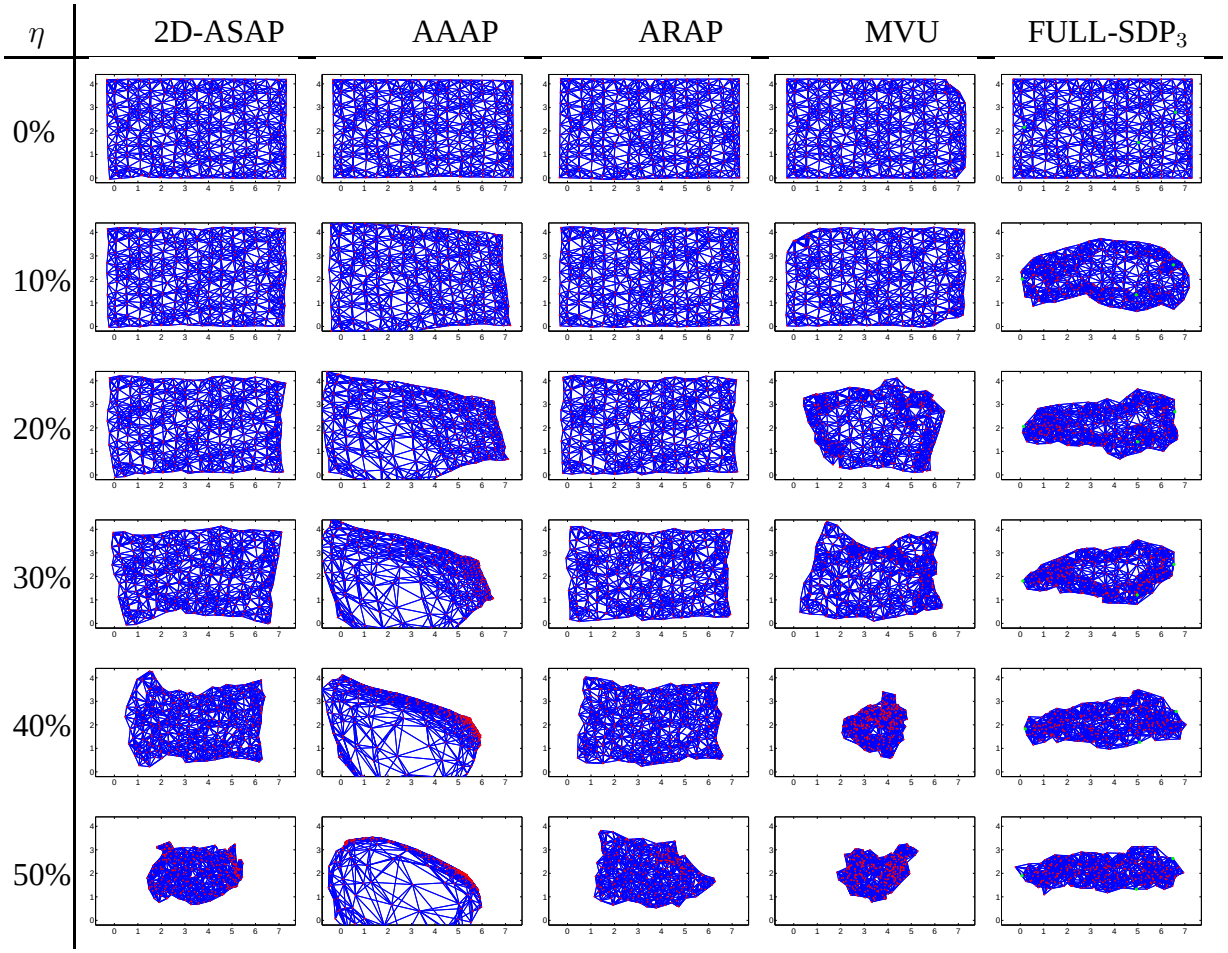


Figure 2.23: Reconstructions of the GRID graph with $n = 272$ nodes, $\rho = 0.7$, and noise levels $\eta = 0\%, 10\%, 20\%, 30\%, 40\%, 50\%$.

η	2D-ASAP	AAAP	ARAP	MVU	FULLSDP$_3$	FULLSDP$_{10}$
0%	0.0000	0.0036	0.0060	0.0273	0.2329	0.0652
10%	0.0155	0.0443	0.0092	0.0619	0.2917	0.1689
20%	0.0524	0.1256	0.0165	0.1004	0.2858	0.1760
30%	0.0835	0.2023	0.0277	0.1646	0.2767	0.1793
40%	0.1543	0.2551	0.1114	0.2114	0.2588	0.1722
50%	0.3150	0.2899	0.3132	0.2540	0.2558	0.1714

Table 2.13: Reconstruction errors (measured in ANE) for the SQUARE graph with $n = 250$ nodes, sensing radius $\rho = 1.51$. and average degree $deg = 6.8$.

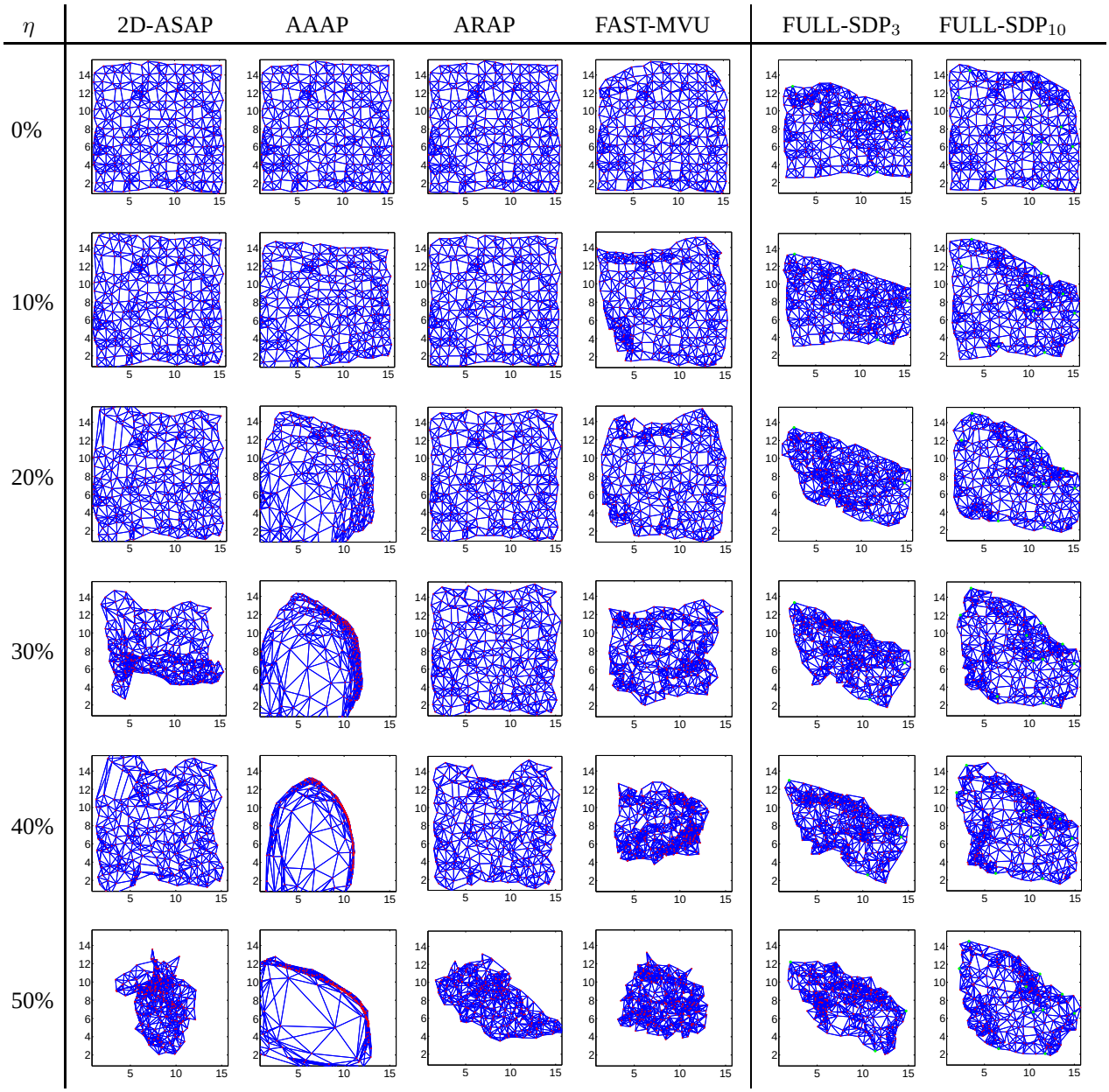


Figure 2.24: Reconstructions of the SQUARE graph with $n = 250$ nodes, sensing radius $\rho = 1.51$, and average degree $deg = 6.8$, for different levels of noise $\eta = 0\%, 10\%, 20\%, 30\%, 40\%, 50\%$.

2.8 Summary and discussion

In this chapter, we introduced 2D-As-Synchronized-As-Possible (2D-ASAP), a non-incremental non-iterative anchor-free algorithm for localizing sensor networks. As our numerical experiments show, 2D-ASAP is extremely robust to high levels of noise in the measured distances and to sparse connectivity of the measurement graph. Our algorithm starts with local coordinate computations based only on 1-hop neighborhood information, but unlike existing incremental methods, it synchronizes all such local information in a noise robust global optimization process using an efficient eigenvector computation.

Across all graphs that we have tested, 2D-ASAP and ARAP almost always give the best results in terms of the averaged normalized error. In particular, whenever another algorithm like MVU or SDP does well, 2D-ASAP and ARAP are also successful. Also to the naked eye, the localization results of 2D-ASAP and ARAP best resemble the true positioning. The SPIRAL graph demonstrates that there are cases for which 2D-ASAP and ARAP can give significantly different results, in this case, to the favor of the former. When comparing 2D-ASAP and AAAP, both of which are fully ab-initio reconstructions, it is clear that 2D-ASAP gives much better results. Although 2D-ASAP and ARAP usually give similar results, there is a fundamental difference between the two algorithms. While ARAP requires an initial guess provided by AAAP to start its iterative refinement process, 2D-ASAP is a fully ab-initio reconstruction method that scales almost linearly in the size of the network, as demonstrated by our example of localizing a network with 100,000 nodes. In practice, 2D-ASAP can (and perhaps should) be followed by a refinement procedure, such as stress minimization or even ARAP.

The unit disc graph assumption, which comes up naturally in many problems of practical interest, is essential to the performance of the 2D-ASAP algorithm as it favors the existence of many globally rigid patches of relatively large size. When the disc graph assumption does not hold, the 1-hop neighborhood of a node may be extremely sparse, and thus breaking up such a sparse star graph leads to many small maximally globally rigid components (i.e., most of them may contain only a few nodes), with only a few of them having a large enough pairwise intersection. Since small patches lead to small patch intersections, it would therefore be difficult for 2D-ASAP to align patches correctly and compute a robust final solution. Except for the combinatorial score method in Section 2.5, the 2D-ASAP algorithm does not (explicitly) make use of the sensing radius of the disc graph model assumption.

Combining both distance and angular measurements to increase accuracy and robustness to noise is another possible generalization. The inclusion of angular measurements in localization algorithms has not been studied as thoroughly as distance measurements, but recent papers, such as [21], provide insight for this potential improvement. In the context of the 2D-ASAP algorithm, angular measurements should lead to better localization and alignment of the patches.

There are a few possible ways in which the 2D-ASAP algorithm can be improved. First, it is possible to weigh the entries of the Z and R matrices from Steps 1 and 2. One possible weighing scheme would be to assign weights that are proportional to some confidence measure in the alignment of the patches. For example, a high residual obtained when aligning two patches hints that the relative reflection or rotation may be incorrect. Also, aligning

two patches with a large overlap is significantly more robust than aligning two patches that overlap in just a few vertices. Another possible scheme is to design weights that will minimize the mixing time of the random walk on the patch graph, or equivalently, maximizing the second eigenvalue (the spectral gap) of the graph Laplacian. This approach is motivated by our unreported matrix perturbation analysis initiated in Section 4.2 and by the intuition that faster mixing of the random walk should assist the eigenvector method to integrate and propagate consistency relations over cycles in the patch graph. In [99] it is shown that this fastest mixing design problem is convex and can be solved using SDP, which interestingly enough draws similarities with the MVU approach to localization. The synchronization problems in Steps 1 and 2 can be solved using SDP instead of the eigenvector method (see [92]), but our experience show that the improvement is usually marginal.

Targeting bad patches and bad distance measurements (outliers detection) is another approach that may increase the robustness of the algorithm to noise. After Step 1 of the algorithm, we have the estimated reflections $\hat{z}_i$, and it is possible now to target bad patches P_i for which $\hat{z}_i\hat{z}_j^{-1}$ differs from z_{ij} for a large number of neighboring patches P_j. Even before Step 1, one may wish to discard patches whose area inside their convex hull is too small, as such patches are likely to result in incorrect reflections. Also, in the spirit of the weighted-matrices scenario mentioned above, one may start by aligning patches and localizing nodes only in regions of the graph that have a higher density of edges. Targeting and removing bad patches that have low confidence may disconnect the patch graph in Steps 1 and 2. One should then run the 2D-ASAP algorithm on each connected component of the patch graph, thus localizing all the nodes within them. Some of the missing distances can now be inferred, and can be added to the problem by an iterative application of the 2D-ASAP algorithm. Although this method seems incremental in nature, at each step the eigenvector computation will enforce global consistencies.

We believe that the eigenvector synchronization method has the potential of being useful in many applications other than the localization problem of sensor networks. In particular, in [93, 53] we demonstrated its usefulness in cryo-electron microscopy [40], and showed its mathematical connection to the parallel transport and the connection-Laplacian operators from differential geometry. We are currently extending this approach to the 3D structure from motion problem in computer vision and to the analysis of high dimensional data point clouds [94], specifically, to the generalization of Laplacian eigenmaps and diffusion maps [11, 25] that are popular methods for dimensionality reduction and spectral clustering.

Although this chapter pertains to the the localization of planar networks, we remark that 2D-ASAP can be generalized to solve the localization problem in $\mathbb{R}^3$, a problem which is motivated by three-dimensional structure determination of macromolecules using NMR spectroscopy [56]. The three-dimensional localization problem can be formulated as a synchronization problem over Euc(3), and can be similarly solved in three steps: an eigenvector synchronization for the reflections over $\mathbb{Z}_2$, an eigenvector synchronization for the rotations over SO(3), and a least squares solution for the translations in $\mathbb{R}^3$. In the second step of the algorithm, the optimal rotations between pairs of patches will be represented by 3×3 rotation matrices, and the elements of SO(3) will be obtained from the top three eigenvectors instead of just the top one. In the next chapter, we extend on the approach used in 2D-ASAP to accommodate for the additional challenges posed by rigidity theory

in $\mathbb{R}^3$ and other constraints that are specific to the molecule problem, the main application for the three dimensional version of the graph realization problem.

Chapter 3

Graph realization in $\mathbb{R}^3$ and the molecule problem

3.1 Introduction

This chapter pertains to the graph realization problem in $\mathbb{R}^3$ and its application to a problem from structural biology. We use the same notation as in the two dimensional case. One is given a graph $G = (V, E)$ consisting of a set of $|V| = n$ nodes and $|E| = m$ edges, together with a non-negative distance measurement d_{ij} associated with each edge, and is asked to compute a realization of G in $\mathbb{R}^3$. In other words, for any pair of adjacent nodes i and j, $(i, j) \in E$, the distance $d_{ij} = d_{ji}$ is available, and the goal is to find a three dimensional embedding $p_1, p_2, \ldots, p_n \in \mathbb{R}^3$ such that $\|p_i - p_j\| = d_{ij},$ for all $(i, j) \in E$. We follow the same graph model as in the two dimensional graph realization problem, i.e., the geometric graph model, where the distance between two nodes is available if and only if they are within sensing radius ρ of each other, i.e., $(i, j) \in E \iff d_{ij} \leq \rho$. Figure 3.1 shows an example of a measurement graph for a data set of $n = 500$ nodes, with sensing radius $\rho = 0.092$ and average degree $deg = 18$, i.e. each node knows, on average, the distance to its 18 closest neighbors.

The three dimensional graph realization problem arises naturally in the application of nuclear magnetic resonance (NMR) to structural biology, and is often referred to as the molecular distance geometry problem, or simply the *molecule problem*. A brief description of this application is given in Section 3.2. In Chapter 2 we introduced 2D-ASAP, an eigenvector based synchronization algorithm that solves the sensor network localization problem in $\mathbb{R}^2$. We summarize below the approach used in 2D-ASAP, and further explain the differences and improvements that its generalization to three-dimensions brings. Figure 3.2 shows a schematic overview of our algorithm, which we call 3D-As-Synchronized-As-Possible (3D-ASAP). In this chapter, we focus exclusively on the molecule problem.

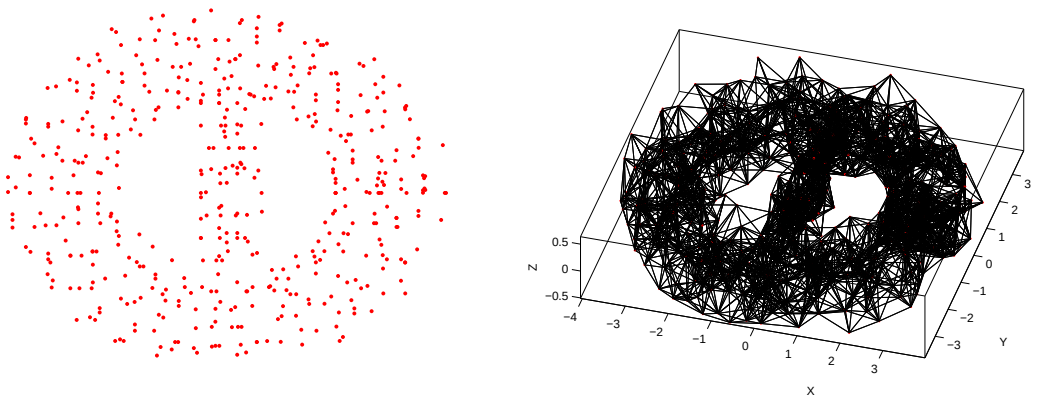


Figure 3.1: 2D view of the BRIDGE-DONUT data set, a cloud of $n = 500$ points in $\mathbb{R}^3$ in the shape of a donut (left), and the associated measurement geometric graph of radius $\rho = 0.92$ and average degree $deg = 18$ (right).

The 2D-ASAP algorithm proposed in [32] and described in Chapter 2 belongs to the group of algorithms that integrate local distance information into a global structure determination. For every sensor, we first identify globally rigid subgraphs of its 1-hop neighborhood that we call patches. Each patch is then separately localized in a coordinate system of its own using either the stress minimization approach of [46], or by using semidefinite programming (SDP). In the noise-free case, the computed coordinates of the sensors in each patch must agree with their global positioning up to some unknown rigid motion, that is, up to translation, rotation and possibly reflection. To every patch there corresponds an element of the Euclidean group Euc(2) of rigid transformations in the plane, and the goal is to estimate the group elements that will properly align all the patches in a globally consistent way. By finding the optimal alignment of all pairs of patches whose intersection is large enough, we obtain measurements for the ratios of the unknown group elements. Finding group elements from noisy measurements of their ratios is also known as the synchronization problem [66, 41]. For example, the synchronization of clocks in a distributed network from noisy measurements of their time offsets is a particular example of synchronization over $\mathbb{R}$. [92] introduced an eigenvector method for solving the synchronization problem over the group SO(2) of planar rotations. This algorithm serves as the basic building block for our 2D-ASAP and 3D-ASAP algorithms. Namely, we reduce the graph realization problem to three consecutive synchronization problems that overall solve the synchronization problem over Euc(2). Intuitively, we use the eigenvector method for the compact part of the group (reflections and rotations), and use the least-squares method for the non-compact part (translations). In the first step, we solve a synchronization problem over $\mathbb{Z}_2$ for the possible reflections of the patches using the eigenvector method. In the second step, we solve a synchronization problem over SO(2) for the rotations, also using the eigenvector method. Finally, in the third step, we solve a synchronization problem over $\mathbb{R}^2$ for the translations by solving an overdetermined linear system of equations using the method of least squares. This solution yields the estimated coordinates of all the sensors up to a global rigid transformation. Note that the groups $\mathbb{Z}_2$ and SO(2) are compact, allowing us

to use the eigenvector method, while the group $\mathbb{R}^2$ is non-compact and requires a different synchronization method such as least squares.

In this chapter, we extend on the approach used in 2D-ASAP to accommodate for the additional challenges posed by rigidity theory in $\mathbb{R}^3$ and other constraints that are specific to the molecule problem. In addition, we also increase the robustness to noise and speed of the algorithm. The following paragraphs are a brief summary of the improvements 3D-ASAP brings, in the order in which they appear in the algorithm.

First, we address the issue of using a divide and conquer approach from the perspective of three dimensional global rigidity, i.e., of decomposing the initial measurement graph into many small overlapping patches that can be uniquely localized. Sufficient and necessary conditions for two dimensional combinatorial global rigidity have been established only recently, and motivated our approach for building patches in 2D-ASAP [60, 55]. Due to the recent coning result in rigidity theory [28], it is also possible to extract globally rigid patches in dimension three. However, such globally rigid patches cannot always be localized accurately by SDP algorithms, even in the case of noiseless data. To that end, we rely on the notion of *unique localizability* [96] to localize noiseless graphs, and introduce the notion of a weakly uniquely localizable (WUL) graph, in the case of noisy data.

Second, we use a median-based denoising algorithm in the preprocessing step, that overall produces more accurate patch localizations. Our approach is based on the observation that a given edge may belong to several different patches, the localization of each of which may result in a different estimation for the distance. The median of these different estimators from the different patches is a more accurate estimator of the underlying distance.

Third, we incorporated in 3D-ASAP the possibility to integrate prior available information. As it is often the case in real applications (such as NMR), one has readily available structural information on various parts of the network that we are trying to localize. For example, in the NMR application, there are often subsets of atoms (referred to as "molecular fragments", by analogy with the fragment molecular orbital approach, e.g., [39]) whose relative coordinates are known a priori, and thus it is desirable to be able to incorporate such information in the reconstruction process. Of course, one may always input into the problem all pairwise distances within the molecular fragments. However, this requires increased computational efforts while still not taking full advantage of the available information, i.e., the orientation of the molecular fragment. Nodes that are aware of their location are often referred to as anchors, and anchor-based algorithms make use of their existence when computing the coordinates of the remaining sensors. Since in some applications the presence of anchors is not a realistic assumption, it is important to have efficient and noise-robust anchor-free algorithms, that can also incorporate the location of anchors if provided. However, note that having molecular fragments is not the same as having anchors; given a set of (possibly overlapping) molecular fragments, no two of which can be joined in a globally rigid body, only one molecular fragment can be treated as anchor information (the nodes of that molecular fragment will be the anchors), as we do not know a priori how the individual molecular fragments relate to each other in the same global coordinate system.

Fourth, we allow for the possibility of combining the first two steps (computing the reflections and rotations) into one single step, thus doing synchronization over the group of orthogonal transformations $\mathrm{O}(3) = \mathbb{Z}_2 \times \mathrm{SO}(3)$ rather than individually over $\mathbb{Z}_2$ fol-

lowed by SO(3). However, depending on the problem being considered and the type of available information, one may choose not to combine the above two steps. For example, when molecular fragments are present, we first do synchronization over $\mathbb{Z}_2$ with anchors, followed by synchronization over SO(3). We defer to Section 4.4 of the next chapter the detailed analysis of the synchronization problem over $\mathbb{Z}_2$ with anchor information.

Fifth, we incorporate another median-based heuristic in the final step, where we compute the translations of each patch by solving, using least squares, three overdetermined linear systems, one for each of the x, y and z axis. For a given axis, the displacement between a pair of nodes appears in multiple patches, each resulting in a different estimation of the displacement along that axis. The median of all these different estimators from different patches provides a more accurate estimator for the displacement. In addition, after the least squares step, we introduce a simple heuristic that corrects the scaling of the noisy distance measurements. Due to the geometric graph model assumption and the uniform noise model, the distance measurements taken as input by 3D-ASAP are significantly scaled down, and the least squares step further shrinks the distances between nodes in the initial reconstruction.

Finally, we introduce 3D-SP-ASAP, a variant of 3D-ASAP which uses a spectral partitioning algorithm in the pre-processing step of building the patches. This approach is somewhat similar to the recently proposed DISCO algorithm of [74]. The philosophy behind DISCO is to recursively divide large problems into two smaller problems, thus building a binary tree of subproblems, which can ultimately be solved by the traditional SDP-based localization methods. 3D-ASAP has the disadvantage of generating a number of smaller subproblems (patches) that is linear in the size of the network, and localizing all resulting patches is a computationally expensive task, which is exactly the issue addressed by 3D-SP-ASAP.

From a computational point of view, all steps of the algorithm can be implemented in a distributed fashion and scale linearly in the size of the network, except for the eigenvector computation, which is nearly-linear[1]. We show the results of numerous numerical experiments that demonstrate the robustness of our algorithm to noise and various topologies of the measurement graph.

This chapter is organized as follows:

1. Section 3.2 describes an application of nuclear magnetic resonance (NMR) to structural biology, and motivates the use of our localization algorithm for the molecule problem.

2. Section 3.3 is a brief survey of related approaches for solving the graph realization problem in $\mathbb{R}^3$, although some of the methods described in the previous chapter in Section 2.2 also apply to the three dimensional case.

3. Section 3.4 gives an overview of the 3D-ASAP algorithm we propose.

[1]Every iteration of the power method or the Lanczos algorithm that are used to compute the top eigenvectors is linear in the number of edges of the graph, but the number of iterations is greater than $O(1)$ as it depends on the spectral gap.

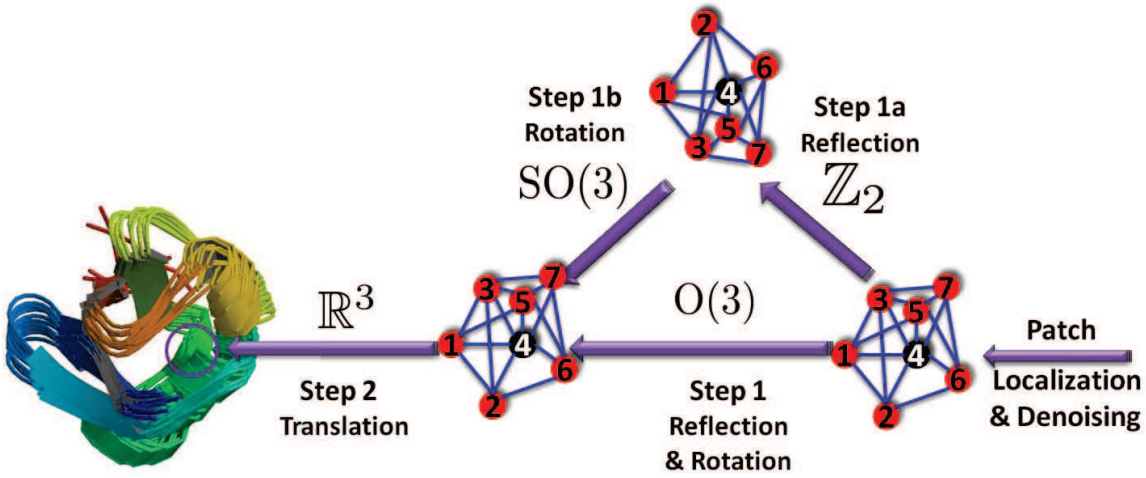


Figure 3.2: The 3D-ASAP recovery process for a patch in the 1d3z molecule graph. The localization of the rightmost subgraph in its own local frame is obtained from the available accurate bond lengths and noisy NOE measurements by using one of the SDP algorithms. To every patch, like the one shown here, there corresponds an element of Euc(3) that we try to estimate. Using the pairwise alignments, in Step 1 we estimate both the reflection and rotation matrix from an eigenvector synchronization computation over O(3), while in Step 2 we find the estimated coordinates by solving an overdetermined system of linear equations. If there is available information on the reflection or rotations of some patches, one may choose to further divide Step 1 into two consecutive steps. Step 1a is synchronization over $\mathbb{Z}_2$, while Step 1b is synchronization over SO(3), in which the missing reflections and rotations are estimated.

4. Section 3.5 introduces the notion of weakly uniquely localizable graphs used for breaking up the initial large network into patches, and explains the process of embedding and aligning the patches.

5. Section 3.6 proposes a variant of the 3D-ASAP algorithm by using a spectral clustering algorithm as a preprocessing step in breaking up the measurement graph into patches.

6. In Section 3.7 we introduce a novel median-based denoising technique that improves the localization of individual patches, as well as a heuristic that corrects the shrinkage of the distance measurements.

7. In Section 3.8, we detail the results of numerical simulations in which we tested the performance of our algorithms in comparison to existing state-of-the-art algorithms.

8. Finally, Section 3.9 is a summary and a discussion of possible extensions of the algorithm and its usefulness in other applications.

3.2 NMR spectroscopy and the molecule problem

The graph realization problem in $\mathbb{R}^3$ is of particular importance because it arises naturally in the application of nuclear magnetic resonance (NMR) to structural biology. NMR spectroscopy is a well established modality for atomic structure determination, especially for relatively small proteins (i.e., with atomic mass less than 40kDa) [110], and contributes to progress in structural genomics [1]. General properties of proteins such as bond lengths and angles can be translated into accurate distance constraints. In addition, peaks in the NOESY experiments are used to infer spatial proximity information between pairs of nearby hydrogen atoms, typically in the range of 1.8 to 6 angstroms. The intensity of such NOESY peaks is approximately proportional to the distance to the minus 6^{th} power, and it is thus used to infer distance information between pairs of hydrogen atoms (NOEs). Unfortunately, NOEs provide only a rough estimate of the true distance, and hence the need for robust algorithms that are able to provide accurate reconstructions even at high levels of noise in the NOE data. In addition, the experimental data often contains potential constraints that are ambiguous, because of signal overlap resulting in incomplete assignment [82]. The structure calculation based on the entire set of distance constraints, both accurate measurements and NOE measurements, can be thought of as instance of the graph realization problem in $\mathbb{R}^3$ with noisy data.

It complements crystallography by providing information in solution, free of crystal packing effects, and is a rich source of information about local molecular dynamics [80] and electronic properties. In contrast to crystallography, the process of structure determination is indirect and complex involving multiple cycles of integration of experimental constraints (predominantly from the nuclear Overhauser effect NOE) and preexisting chemical knowledge of primary structure and typical molecular geometry at the atomic and fragment level. Initial efforts on using the NOEs focused on using the methods of distance geometry [54], and later developments elaborated this using representations in torsion angle space e.g [52] and integration with full molecular energy and dynamics calculations, e.g. [88]. While these methods have provided powerful tools for NMR structure determination, they are relatively slow, and do not provide for direct integration with experimental data, because of the normally required intermediate step of assignment, i.e. the linkage map of what spectral signal (principally NOEs) belongs to which hydrogen or set of hydrogens. Various ingenious ways to circumvent this tedious and error prone step have been proposed (reviewed in [51]) involving back calculation from preliminary structures, incorporation of probabilities of typing and sequence for assignment, and direct calculation of hydrogen density from complete NOE sets [50]. It is generally recognized that whatever the robustness of the algorithmic approach, an "assignment-free" procedure is not currently practical because of imprecision of measurements of NOEs, lack of completeness of NOE data sets, and ambiguities arising from spectral overlap. However, if the steps of incorporation of preexisting chemical information, and calculation of preliminary structure were considerably faster, then it is quite likely that assignment-free procedures could be developed and could be robust, using statistical correlation from preexisting data, Monte Carlo calculations and other approaches. In addition to the specific NMR approach, where the scalar nature of the NOE data suggests an obvious application, basic speed up of preliminary structure generation would have value in homology modeling [85], and molecular replacement in crystallogra-

phy [111]. The value of a new ultrafast method of generating preliminary structures from a variety of experimental data using chemical structure constraints is likely to have very broad applications.

3.3 Related work

Due to the importance of the graph realization problem, many heuristic strategies and numerical algorithms have been proposed in the last decade. A popular approach to solving the graph realization problem is based on SDP, and has attracted considerable attention in recent years [17, 12, 13, 14, 114]. We defer to Section 3.5.4 a description of existing SDP relaxations of the graph realization problem. Such SDP techniques are usually able to localize accurately small to medium sized problems (up to a couple thousands atoms). However, many protein molecules have more than 10,000 atoms and the SDP approach by itself is no longer computationally feasible due to its increased running time. In addition, the performance of the SDP methods is significantly affected by the number and location of the anchors, and the amount of noise in the data. Note that many of the algorithms described in the previous chapter in Section 2.2 can also be generalized to the three dimensional case. To overcome the computational challenges posed by the limitations of the SDP solvers, several divide and conquer approaches have been proposed recently for the graph realization problem. One of the earlier methods appears in [16], and more recent methods include the Distributed Anchor Free Graph Localization (DAFGL) algorithm of [15], and the DISCO algorithm (DIStributed COnformation) of [74].

One of the critical assumptions required by the distributed SDP algorithm in [16] is that there exist anchor nodes distributed uniformly throughout the physical space. The algorithm relies on the anchor nodes to divide the sensors into clusters, and solves each cluster separately via an SDP relaxation. Combining smaller subproblems together can be a challenging task, however this is not an issue if there exist anchors within each smaller subproblem (as it happens in the sensor network localization problem) because the solution to the clusters induces a global positioning; in other words the alignment process is trivially solved by the existence of anchors within the smaller subproblems. Unfortunately, for the molecule problem, anchor information is scarce, almost inexistent, hence it becomes crucial to develop algorithms that are amenable to a distributed implementation (to allow for solving large scale problems) despite there being no anchor information available. The DAFGL algorithm of [15] attempted to overcome this difficulty and was successfully applied to molecular conformations, where anchors are inexistent. However, the performance of DAFGL was significantly affected by the sparsity of the measurement graph, and the algorithm could tolerate only up to 5% multiplicative noise in the distance measurements.

The recent DISCO algorithm of [74] addressed some of the shortcomings of DAFGL, and used a similar divide-and-conquer approach to successfully reconstruct conformations of very large molecules. At each step, DISCO checks whether the current subproblem is small enough to be solved by itself, and if so, solves it via SDP and further improves the reconstruction by gradient descent. Otherwise, the current subproblem (subgraph) is further divided into two subgraphs, each of which is then solved recursively. To combine two

subgraphs into one larger subgraph, DISCO aligns the two overlapping smaller subgraphs, and refines the coordinates by applying gradient descent. In general, a divide-and-conquer algorithm consists of two ingredients: dividing a bigger problem into smaller subproblems, and combining the solutions of the smaller subproblems into a solution for a larger subproblem. With respect to the former aspect, DISCO minimizes the number of edges between the two subgroups (since such edges are not taken into account when localizing the two smaller subgroups), while maximizing the number of edges within subgroups, since denser graphs are easier to localize both in terms of speed and robustness to noise. As for the latter aspect, DISCO divides a group of atoms in such a way that the two resulting subgroups have many overlapping atoms. Whenever the common subgroup of atoms is accurately localized, the two subgroups can be further joined together in a robust manner. DISCO employs several heuristics that determine when the overlapping atoms are accurately localized, and whether there are atoms that cannot be localized in a given instance (they do not attach to a given subgraph in a globally rigid way). Furthermore, in terms of robustness to noise, DISCO compared favorably to the above-mentioned divide-and-conquer algorithms.

3.4 The 3D-ASAP Algorithm

3D-ASAP is a divide and conquer algorithm that breaks up the large graph into many smaller overlapping subgraphs, that we call patches, and "stitches" them together consistently in a global coordinate system with the purpose of localizing the entire measurement graph. Unlike previous graph localization algorithms, we build patches that are globally rigid[2] (or weakly uniquely localizable, that we define later), in order to avoid foldovers in the final solution[3]. We also assume that the given measurement graph is globally rigid to begin with; otherwise the algorithm will discard the parts of the graph that do not attach globally rigid to the rest of the graph.

We build the patches in the following way. For every node i we denote by $V(i) = \{j : (i,j) \in E\} \cup \{i\}$ the set of its neighbors together with the node itself, and by $G(i) = (V(i), E(i))$ its subgraph of 1-hop neighbors. If $G(i)$ is globally rigid, which can be checked efficiently using the randomized algorithm of [45], then we embed it in $\mathbb{R}^3$. Using SDP for globally rigid (sub)graphs can still produce incorrect localizations, even for noiseless data. In order to ensure that SDP would give the correct localization, a stronger notion of rigidity is needed, that of unique localizability [96]. However, in practical applications the distance measurements are noisy, so we introduce the notion of weakly localizable subgraphs, and use it to build patches that can be accurately localized. The exact way we break up the 1-hop neighborhood subgraphs into smaller globally rigid or weakly uniquely localizable subgraphs is detailed in Section 3.5.2. In Section 3.6 we describe an alternative

[2]There are several different notions of rigidity that appear in the literature, such as local and global rigidity, and the more recent notions of universal rigidity and unique localizability [114, 96].

[3]We remark that in the geometric graph model, the non-edges also provide distance information since $(i,j) \notin E$ implies $d_{ij} > \rho$. This information sometimes allows to uniquely localize networks that are not globally rigid to begin with. However, we do not use this information in the standard formulation of our algorithm, but this could be further incorporated to enhance the reconstruction of very sparse networks.

method for decomposing the measurement graph into patches, using a spectral partitioning algorithm. We denote by N the number of patches obtained in the above decomposition of the measurement graph, and note that it may be different from n, the number of nodes in G, since the neighborhood graph of a node may contribute several patches or none. Also, note that the embedding of every patch in $\mathbb{R}^3$ is given in its own local frame. To compute such an embedding, we use the following SDP-based algorithms: FULL-SDP for noiseless data [17], and SNL-SDP for noisy data [101]. Once each patch is embedded in its own coordinate system, one must find the reflections, rotations and translations that will stitch all patches together in a consistent manner, a process to which we refer as *synchronization*.

We denote the resulting patches by $P_1, P_2, \ldots, P_N$. To every patch P_i there corresponds an element $e_i \in$ Euc(3), where Euc(3) is the Euclidean group of rigid motions in $\mathbb{R}^3$. The rigid motion e_i moves patch P_i to its correct position with respect to the global coordinate system. Our goal is to estimate the rigid motions $e_1, \ldots, e_N$ (up to a global rigid motion) that will properly align all the patches in a globally consistent way. To achieve this goal, we first estimate the alignment between any pair of patches P_i and P_j that have enough nodes in common, a procedure we detail later in Section 3.5.6. The alignment of patches P_i and P_j provides a (perhaps noisy) measurement for the ratio $e_i e_j^{-1}$ in Euc(3). We solve the resulting synchronization problem in a globally consistent manner, such that information from local alignments propagates to pairs of non-overlapping patches. This is done by replacing the synchronization problem over Euc(3) by two different consecutive synchronization problems.

In the first synchronization problem, we simultaneously find the reflections and rotations of all the patches using the eigenvector synchronization algorithm over the group O(3) of orthogonal matrices. When prior information on the reflections of some patches is available, one may choose to replace this first step by two consecutive synchronization problems, i.e., first estimate the missing rotations by doing synchronization over $\mathbb{Z}_2$ with molecular fragment information, as described in Section 4.4, followed by another synchronization problem over SO(3) to estimate the rotations of all patches. Once both reflections and rotations are estimated, we estimate the translations by solving an overdetermined linear system. Taken as a whole, the algorithm integrates all the available local information into a global coordinate system over several steps by using the eigenvector synchronization algorithm and least squares over the isometries of the Euclidean space. The main advantage of the eigenvector method is that it can recover the reflections and rotations even if many of the pairwise alignments are incorrect. The algorithm is summarized in Table 3.1.

3.4.1 Step 1: Synchronization over O(3) to estimate reflections and rotations

As mentioned earlier, for every patch P_i that was already embedded in its local frame, we need to estimate whether or not it needs to be reflected with respect to the global coordinate system, and what is the rotation that aligns it in the same coordinate system. In 2D-ASAP, we first estimated the reflections, and based on that, we further estimated the rotations. However, it makes sense to ask whether one can combine the two steps, and perhaps further

INPUT	$G=(V,E),\ \|V\|=n,\ \|E\|=m,\ d_{ij}$ for $(i,j)\in E$
Pre-processing Step Patch Localization	1. Break the measurement graph G into N globally rigid or weakly uniquely localizable patches $P_1,\ldots,P_N$. 2. Embed each patch P_i separately using either FULL-SDP (for noiseless data), or SNL-SDP (for noisy data), or cMDS (for complete patches).
Step 1 Estimating Reflections and Rotations	1. Align all pairs of patches (P_i,P_j) that have enough nodes in common. 2. Estimate their relative rotation and possibly reflection $h_{ij}\in O(3)$. 3. Build a sparse $3N\times 3N$ symmetric matrix $H=(h_{ij})$ as defined in (4.2). 4. Define $\mathcal{H}=D^{-1}H$, where D is a diagonal matrix with $D_{3i-2,3i-2}=D_{3i-1,3i-1}=D_{3i,3i}=deg(i)$, for $i=1,\ldots,N$. 5. Compute the top 3 eigenvectors $v_i^{\mathcal{H}}$ of $\mathcal{H}$ satisfying $\mathcal{H}v_i^{\mathcal{H}}=\lambda_i^{\mathcal{H}}v_i^{\mathcal{H}}, i=1,2,3$. 6. Estimate the global reflection and rotation of patch P_i by the orthogonal matrix $\hat{h}_i$ that is closest to $\widetilde{h}_i$ in Frobenius norm, where $\widetilde{h}_i$ is the submatrix corresponding to the i$^{\text{th}}$ patch in the $3N\times 3$ matrix formed by the top three eigenvectors $[v_1^{\mathcal{H}}v_2^{\mathcal{H}}v_3^{\mathcal{H}}]$. 7. Update the current embedding of patch P_i by applying the orthogonal transformation $\hat{h}_i$ obtained above (rotation and possibly reflection)
Step 2 Estimating Translations	1. Build the $m\times n$ overdetermined system of linear equations given in (3.14), after applying the median-based denoising heuristic. 2. Include the anchors information (if available) into the linear system. 3. Compute the least squares solution for the x-axis, y-axis and z-axis coordinates.
OUTPUT	Estimated coordinates $\hat{p}_1,\ldots,\hat{p}_n$

Table 3.1: Overview of the 3D-ASAP Algorithm

increase the robustness to noise of the algorithm. By doing this, information contained in the pairwise rotation matrices helps in better estimating the reflections, and vice-versa, information on the pairwise reflection between patches helps in improving the estimated rotations. Combining these two steps also reduces the computational effort by half, since we need to run the eigenvector synchronization algorithm only once.

We denote the orthogonal transformation of patch P_i by $h_i\in$ O(3), which is defined up to a global orthogonal rotation and reflection. The alignment of every pair of patches P_i and P_j whose intersection is sufficiently large, provides a measurement h_{ij} (a 3×3 orthogonal matrix) for the ratio $h_ih_j^{-1}$. However, some ratio measurements can be corrupted because of errors in the embedding of the patches due to noise in the measured distances. We denote by $G^P=(V^P,E^P)$ the patch graph whose vertices V^P are the patches $P_1,\ldots,P_N$, and two patches P_i and P_j are adjacent, $(P_i,P_j)\in E^P$, iff they have enough vertices in common to be aligned such that the ratio $h_ih_j^{-1}$ can be estimated. We let A^P denote the adjacency matrix of the patch graph, i.e., $A^P_{ij}=1$ if $(P_i,P_j)\in E^P$, and $A^P_{ij}=0$ otherwise. Obviously, two patches that are far apart and have no common nodes cannot be aligned, and there must be enough[4] overlapping nodes to make the alignment possible. Figures 3.8 and 3.9 show a typical example of the sizes of the patches we consider, as well as their intersection sizes.

The first step of 3D-ASAP is to estimate the appropriate rotation and reflection of each patch. To that end, we use the eigenvector synchronization method as it was shown to

[4]E.g., four common vertices, although the precise definition of "enough" will be discussed later.

perform well even in the presence of a large number of errors. The eigenvector method starts off by building the following $3N \times 3N$ sparse symmetric matrix $H = (h_{ij})$, where h_{ij} is the a 3×3 orthogonal matrix that aligns patches P_i and P_j

$$H_{ij} = \begin{cases} h_{ij} & (i,j) \in E^P \text{ (P_i and P_j have enough points in common)} \\ O_{3\times 3} & (i,j) \notin E^P \text{ (P_i and P_j cannot be aligned)} \end{cases} \tag{3.1}$$

We explain in more detail in Section 3.5.6 the procedure by which we align pairs of patches, if such an alignment is at all possible.

Prior to computing the top eigenvectors of the matrix H, as introduced originally in [92], we choose to use the following normalization (similar to 2D-ASAP in [32]). Let D be a $3N \times 3N$ diagonal matrix[5], whose entries are given by $D_{3i-2,3i-2} = D_{3i-1,3i-1} = D_{3i,3i} = deg(i)$, for $i = 1, \ldots, N$. We define the matrix

$$\mathcal{H} = D^{-1}H, \tag{3.2}$$

which is similar to the symmetric matrix $D^{-1/2}HD^{-1/2}$ through

$$\mathcal{H} = D^{-1/2}(D^{-1/2}HD^{-1/2})D^{1/2}.$$

Therefore, $\mathcal{H}$ has $3N$ real eigenvalues $\lambda_1^{\mathcal{H}} \geq \lambda_2^{\mathcal{H}} \geq \lambda_3^{\mathcal{H}} \geq \lambda_4^{\mathcal{H}} \geq \cdots \geq \lambda_{3N}^{\mathcal{H}}$ with corresponding $3N$ orthogonal eigenvectors $v_1^{\mathcal{H}}, \ldots, v_{3N}^{\mathcal{H}}$, satisfying $\mathcal{H}v_i^{\mathcal{H}} = \lambda_i^H v_i^{\mathcal{H}}$. As shown in the next paragraphs, in the noise free case, $\lambda_1^{\mathcal{H}} = \lambda_2^{\mathcal{H}} = \lambda_3^{\mathcal{H}}$, and furthermore, if the patch graph is connected, then $\lambda_3^{\mathcal{H}} > \lambda_4^{\mathcal{H}}$. We define the estimated orthogonal transformations $\hat{h}_1, \ldots, \hat{h}_N \in$ O(3) using the top three eigenvectors $v_1^{\mathcal{H}}, v_2^{\mathcal{H}}, v_3^{\mathcal{H}}$, following the approach used in [93]. In Section 4.3 of the next chapter we show that, in the noise free case, the top three eigenvectors of $\mathcal{H}$ perfectly recover the true solution $h_1, \ldots, h_N \in$ O(3), up to a global orthogonal transformation.

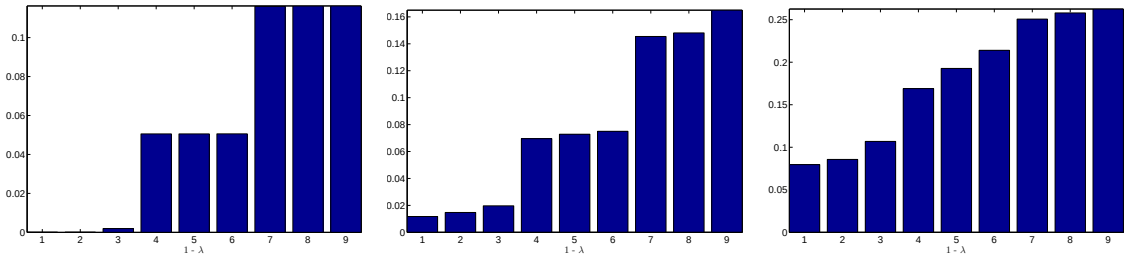
However, when distance measurements are noisy and the pairwise alignments between patches are inaccurate, an estimated transformation $\widetilde{h}_i$ may not coincide with h_i, and in fact may not even be an orthogonal transformation. For that reason, we estimate h_i by the closest orthogonal matrix to $\widetilde{h}_i$ in the Frobenius matrix norm[6]

$$\hat{h}_i = \underset{X \in O(3)}{\operatorname{argmin}} \|\widetilde{h}_i - X\|_F \tag{3.3}$$

We do this by using the well known procedure (e.g.,[3]), $\hat{h}_i = U_i V_i^T$, where $\widetilde{h}_i = U_i \Sigma_i V_i^T$ is the singular value decomposition of $\widetilde{h}_i$, see also [36] and [67]. Note that the estimation of the orthogonal transformations of the patches are up to a global orthogonal transformation (i.e., a global rotation and reflection with respect to the original embedding). Also, the only difference between this step and the angular synchronization algorithm in [92] is the normalization of the matrix prior to the computation of the top eigenvector. The usefulness of this normalization was first demonstrated in 2D-ASAP, in the synchronization process over $\mathbb{Z}_2$ and SO(2).

[5]The diagonal matrix D should not be confused with the partial distance matrix.

[6]We remind the reader that the Frobenius norm of an $m \times n$ matrix A can be defined in several ways $\|A\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n |a_{ij}|^2 = Tr(A^T A) = \sum_{i=1}^{\min(n,m)} \sigma_i^2$, where σ_i are the singular values of A.



(a) $\eta = 0\%$, $\tau = 0\%$, and $MSE = 6e-4$ (b) $\eta = 20\%$, $\tau = 0\%$, and $MSE = 0.05$ (c) $\eta = 40\%$, $\tau = 4\%$, and $MSE = 0.36$

Figure 3.3: Bar-plot of the top 9 eigenvalues of $\mathcal{H}$ for the UNITCUBE and various noise levels η. The resulting error rate τ is the percentage of patches whose reflection was incorrectly estimated. To ease the visualization of the eigenvalues of $\mathcal{H}$, we choose to plot $1 - \lambda^{\mathcal{H}}$ because the top eigenvalues of $\mathcal{H}$ tend to pile up near 1, so it is difficult to differentiate between them by looking at the bar plot of $\lambda^{\mathcal{H}}$.

We use the mean squared error (MSE) to measure the accuracy of this step of the algorithm in estimating the orthogonal transformations. To that end, we look for an optimal orthogonal transformation $\hat{O} \in \mathrm{O}(3)$ that minimizes the sum of squared distances between the estimated orthogonal transformations and the true ones:

$$\hat{O} = \underset{O \in O(3)}{\operatorname{argmin}} \sum_{i=1}^{N} \|h_i - O\hat{h}_i\|_F^2 \tag{3.4}$$

In other words, $\hat{O}$ is the optimal solution to the registration problem between two sets of orthogonal transformations in the least squares sense. Following the analysis of [93], we make use of properties of the trace such as $Tr(AB) = Tr(BA)$, $Tr(A) = Tr(A^T)$ and notice that

$$\begin{aligned} \sum_{i=1}^{N} \|h_i - O\hat{h}_i\|_F^2 &= \sum_{i=1}^{N} Tr\left[\left(h_i - O\hat{h}_i\right)\left(h_i - O\hat{h}_i\right)^T\right] \\ &= \sum_{i=1}^{N} Tr\left[2I - 2O\hat{h}_i h_i^T\right] = 6N - 2Tr\left[O\sum_{i=1}^{N} \hat{h}_i h_i^T\right] \end{aligned} \tag{3.5}$$

If we let Q denote the 3×3 matrix

$$Q = \frac{1}{N}\sum_{i=1}^{N} \hat{h}_i h_i^T \tag{3.6}$$

it follows from (3.5) that the MSE is given by minimizing

$$\frac{1}{N}\sum_{i=1}^{N} \|h_i - O\hat{h}_i\|_F^2 = 6 - 2Tr(OQ). \tag{3.7}$$

In [3] it is proven that $Tr(OQ) \leq Tr(VU^TQ)$, for all $O \in O(3)$, where $Q = U\Sigma V^T$ is the singular value decomposition of Q. Therefore, the MSE is minimized by the orthogonal matrix $\hat{O} = VU^T$ and is given by

$$\frac{1}{N}\sum_{i=1}^{N}\|h_i - \hat{O}\hat{h}_i\|_F^2 = 6 - 2Tr(VU^TU\Sigma V^T) = 6 - 2\sum_{k=1}^{3}\sigma_k \tag{3.8}$$

where $\sigma_1, \sigma_2, \sigma_3$ are the singular values of Q. Therefore, whenever Q is an orthogonal matrix for which $\sigma_1 = \sigma_2 = \sigma_3 = 1$, the MSE vanishes. Indeed, the numerical experiments in Table 3.2 confirm that for noiseless data, the MSE is very close to zero. To illustrate the success of the eigenvector method in estimating the reflections, we also compute τ, the percentage of patches whose reflection was incorrectly estimated. Finally, the last two columns in Table 3.2 show the recovery errors when, instead of doing synchronization over O(3), we first synchronize over $\mathbb{Z}_2$ followed by SO(3).

	O(3)		$\mathbb{Z}_2$ and SO(3)	
η	τ	MSE	τ	MSE
0%	0%	6e-4	0%	7e-4
10%	0%	0.01	0%	0.01
20%	0%	0.05	0%	0.05
30%	5.8%	0.35	5.3%	0.32
40%	4%	0.36	5%	0.40
50%	7.4%	0.65	9%	0.68

Table 3.2: The errors in estimating the reflections and rotations when aligning the $N = 200$ patches resulting from for the UNITCUBE graph on $n = 212$ vertices, at various levels of noise. We used τ to denote the percentage of patches whose reflection was incorrectly estimated.

3.4.2 Step 2: Synchronization over $\mathbb{R}^3$ to estimate translations

The final step of the 3D-ASAP algorithm is computing the global translations of all patches and recovering the true coordinates. For each patch P_k, we denote by $G_k = (V_k, E_k)$[7] the graph associated to patch P_k, where V_k is the set of nodes in P_k, and E_k is the set of edges induced by V_k in the measurement graph $G = (V, E)$. We denote by $p_i^{(k)} = (x_i^{(k)}, y_i^{(k)}, z_i^{(k)})^T$ the known local frame coordinates of node $i \in V_k$ in the embedding of patch P_k (see Figure 3.4).

At this stage of the algorithm, each patch P_k has been properly reflected and rotated so that the local frame coordinates are consistent with the global coordinates, up to a translation $t^{(k)} \in \mathbb{R}^3$. In the noise-free case we should therefore have

$$p_i = p_i^{(k)} + t^{(k)}, \quad i \in V_k, \quad k = 1, \ldots, N. \tag{3.9}$$

[7]Not to be confused with $G(i) = (V(i), E(i))$ defined in the beginning of this section.

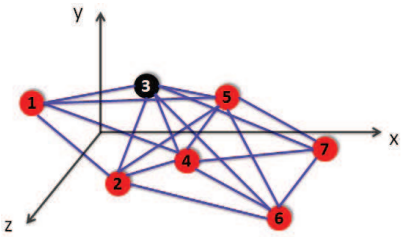


Figure 3.4: An embedding of a patch P_k in its local coordinate system (frame) after it was appropriately reflected and rotated. In the noise-free case, the coordinates $p_i^{(k)} = (x_i^{(k)}, y_i^{(k)}, z_i^{(k)})^T$ agree with the global positioning $p_i = (x_i, y_i, z_i)^T$ up to some translation $t^{(k)}$ (unique to all i in V_k).

We can estimate the global coordinates $p_1, \ldots, p_n$ as the least squares solution to the overdetermined system of linear equations (3.9), while ignoring the by-product translations $t^{(1)}, \ldots, t^{(N)}$. In practice, we write a linear system for the displacement vectors $p_i - p_j$ for which the translations have been eliminated. Indeed, from (3.9) it follows that each edge $(i, j) \in E_k$ contributes a linear equation of the form [8]

$$p_i - p_j = p_i^{(k)} - p_j^{(k)}, \quad (i, j) \in E_k, \quad k = 1, \ldots, N. \tag{3.10}$$

In terms of the x, y and z global coordinates of nodes i and j, (3.10) is equivalent to

$$x_i - x_j \;\; = \;\; x_i^{(k)} - x_j^{(k)}, \quad (i, j) \in E_k, \quad k = 1, \ldots, N, \tag{3.11}$$

and similarly for the y and z equations. We solve these three linear systems separately, and recover the coordinates $x_1, \ldots, x_n$, $y_1, \ldots, y_n$, and $z_1, \ldots, z_n$. Let T be the least squares matrix associated with the overdetermined linear system in (3.11), x be the $n \times 1$ vector representing the x-coordinates of all nodes, and b^x be the vector with entries given by the right-hand side of (3.11). Using this notation, the system of equations given by (3.11) can be written as

$$Tx = b^x, \tag{3.12}$$

and similarly for the y and z coordinates. Note that the matrix T is sparse with only two non-zero entries per row and that the all-ones vector $\mathbf{1} = (1, 1, \ldots, 1)^T$ is in the null space of T, i.e., $T\mathbf{1} = 0$, so we can find the coordinates only up to a global translation.

To avoid building a very large least squares matrix, we combine the information provided by the same edges across different patches in only one equation, as opposed to having one equation per patch. For 2D-ASAP, we achieved this by adding up all equations of the form (3.11) corresponding to the same edge (i, j) from different patches, into a single equation, i.e.,

$$\sum_{k \in \{1,\ldots,N\} \; s.t. (i,j) \in E_k} x_i - x_j = \sum_{k \in \{1,\ldots,N\} \; s.t. (i,j) \in E_k} x_i^{(k)} - x_j^{(k)}, \quad (i, j) \in E, \tag{3.13}$$

[8] In fact, we can write such equations for every $i, j \in V_k$ but choose to do so only for edges of the original measurement graph.

and similarly for the y and z-coordinates (see Section 2.3.3). For very noisy distance measurements, the displacements $x_i^{(k)} - x_j^{(k)}$ will also be corrupted by noise and the motivation for (3.13) was that adding up such noisy values will average out the noise. However, as the noise level increases, some of the embedded patches will be highly inaccurate and will thus generate outliers in the list of displacements above. To make this step of the algorithm more robust to outliers, instead of averaging over all displacements, we select the median value of the displacements and use it to build the least squares matrix

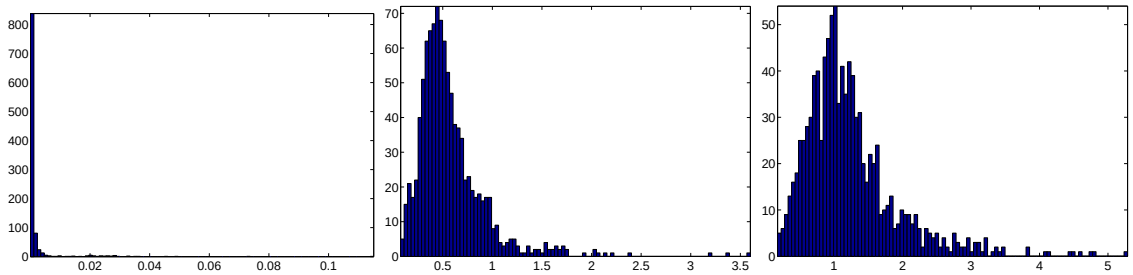
$$x_i - x_j = \underset{k\in\{1,\ldots,N\}\ s.t.(i,j)\in E_k}{\text{median}} \{x_i^{(k)} - x_j^{(k)}\}, \quad (i,j) \in E, \tag{3.14}$$

We denote the resulting $m \times n$ matrix by $\tilde{T}$, and its $m \times 1$ right-hand-side vector by $\tilde{b}^x$. Note that $\tilde{T}$ has only two nonzero entries per row [9], $\tilde{T}_{e,i} = 1, \tilde{T}_{e,j} = -1$, where e is the row index corresponding to the edge (i,j). The least squares solution $\hat{p} = \hat{p}_1, \ldots, \hat{p}_n$ to

$$\tilde{T}x = \tilde{b}^x, \quad \tilde{T}y = \tilde{b}^y, \text{ and } \tilde{T}z = \tilde{b}^z, \tag{3.15}$$

is our estimate for the coordinates $p = p_1, \ldots, p_n$, up to a global rigid transformation.

Whenever the ground truth solution p is available, we can compare our estimate $\hat{p}$ with p. To that end, we remove the global reflection, rotation and translation from $\hat{p}$, by computing the best procrustes alignment between p and $\hat{p}$, i.e. $\tilde{p} = O\hat{p} + t$, where O is an orthogonal rotation and reflection matrix, and t a translation vector, such that we minimize the distance between the original configuration p and $\tilde{p}$, as measured by the least squares criterion $\sum_{i=1}^{n} \|p_i - \tilde{p}_i\|^2$. Figure 3.5 shows the histogram of errors in the coordinates, where the error associated with node i is given by $\| p_i - \hat{p}_i \|$.



(a) $\eta = 0\%, ERR_c = 2e-3$ (b) $\eta = 30\%, ERR_c = 0.57$ (c) $\eta = 50\%, ERR_c = 1.23$

Figure 3.5: Histograms of coordinate errors $\|p_i - \hat{p}_i\|$ for all atoms in the 1d3z molecule, for different levels of noise. In all figures, the x-axis is measured in Angstroms. Note the change of scale for Figure (a), and the fact that the largest error showed there is 0.12. We used ERR_c to denote the average coordinate error of all atoms.

We remark that in 3D-ASAP anchor information can be incorporated similarly to the 2D-ASAP algorithm, as explained at the end of Section 2.3.3; however we do not elaborate on this here since there are no anchors in the molecule problem.

[9] Note that some edges in E may not be contained in any patch P_k, in which case the corresponding row in $\tilde{T}$ has only zero entries.

3.5 Extracting, embedding, and aligning patches

This section describes how to break up the measurement graph into patches, and how to embed and pairwise align the resulting patches. We start in Section 3.5.1 with a description of how to extract globally rigid subgraphs from 1-hop neighborhood graphs, and discuss the advantages that k-star graphs bring. However, in practice, we do not follow either approach to extract patches, since SDP-based methods may still produce inaccurate localizations of globally rigid patches even in the case of noiseless data. Therefore, in Section 3.5.2 we recall a recent result of [96] on *uniquely d-localizable* graphs, which can be accurately localized by SDP-based methods. We thus lay the ground for the notion of *weakly uniquely localizable* (WUL) graphs, which we introduce with the purpose of being able to localize the resulting patches even when the distance measurements are noisy. Section 3.5.3 discusses the issue of finding "pseudo-anchor" nodes, which are needed when extracting WUL subgraphs. In Section 3.5.4 we discuss several SDP-relaxations to the graph localization problem, which we use to embed the WUL patches. In Section 3.5.5 we remark on several additional constraints specific to the molecule problem, which are currently not incorporated in 3D-ASAP. Finally, Section 3.5.6 explains the procedure for aligning a pair of overlapping patches.

3.5.1 Extracting globally rigid subgraphs

Although there is no combinatorial characterization of global rigidity in $\mathbb{R}^3$ (but only necessary conditions [55]), one may exploit the fact that a 1-hop neighborhood subgraph is always a "star" graph, i.e., a graph with at least one central node connected to everybody else. The process of coning a graph G adds a new vertex v, and adds edges from v to all original vertices in G, creating the cone graph $G * v$. A recent result of [28] states that a graph is generically globally rigid in $\mathbb{R}^{d-1}$ iff the cone graph is generically globally rigid in $\mathbb{R}^d$. This result allows us to reduce the notion of global rigidity in $\mathbb{R}^3$ to global rigidity in $\mathbb{R}^2$, after removing the center node. We recall that sufficient and necessary conditions for two dimensional combinatorial global rigidity have been recently established [60, 55], i.e., 3-connectivity and redundant rigidity (meaning the graph remains locally rigid after the removal of any single edge). Therefore, breaking a graph into maximally globally rigid components amounts to finding the maximally 3-connected components, and furthermore, extracting the maximally redundantly rigid components. The first $O(n+m)$ algorithm for finding the 3-connected components of a general graph was given by Hopcroft and Tarjan [58]. A combinatorial characterization of redundant rigidity in dimension two was given in [55], together with an $O(n^2)$ algorithm.

The rest of this section presents an alternative method for extracting globally rigid subgraphs, while avoiding the notion of redundant rigidity. Motivated by the fact that 1-hop neighborhood graphs may have multiple centers (especially in random geometric graphs), we introduce the following family of graphs. A k-**star graph** is a graph which contains at least k vertices (centers) that are connected to all remaining nodes. For example, for each node i, the local graph $G(i)$ composed of the central node i and all its neighbors takes the form of a 1-star graph (which we simply refer to as a star graph). Note that in our defini-

tion, unlike perhaps more conventional definitions of star graphs, we allow edges between non-central nodes to exist.

Proposition 3.5.1. *A $(k-1)$-star graph is generically globally rigid in $\mathbb{R}^k$ iff it is $(k+1)$-vertex-connected.*

Proof. We prove the statement in the proposition by induction on k. The case $k = 2$ was previously shown in [32]. Assuming the statement holds true for $k-2$, we show it remains true for $k-1$ as well.

Let H be a $(k+1)$-vertex-connected $(k-1)-$star graph, $\mathcal{S} = \{v_1, \ldots, v_{k-1}\}$ be its center nodes, and H^* the graph obtained by removing one of its center nodes. Since H is $(k+1)$-vertex-connected then H^* must be k-vertex-connected, since otherwise, if u is a cut-vertex in H^*, then $\{v_1, u\}$ is a vertex-cut of size 2 in H, a contradiction. By induction, H^* is generically globally rigid in $\mathbb{R}^{k-1}$, since it is a $(k-2)$-star graph that is k-vertex-connected. Using the coning theorem, the generic globally rigidity of H^* in $\mathbb{R}^{k-1}$ implies that H is generically globally rigid in $\mathbb{R}^k$.

The converse is a well known statement in rigidity theory [55]. We first say that a framework in $\mathbb{R}^k$ allows a reflection if a separating set of vertices lies in $(k-1)$-dimensional subspace. However, realization for which more than k vertices lie in a $(k-1)$-dimensional subspace are not generic, and therefore, for generic frameworks, reflections occur when there is a subset of k or fewer vertices whose removal disconnects the graph, i.e., G is not $(k+1)$-vertex-connected. In other words, if H is generically globally rigid in $\mathbb{R}^k$, then it must be $(k+1)$-vertex-connected. □

Using the above result for $k = 3$ implies that if the 1-hop neighborhood $G(i)$ of node i has another center node $j \neq i$, then one can break $G(i)$ into maximally globally rigid subgraphs (in $\mathbb{R}^3$) by simply finding the 4-connected components of $G(i)$. Since $G(i)$ has two center nodes i and j, the problem amounts to finding the 2-connected components of the remaining graph.

This observation suggests another possible approach to solving the network localization problem. Instead of breaking the initial graph G into patches by first using 1-hop neighborhoods of each node, one can start by considering the 1-hop neighborhood $G(i,j)$ of a pair of nodes $(i,j) \in E(G)$, where vertices of graph $G(i,j)$ are the intersection of the

1-hop neighbors of nodes i and j. This approach assures that $G(i,j)$ is a 2-star graph, and by the above result, can be easily broken into maximally globally rigid subgraphs, without involving the notion of redundant rigidity.

Note that, in practice, we do not follow either of the two approaches introduced in this section, since SDP-based methods may compute inaccurate localizations of globally rigid graphs, even in the case of noiseless data. Instead, we use the notion of weakly uniquely localizable graphs, which we introduce in the next section.

3.5.2 Extracting Weakly Uniquely Localizable (WUL) subgraphs

We first recall some of the notation introduced earlier, that is needed throughout this section and Section 4.4 on synchronization over $\mathbb{Z}_2$ with anchors. We consider a sensor network in $\mathbb{R}^3$ with k anchors denoted by $\mathcal{A}$, and n sensors denoted by $\mathcal{S}$. An anchor is a node whose location $a_i \in \mathbb{R}^3$ is readily available, $i = 1, \ldots, k$, and a sensor is a node whose location x_j is to be determined, $j = 1, \ldots, n$. Note that for aesthetic reasons and consistency with the notation used in [96], we use x in this section to denote the true coordinates, as opposed to p used throughout the rest of the thesis[10]. We denote by d_{ij} the Euclidean distance between a pair of nodes, $(i,j) \in \mathcal{A} \cup \mathcal{S}$. In most applications, not all pairwise distance measurements are available, therefore we denote by $E(\mathcal{S},\mathcal{S})$ and $E(\mathcal{S},\mathcal{A})$ the set of edges denoting the measured sensor-sensor and sensor-anchor distances. We represent the available distance measurements in an undirected graph $G = (V,E)$ with vertex set $V = \mathcal{A} \cup \mathcal{S}$ of size $|V| = n+k$, and edge set of size $|E| = m$. An edge of the graph corresponds to a distance constraint, that is $(i,j) \in E$ iff the distance between nodes i and j is available and equals $d_{ij} = d_{ji}$, where $i,j \in \mathcal{A} \cup \mathcal{S}$. We denote the partial distance measurements matrix by $D = \{d_{ij} : (i,j) \in E(\mathcal{S},\mathcal{S}) \cup E(\mathcal{S},\mathcal{A})\}$. A solution x together with the anchor set a comprise a *localization* or *realization* $p = (x,a)$ of G. A *framework* in $\mathbb{R}^d$ is the ensemble (G,p), i.e., the graph G together with the realization p which assigns a point p_i in $\mathbb{R}^d$ to each vertex i of the graph. We refer the reader to Chapter 1 for more background on rigidity theory.

Given a partial set of noiseless distances and the anchor set a, the graph realization problem can be formulated as the following system

$$\begin{aligned} \|x_i - x_j\|_2^2 &= d_{ij}^2 \quad \text{for } (i,j) \in E(\mathcal{S},\mathcal{S}) \\ \|a_i - x_j\|_2^2 &= d_{ij}^2 \quad \text{for } (i,j) \in E(\mathcal{S},\mathcal{A}) \\ x_i &\in \mathbb{R}^d \quad \text{for } i = 1, \ldots, n \end{aligned} \tag{3.16}$$

Unless the above system has enough constraints (i.e., the graph G has sufficiently many edges), then G is not globally rigid and there could be multiple solutions. However, if the graph G is known to be (generically) globally rigid in $\mathbb{R}^3$, and there are at least four anchors (i.e., $k \geq 4$), and G admits a generic realization[11], then (3.16) admits a unique solution. Due to recent results on the characterization of generic global rigidity, there now exists a

[10] Not to be confused with the x-axis projections of the distances in the least squares step.

[11] A realization is *generic* if the coordinates do not satisfy any non-zero polynomial equation with integer coefficients.

randomized efficient algorithm that verifies if a given graph is generically globally rigid in $\mathbb{R}^d$ [45]. However, this efficient algorithm does not translate into an efficient method for actually computing a realization of G. Knowing that a graph is generically globally rigid in $\mathbb{R}^d$ still leaves the graph realization problem intractable, as shown in [7]. Motivated by this gap between deciding if a graph is generically globally rigid and computing its realization (if it exists), So and Ye introduced the following notion of unique d-localizability [96]. An instance (G, p) of the graph localization problem is said to be *uniquely d-localizable* if

1. the system (3.16) has a unique solution $\tilde{x} = (\tilde{x}_1; \ldots; \tilde{x}_n) \in \mathbb{R}^{nd}$, and
2. for any $l > d$, $\tilde{x} = ((\tilde{x}_1; \mathbf{0}), \ldots, (\tilde{x}_n; \mathbf{0})) \in \mathbb{R}^{nl}$ is the unique solution to the following system:

$$\begin{aligned} \|x_i - x_j\|_2^2 &= d_{ij}^2 \quad \text{for } (i,j) \in E(\mathcal{S}, \mathcal{S}) \\ \|(a_i; \mathbf{0}) - x_j\|_2^2 &= d_{ij}^2 \quad \text{for } (i,j) \in E(\mathcal{S}, \mathcal{A}) \\ x_i &\in \mathbb{R}^l \quad \text{for } i = 1, \ldots, n \end{aligned} \tag{3.17}$$

where $(v; \mathbf{0})$ denotes the concatenation of a vector v of size d with the all-zeros vector $\mathbf{0}$ of size $l - d$. The second condition states that the problem cannot have a non-trivial localization in some higher dimensional space $\mathbb{R}^l$ (i.e, a localization different from the one obtained by setting $x_j = (\tilde{x}_j; \mathbf{0})$ for $j = 1, \ldots, n$), where anchor points are trivially augmented to $(a_i; \mathbf{0})$, for $i = 1, \ldots, k$. A crucial observation should now be made: unlike global rigidity, which is a generic property of the graph G, the notion of *unique localizability* depends not only on the underlying graph G but also on the particular realization p, i.e., it depends on the framework (G, p).

We now introduce the notion of a weakly uniquely localizable graph, essential for the preprocessing step of the 3D-ASAP algorithm, where we break the original graph into overlapping patches. A graph is *weakly uniquely d-localizable* if there exists at least one realization $p \in \mathbb{R}^{(n+k)d}$ (we call this a certificate realization) such that the framework (G, p) is uniquely localizable. Note that if a framework (G, p) is uniquely localizable, then G is a weakly uniquely localizable graph; however the reverse is not necessarily true since unique localizability is not a generic property.

The advantage of working with uniquely localizable graphs becomes clear in light of the following result by [96], which states that the problem of deciding whether a given graph realization problem is uniquely localizable, as well as the problem of determining the node positions of such a uniquely localizable instance, can be solved efficiently by considering the following SDP

$$\begin{aligned} \text{maximize} \quad & 0 \\ \text{subject to} \quad & (\mathbf{0}; e_i - e_j)(\mathbf{0}; e_i - e_j)^T \cdot Z = d_{ij}^2, \quad \text{for } (i,j) \in E(\mathcal{S}, \mathcal{S}) \\ & (a_i; -e_j)(a_i; -e_j)^T \cdot Z = d_{ij}^2, \quad \text{for } (i,j) \in E(\mathcal{S}, \mathcal{A}) \\ & Z_{1:d,1:d} = I_d \\ & Z \in \mathcal{K}^{n+d} \end{aligned} \tag{3.18}$$

where e_i denotes the all-zeros vector with a 1 in the i^{th} entry, and $\mathcal{K}^{n+d} = \{Z_{(n+d)\times(n+d)} | Z = \begin{bmatrix} I_d & X \\ X^T & Y \end{bmatrix} \succeq 0\}$, where $Z \succeq 0$ means that Z is a positive semidefinite matrix. The SDP method relaxes the constraint $Y = XX^T$ to $Y \succeq XX^T$, i.e., $Y - XX^T \succeq 0$, which is equivalent to the last condition in (3.18). The following predictor for uniquely localizable graphs introduced in [96], established for the first time that the graph realization problem is uniquely localizable if and only if the relaxation solution computed by an interior point algorithm (which generates feasible solutions of max-rank) has rank d and $Y = XX^T$.

Theorem 3.5.2 ([96, Theorem 2])**.** *Suppose G is a connected graph. Then the following statements are equivalent:*

a) Problem (3.16) is uniquely localizable

b) The max-rank solution matrix Z of (3.18) has rank d

c) The solution matrix Z represented by (b) satisfies $Y = XX^T$.

Algorithm 1 summarizes our approach for extracting a WUL subgraph of a given graph, motivated by the results of Theorem 3.5.2 and the intuition and numerical experiments described in the next paragraph. Note, however, that the statements in Theorem 3.5.2 hold true as long as the graph G has at least four anchor nodes. While this may seem a very restrictive condition (since in many real life applications anchor information is rarely available) there is an easy way to get around this, provided the graph contains a clique (complete subgraph) of size at least 4. As discussed in Section 3.5.3, a patch of size at least 10 contains such a clique with very high probability. Once such a clique has been found, one may use cMDS to embed it and use the coordinates as anchors. We call such nodes *pseudo-anchors*.

Algorithm 1 Finding a weakly uniquely localizable (WUL) subgraph of a graph with four anchors or pseudo-anchors

Require: Simple graph $G = (V, E)$ with n sensors, k anchors and m edges corresponding to known pairwise Euclidean distances, and ϵ a small positive constant (e.g. 10^{-4})

1: Randomize a realization $p_1, \ldots, p_n$ in $\mathbb{R}^3$
2: If $k < 4$, find a complete subgraph of G on 4 vertices (i.e., K_4) and compute an embedding of it (using classical MDS). Denote the set of pseudo-anchors by $\mathcal{A}$
3: Solve the SDP relaxation problem formulated in (3.18) using the anchor set $\mathcal{A}$
4: Denote by the vector w the diagonal elements of the matrix $Y - XX^T$.
5: Find the subset of nodes $V_0 \in V \backslash \mathcal{A}$ such that $w_i < \epsilon$
6: Denote $G_0 = (V_0, E_0)$ the weakly uniquely localizable subgraph of G.

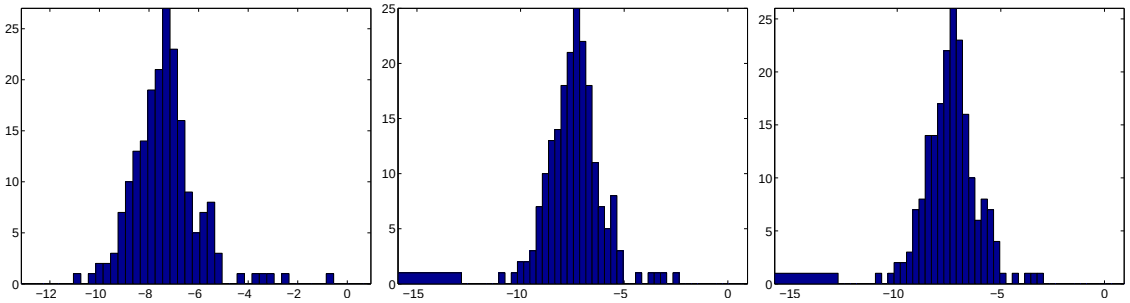
Two known necessary conditions for global rigidity in $\mathbb{R}^3$ are 4-connectivity and redundant rigidity (meaning the graph remains locally rigid after the removal of any single

edge) [55, 27]. One approach to breaking up a patch graph into globally rigid subgraphs, used by the ABBIE algorithm of [56], is to recurse on each 4-connected component of the graph, and then on each redundantly rigid subcomponent; but even then we are still not sure that the resulting subgraphs are globally rigid. Our approach is to extract a WUL subgraph from the 4-connected components of each patch. It may not be clear to the reader at this point what is the motivation for using weak unique localizability since it is not a generic property, and hence attempting to extract a WUL subgraph of a 4-connected graph seems meaningless since we do not know a priori what is the true realization of such a graph. However, we have observed in our numerical simulations that this approach significantly improves the accuracy of the embeddings. An intuitive motivation for this approach is the following. If the randomized realization in Algorithm 1 (or what remains of it after removing some of the nodes) is "faithful", meaning close enough to the true realization, then the WUL subgraph is perhaps generically uniquely localizable, and hence its localization using the SDP in (3.18) under the original distance constraints can be computed accurately, as predicted by Theorem 3.5.2. We also consider a slight variation of Algorithm 1, where we replace step 3 with the SDP relaxation introduced in the FULL-SDP algorithm of [17]. We refer to this different approach as Algorithm 2. Note that we also consider Algorithm 2 in our simulations only for computational reasons, since the running time of the FULL-SDP algorithm is significantly smaller compared to our CVX-based SDP implementation [48, 49] of problem (3.18).

Our intuition about the usefulness of the WUL subgraphs is supported by several numerical simulations. Figure 3.6 and Table 3.3 show the reconstruction errors of the patches (in terms of ANE, an error measure introduced in Section 3.8) in the following scenarios. In the first scenario, we directly embed each 4-connected component, without any prior preprocessing. In the second, respectively third, scenario we first extract a WUL subgraph from each 4-connected component using Algorithm 1, respectively Algorithm 2, and then embed the resulting subgraphs. Note that the subgraph embeddings are computed using FULL-SDP, respectively SNL-SDP, for noiseless, respectively noisy data. Figure 3.6 contains numerical results from the UNITCUBE graph with noiseless data, in the three scenarios presented above. As expected, the FULL-SDP embedding in scenario 1 gives the highest reconstruction error, at least one order of magnitude larger when compared to Algorithms 1 and 2. Surprisingly, Algorithm 2 produced more accurate reconstructions than Algorithm 1, despite its lower running time. These numerical computations suggest[12] that Theorem 3.5.2 remains true when the formulation in problem (3.18) is replaced by the one considered in the FULL-SDP algorithm [17].

The results detailed in Figure 3.6, while showing improvements of the second and third scenarios over the first one, may not entirely convince the reader of the usefulness of our proposed randomized algorithm, since in the first scenario a direct embedding of the patches using FULL-SDP already gives a very good reconstruction, i.e. $8.4e{-}4$ on average. We regard 4-connectivity a significant constraint that very likely renders a random geometric star graph to become globally rigid, thus diminishing the marginal improvements of the WUL extraction algorithm. To that end, we run experiments similar to those reported in Figure 3.6, but this time on the 1-hop neighborhood of each node in the UNITCUBE

[12] Personal communication by Yinyu Ye.



(a) Scenario 1: $\overline{ANE} = 8.4e-4$ (b) Scenario 2: $\overline{ANE} = 2.3e-5$ (c) Scenario 3: $\overline{ANE} = 7.2e-6$

Figure 3.6: Histogram of reconstruction errors (measured in ANE) for the noiseless UNITCUBE graph with $n = 212$ vertices, sensing radius $\rho = 0.3$ and average degree $deg = 17$. $\overline{ANE}$ denotes the average errors over all $N = 197$ patches. Note that the x-axis shows the ANE in logarithmic scale. Scenario 1: directly embedding the 4-connected components. Scenario 2: embedding the WUL subgraphs extracted using Algorithm 1. Scenario 3: embedding the WUL subgraphs extracted using Algorithm 2. Note that for the subgraph embeddings we use FULL-SDP.

graph, without further extracting the 4-connected components. In addition, we sparsify the graph by reducing the sensing radius from $\rho = 0.3$ to $\rho = 0.26$. Table 3.3 shows the reconstruction errors, at various levels of noise. Note that in the noise free case, scenarios 2 and 3 yield results which are an order of magnitude better than that of scenario 1, which returns a rather poor average ANE of $5.3e-02$. However, for the noisy case, these marginal improvements are considerably smaller.

η	Scenario 1	Scenario 2	Scenario 3
0%	5.3e-02	4.9e-03	1.3e-03
10%	8.8e-02	5.2e-02	5.3e-02
20%	1.5e-01	1.1e-01	1.1e-01
30%	2.3e-01	2.0e-01	2.0e-01

Table 3.3: Average reconstruction errors (measured in ANE) for the UNITCUBE graph with $n = 212$ vertices, sensing radius $\rho = 0.26$ and average degree $deg = 12$. Note that we only consider patches of size greater than or equal to 7, and there are 192 such patches. Scenario 1: directly embedding the 4-connected components. Scenario 2: embedding the WUL subgraphs extracted using Algorithm 1. Scenario 3: embedding the WUL subgraphs extracted using Algorithm 2. Note that for the subgraph embeddings we use FULL-SDP for noiseless data, and SNL-SDP for noisy data.

Table 3.4 shows the total number of nodes removed from the patches by Algorithms 1 and 2, the number of 1-hop neighborhoods which are readily WUL, and the running times. Indeed, for the sparser UNITCUBE graph with $\rho = 0.26$, the number of patches which are

already WUL is almost half, compared to the case of the denser graph with $\rho = 0.30$.

	$\rho = 0.30, N = 197$		$\rho = 0.26, N = 192$	
	Algorithm 1	Algorithm 2	Algorithm 1	Algorithm 2
Total nr of nodes removed	31	26	258	285
Nr of WUL patches	188	191	104	101
Running time (sec)	887	48	632	26

Table 3.4: Comparison of the two algorithms for extracting WUL subgraphs, for the UNICUBE graphs with sensing radius $\rho = 0.30$ and $\rho = 0.26$, and noise level $\eta = 0\%$. The WUL patches are those patches for which the subgraph extraction algorithms did not remove any nodes.

Finally, we remark on one of the consequences of our approach for breaking up the measurement graph. It is possible for a node not to be contained in any of the patches, even if it attaches in a globally rigid way to the rest of the measurement graph. An easy example is a star graph with four neighbors, no two of which are connected by an edge, as illustrated by the graph in Figure 3.7. However, we expect such pathological examples to be very unlikely in the case of random geometric graphs.

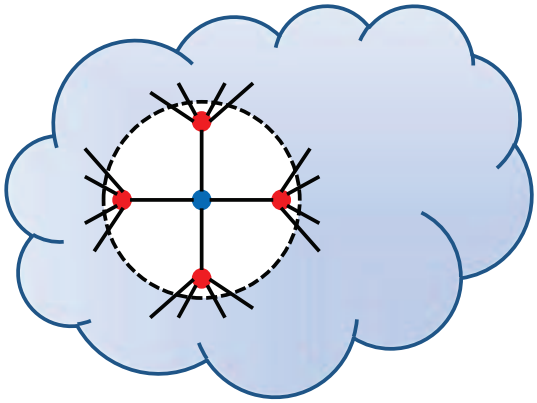


Figure 3.7: An example of a graph with a node that attaches globally rigidly to the rest of the graph, but is not contained in any patch, and thus it will be discarded by 3D-ASAP.

3.5.3 Finding pseudo-anchors

To satisfy the conditions of Theorem 3.5.2, at least $d + 1$ anchors are necessary for embedding a patch, hence for the molecule problem we need $k \geq 4$ such anchors in each patch. Since anchors are not usually available, one may ask whether it is still possible to find such a set of nodes that can be treated as anchors. If one were able to locate a clique of size at least $d+1$ inside a patch graph, then using cMDS it is possible to obtain accurate coordinates for the $d+1$ nodes and treat them as anchors Whenever this is possible, we call such a set of nodes *pseudo-anchors*. Intuitively, the geometric graph assumption should

lead one into thinking that if the patch graph is dense enough, it is very likely to find a complete subgraph on $d+1$ nodes. While a probabilistic analysis of random geometric graphs with forbidden K_{d+1} subgraphs is beyond of scope of this paper, we provide an intuitive connection with the problem of packing spheres inside a larger sphere, as well as numerical simulations that support the idea that a patch of size at least ≈ 10 contains four such pseudo-anchors with some high probability.

To find pseudo-anchors for a given patch graph G_i, one needs to locate a complete subgraph (clique) containing at least $d+1$ vertices. Since any patch G_i contains a center node that is connected to every other node in the patch, it suffices to find a clique of size at least 3 in the 1-hop neighborhood of the center node, i.e., to find a triangle in $G_i \backslash i$. Of course, if a graph is very dense (i.e., has high average degree) then it will be forced to contain such a triangle. To this end, we remind one of the first results in extremal graph theory (Mantel 1907), which states that any given graph on s vertices and more than $\frac{1}{4}s^2$ edges contains a triangle, the bipartite graph with $V_1 = V_2 = \frac{s}{2}$ being the unique extremal graph without a triangle and containing $\frac{1}{4}s^2$ edges. However, this quadratic bound which holds for general graphs is very unsatisfactory for the case of random geometric graphs.

Recall that we are using the geometric graph model, where two vertices are adjacent if and only if they are less than distance ρ apart. At a local level, one can think of the geometric graph model as placing an imaginary ball of radius ρ centered at node i, and connecting i to all nodes within this ball; and also connecting two neighbors j, k of i if and only if j and k are less than ρ units apart. Ignoring the center node i, the question to ask becomes how many nodes can one fit into a ball of radius ρ such that there exist at least d nodes whose pairwise distances are all less than ρ. In other words, given a geometric graph H inscribed in a sphere of radius ρ, what is the smallest number of nodes of H that forces the existence of a K_d.

The astute reader might immediately be lead into thinking that the problem above can be formulated as a sphere packing problem. Denote by $x_1, x_2, \ldots x_m$ the set of m nodes (ignoring the center node) contained in a sphere of radius ρ. We would like to know what is the smallest m such that at least $d = 3$ nodes are pairwise adjacent, i.e. their pairwise distances are all less than ρ.

To any node x_i associate a smaller sphere S_i of radius $\frac{\rho}{2}$. Two nodes x_i, x_j are adjacent, meaning less than distance ρ apart, if and only if their corresponding spheres S_i and S_j overlap. This line of thought leads one into thinking how many non-overlapping small spheres can one pack into a larger sphere. One detail not to be overlooked is that the radius of the larger sphere should be $\frac{3}{2}\rho$, and not ρ, since a node x_i at distance ρ from the center of the sphere has its corresponding sphere S_i contained in a sphere of radius $\frac{3}{2}\rho$. We have thus reduced the problem of asking what is the minimum size of a patch that would guarantee the existence of four anchors, to the problem of determining the smallest number of spheres of radius $\frac{1}{2}\rho$ that can be "packed" in a sphere of radius $\frac{3}{2}\rho$ such that at least three of the smaller spheres pairwise overlap. Rescaling the radii such that $\frac{3}{2}\rho = 1$ (hence $\frac{1}{2}\rho = \frac{1}{3}$), we ask the equivalent problem: *How many spheres of radius $\frac{1}{3}$ can be packed inside a sphere of radius 1, such that at least three spheres pairwise overlap.*

A related and slightly simpler problem is that of finding the densest packing on m equal spheres of radius r in a sphere of radius 1, such that no two of the small spheres overlap. This problem has been recently considered in more depth, and analytical solutions have

been obtained for several values of m. If $r = \frac{1}{3}$ (as in our case) then the answer is $m = 13$ and this constitutes a lower bound for our problem.

However, the arrangements of spheres that prevent the existence of three pairwise overlapping spheres are far from random, and motivated us to running the following experiment. For a given m, we generate m random spheres of radius $\frac{1}{3}$ inside the unit sphere, and count the number of times when at least three spheres pairwise overlap. We ran this experiment $15,000$ times for different values of $m = 5, 6, 7$, respectively 8, and obtained the following success rates $69\%, 87\%, 96\%$, respectively 99%, i.e., the percentage of times when the random realizations of spheres of radius $\frac{1}{3}$ inside a unit sphere produced three pairwise overlapping spheres. The simulation results show that about 9 spheres would guarantee, with very high probability, the existence of three pairwise overlapping spheres. In other words, for a patch of size 10 including the center node, there exist with high probability at least 4 nodes that are pairwise adjacent, i.e., the four pseudo-anchors we are looking for.

3.5.4 Embedding patches

After extracting patches, i.e., WUL subgraphs of the 1-hop neighborhoods, it still remains to localize each patch in its own frame. Under the assumptions of the geometric graph model, it is likely that 1-hop neighbors of the central node will also be interconnected, rendering a relatively high density of edges for the patches. Indeed, as indicated by Figure 3.8 (right panel), most patches have at least half of the edges present. For noiseless distances, we embed the patches using the FULL-SDP algorithm [17], while for noisy distances we use the SNL-SDP algorithm of [101]. To improve the overall localization result, the SDP solution is used as a starting point for a gradient-descent method.

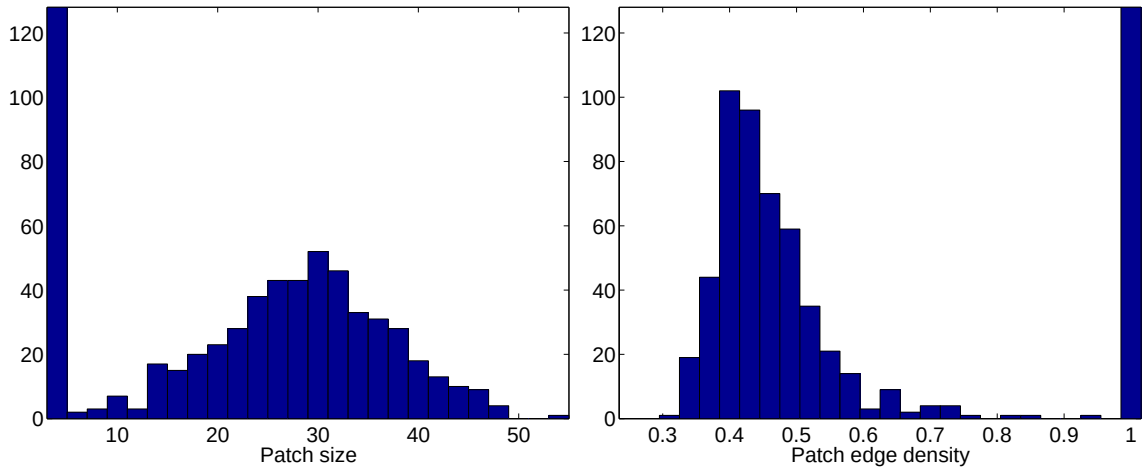


Figure 3.8: Histogram of patch sizes (left) and edge density (right) in the BRIDGE-DONUT graph, $n = 500$ and $deg = 18$. Note that a large number of the resulting patches are of size 4, thus complete graphs on four nodes (K_4), which explains the same large number of patches with edge density 1.

3.5.5 Additional information specific to the molecule problem

In this section we discuss several additional constraints specific to the molecule problem, which are currently not being exploited by 3D-ASAP. While our algorithm can benefit from any existing molecular fragments and their known reflection, there is still information that it does not take advantage of, and which can further improve its performance. Note that many of the remarks below can be incorporated in the pre-processing step of embedding the patches, described in the previous section.

The most important piece of information missing from our 3D-ASAP formulation is the distinction between the "good" edges (bond lengths) and the "bad" edges (noisy NOEs). The current implementations of the FULL-SDP and SNL-SDP algorithms do not incorporate such hard distance constraints.

One other important information which we are ignoring is given by the residual dipolar couplings (RDC) measurements that give noisy angle information ($cos^2(\theta)$) with respect to a global orientation [10].

Another approach is to consider an energy based formulation that captures the interaction between atoms in a readily computable fashion, such as the Lennard-Jones potential. One may then use this information to better localize the patches, and prefer patches that have lower energy.

The minimum distance constraint, also referred to as the "hard sphere" constraint, comes from the fact that any two atoms cannot get closer than a certain distance $\kappa \approx 1$ Angstrom. Note that such lower bounds on distances can be easily incorporated into the SDP formulation.

Another observation one can make use of is set of non-edges of the measurement graph, i.e., the distances corresponding to the missing edges cannot be smaller than the sensing radius ρ. Two remarks are in place however; under the current noise model it is possible for true distances smaller than the sensing radius not to be part of the set of available measurements, and vice-versa, it is possible for true distances larger than the sensing radius to become part of the distance set. However, since this constraint is not as certain as the hard sphere constraint, we recommend using the latter one.

Finally, one can envisage that significant other information can be reduced to distance constraints and incorporated into the approach described here for the calculation of structures and complexes. Such development could significantly speed such calculations if it incorporates larger molecular fragments based on modeling, similarly of chemical shift data etc., as done with computationally intensive experimental energy methods, e.g., HADDOCK [107].

3.5.6 Aligning patches

Given two patches P_k and P_l that have at least four nodes in common, the registration process finds the optimal 3D rigid motion of P_l that aligns the common points. A pictorial illustration of the registration process is shown in Figure 2.3 of the previous chapter. A closed form solution to the registration problem in any dimension was given in [59], where the best rigid transformation between two sets of points is obtained by various matrix ma-

nipulations and eigenvalue/eigenvector decomposition.

Since alignment requires at least four overlapping nodes, the K4 patches that are fully contained in larger patches are initially discarded. Other patches may also be discarded if they do not intersect any other patch in at least four nodes. The nodes belonging to such patches but not to any other patch would not be localized by ASAP.

As expected, in the case of the geometric graph model, the overlap is often small, especially for small values of ρ. It is therefore crucial to have robust alignment methods even when the overlap size is small. We refer the reader to Section 2.5 of the previous chapter for other methods of aligning patches with fewer common nodes in $\mathbb{R}^2$, i.e. the combinatorial method and the link method which can be adjusted for the three dimensional case. The combinatorial score method makes use of the underlying assumption of the geometric graph model. Specifically, we exploit the information in the non-edges that correspond to distances larger than the sensing radius ρ, and use this information for estimating both the relative reflection and rotation for a pair of patches that overlap in just three nodes (or more). The link method is useful whenever two patches have a small overlap, but there exist many cross edges in the measurement graph that connect the two patches. Suppose the two patches P_k and P_l overlap in at least one vertex, and call a *link edge* an edge $(u, v) \in E$ that connects a vertex u in patch P_k (but not in P_l) with a vertex v in patch P_l (but not in P_k). Such link edges can be incorporated as additional information (besides the common nodes) into the registration problem that finds the best alignment between a pair of patches. The right plot in Figure 3.9 shows a histogram of the intersection sizes between patches in the BRIDGE-DONUT graph that overlap in at least 4 nodes.

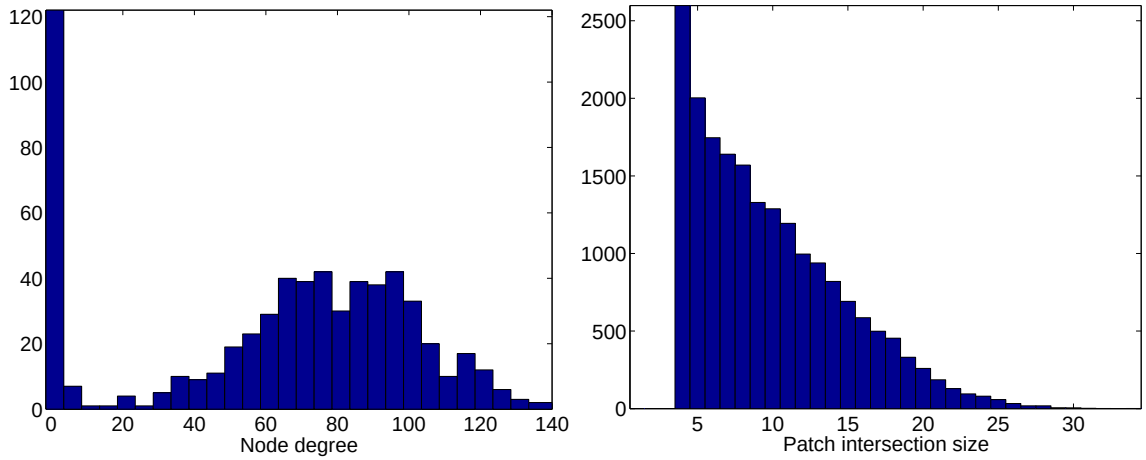


Figure 3.9: Histogram of the node degrees of patches in the patch graph G^P (left) and the intersection size of patches (right), in the BRIDGE-DONUT graph with $n = 500$ and $deg = 18$. G^P has $N = 615$ nodes (i.e. patches) and average degree 24, meaning that, on average, a patch overlaps (in at least 4 nodes) with 24 other patches.

3.6 Spectral-Partitioning-ASAP (3D-SP-ASAP)

In this section we introduce 3D-Spectral-Partitioning-ASAP (3D-SP-ASAP), a variation of the 3D-ASAP algorithm, which uses spectral partitioning as a preprocessing step for the localization process.

3D-SP-ASAP combines ideas from both DISCO [74] and 3D-ASAP. The philosophy behind DISCO is to recursively divide large problems into smaller problems, which can ultimately be solved by the traditional SDP-based localization methods. If the number of atoms in the current group is not too large, DISCO solves the atom positions via SDP, and refines the coordinates by using gradient descent; otherwise, it breaks the current group of atoms into smaller subgroups, solves each subgroup recursively, aligns and combines them together, and finally it improves the coordinates by applying gradient descent. The main question that arises is how to divide a given problem into smaller subproblems. DISCO chooses to divide a large group of nodes into exactly two subproblems, solves each problem recursively and combines the two solutions. In other words, it builds a binary tree of problems, where the leaves are problems small enough to be embedded by SDP. However, not all available information is being used when considering only a single spanning tree of the graph of patches. The 3D-ASAP approach fuses information from different spanning trees via the eigenvector computation. However, compared to the number of patches used in DISCO, 3D-ASAP generates many more patches, since the number of patches in 3D-ASAP is linear in the size of the network. This can be considered as a disadvantage, since localizing all the patches is often the most time consuming step of the algorithm. 3D-SP-ASAP tries to reduce the number of patches to be localized while using the patch graph connectivity in its full.

When dividing a graph into two smaller subgraphs, one wishes to minimize the number of edges between the two subgraphs, since in the localization of the two subgraphs the edges across are being left out. Simultaneously, one wishes to maximize the number of edges within the subgraphs, because this makes the subgraphs more likely to be globally rigid and easier to localize. In general, the graph partitioning problem seeks to decompose a graph into K disjoint subgraphs (clusters), while minimizing the number of cut edges, i.e., edges with endpoints in different clusters. Given the number of clusters K, the Min-Cut problem is an optimization problem that computes a partition $\mathcal{P}_1, \ldots, \mathcal{P}_K$ of the vertex set, by minimizing the number of cut edges

$$\text{Cut}(\mathcal{P}_1, \ldots, \mathcal{P}_K) = \sum_{i=1}^{K} E(\mathcal{P}_i, \overline{\mathcal{P}_i}), \tag{3.19}$$

where $E(X, Y) = \sum_{i \in X, j \in Y} A_{ij}$, and $\overline{X}$ denotes the complement of X. However, it is well known that trying to minimize $\text{Cut}(\mathcal{P}_1, \ldots, \mathcal{P}_K)$ favors cutting off weakly connected individual vertices from the graph, which leads to poor partitioning quality since we would like clusters to consist of a relatively large number of nodes. To penalize clusters $\mathcal{P}_i$ of small size, Shi and Malik [90] suggested minimizing the normalized cut defined as

$$\text{NCut}(\mathcal{P}_1, \ldots, \mathcal{P}_K) = \sum_{i=1}^{K} \frac{\text{Cut}(\mathcal{P}_i, \overline{\mathcal{P}_i})}{\text{Vol}(\mathcal{P}_i)}, \tag{3.20}$$

where $\text{Vol}(P_i) = \sum_{i \in P_i} deg(i)$, and $deg(i)$ denotes the degree of node i in G.

Although minimizing NCut over all possible partitions of the vertex set V is an NP-hard combinatorial optimization problem [108], there exists a spectral relaxation that can be computed efficiently [90]. We use this spectral clustering method to partition the measurement graph in the molecule problem. The gist of the approach is to use the classical K-means clustering algorithm on the Laplacian eigenmap embedding of the set of nodes. If A is the adjacency matrix of the graph G, and D is a diagonal matrix with $D_{i,i} = deg(i), i = 1, \ldots, n$, then the Laplacian eigenmap embedding of node i in $\mathbb{R}^k$ is given by $(\phi_1(i), \phi_2(i), \ldots, \phi_K(i))$, where ϕ_j is the j^{th} eigenvector of the matrix $D^{-1}A$. For an extensive literature survey on spectral clustering algorithms we refer the reader to [106]. We remark that other clustering algorithms (e.g., [57]) may also be used to partition the graph.

The approach we used for localization in conjunction with the above normalized spectral clustering algorithm is as follows (3D-SP-ASAP):

1. We first decompose the measurement graph into K partitions $\mathcal{P}_1, \ldots, \mathcal{P}_K$, using the normalized spectral clustering algorithm.

2. We extend each partition $\mathcal{P}_i$, $i = 1, \ldots, K$ to include its 1-hop neighborhood, and denote the new patches by P_i, $i = 1, \ldots, K$.

3. For every pair of patches P_i and P_j which have nodes in common or are connected by link edges [13], we build a new (link) patch which contains all the common points and link edges. The vertex set of the new patch consists of the nodes that are common to both P_i and P_j, together with the endpoints of the link edges that span across the two patches. Note that the new list of patches contains the extended patches built in Step (2) $P_1, \ldots, P_K$, as well as the newly built patches $P_{K+1}, \ldots, P_L$.

4. We extract from each patch P_i, $i = 1, \ldots, L$ the WUL subgraph, and embed it using the FULL-SDP algorithm for noiseless data, and the SNL-SDP algorithm for noisy data.

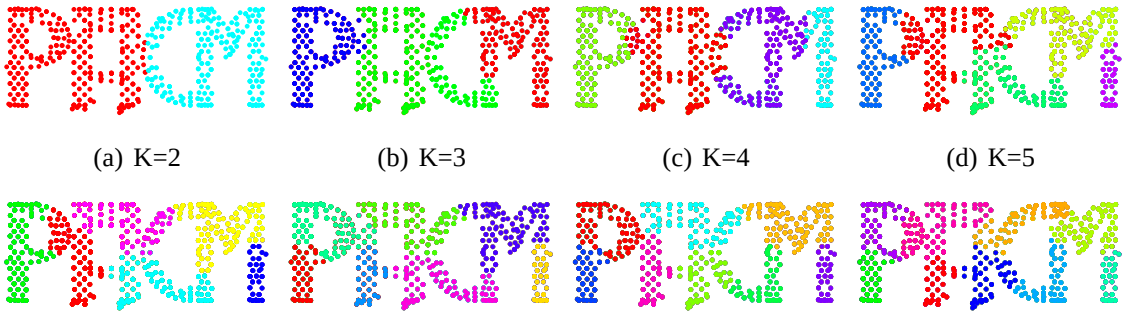
5. Synchronize all available patches using the eigenvector synchronization algorithm used in ASAP.

Note that when the extended patches from Step (2) are highly overlapping, Step (3) of the algorithm should be omitted, for reasons detailed below related to the robustness of the embedding. The reason for having Steps (2), and possibly (3), is to be able to align nearby patches. Without Step (2) patches will have disjoint sets of nodes, and the alignment will be based only on the link edges, which is not robust for high levels of noise. Without Step (3) the existing patches may have little overlap, in which case we expect the alignment involving link edges not to be robust at high levels of noise. By building the link patches, we provide 3D-SP-ASAP more accurate pairwise alignments. Note that embedding link patches is less robust to noise, due to their bipartite-like structure, especially when the bipartitions are very loosely connected to each other (also confirmed

[13] edges with endpoints in different patches

by our computations involving link patches). In addition, having to localize a larger number of patches may significantly increase the running time of the algorithm. However, for our numerical experiments with 3D-SP-ASAP conducted on the BRIDGE-DONUT graph, the extended partitions built in Step (2) were highly overlapping, and allowed us to localize the entire network without the need to build the link patches in Step (3).

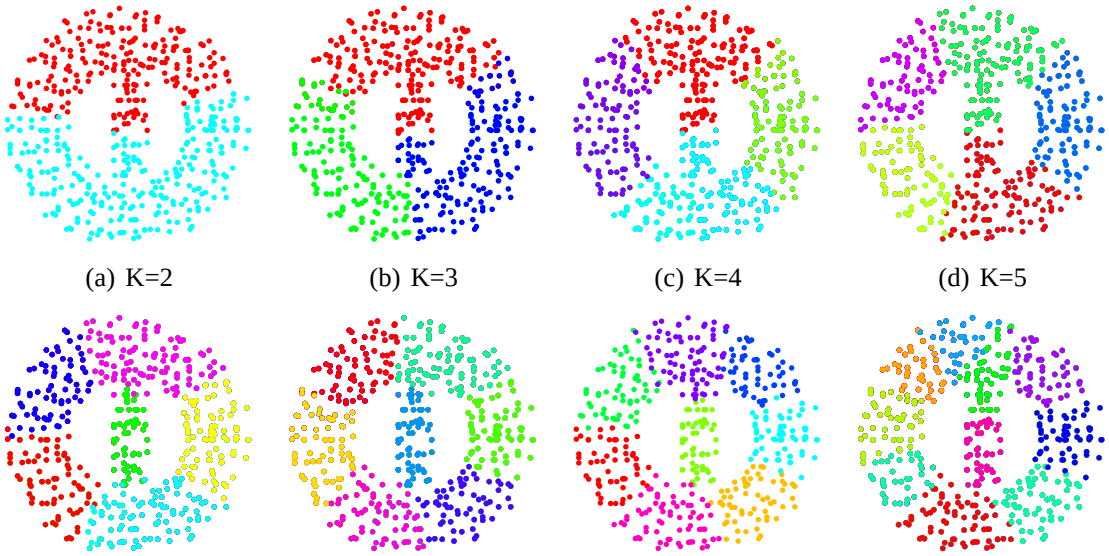
The advantage of combining a spectral partitioning algorithm with 3D-ASAP is a decrease in running time as shown in Tables 3.13 and 3.14, due to a significantly smaller number of patches that need to be localized. Note that the graph partitioning algorithm is extremely fast, and partitions the BRIDGE-DONUT measurement graph in less than half a second. Table 3.11 shows the ANE reconstruction errors for the BRIDGE-DONUT graph, when we partition the measurement graph into $K = 8$ and $K = 25$ clusters. For $K = 8$, some of the extended partitions become very large, containing as many as 150 nodes, and SNL-SDP does a very poor job at embedding such large patches when the distance measurements are noisy. By increasing the number of partitions to $K = 25$, the extended partitions contain less nodes, and are small enough for SNL-SDP to localize accurately even for high levels of noise. Note that the ANE errors for 3D-SP-ASAP with $K = 25$ are comparable with those of ASAP, while the running time is dramatically reduced (by an order of magnitude, for the BRIDGE-DONUT example with $\eta = 35\%$). Figures 3.10 and 3.11 show various K-partitions of the PACM and the BRIDGE-DONUT graphs.



(a) K=2 (b) K=3 (c) K=4 (d) K=5

(e) K=6 (f) K=7 (g) K=8 (h) K=9

Figure 3.10: Partitions of the PACM graph (K is the number of partitions)



(a) K=2 (b) K=3 (c) K=4 (d) K=5

(e) K=6 (f) K=7 (g) K=8 (h) K=9

Figure 3.11: Partitions of the BRIDGE-DONUT graph (K is the number of partitions)

3.7 The Median Denoising Algorithm (MDA) and a rescaling heuristic

In this section we describe a method to improve the localization of the individual patches prior to the patch alignment and global synchronization steps. The main observation here is the following: suppose a pair of nodes appear in several different patches, then each patch provides a different estimation for their distance, and all these estimators can be averaged together to provide a perhaps more accurate estimator. The improved reestimated distances are then used to localize the individual patches again, this time using the cMDS algorithm since patches no longer contain any missing distances. The second part of this section introduces a simple rescaling procedure, that increases the accuracy of the final reconstruction.

Denote by C_k ($k = 1, \ldots, N$) the set of all pairwise edges within patch P_k (so if P_k has t nodes then C_k contains all $\binom{t}{2}$ edges). Let $\mathcal{C}$ denote the set of all edges that appear in at least one patch, i.e., $\mathcal{C} = \cup_{k=1}^{N} C_k$. Note that $\mathcal{C}$ contains edges (i, j) that were initially missing from the measurement set (i.e. $(i, j) \notin E$) but for which we now have an estimate due to the initial localization of the patches (using, say, SDP). After the initial embedding, we have at least one estimated distance for each edge $(i, j) \in \mathcal{C}$, but because patches overlap, most of the edges appear in more than one patch. Denote by $d_{ij}^1, d_{ij}^2, \ldots$ the set of estimates for the (possibly missing) distance (i, j), where d_{ij}^k is the distance between nodes

i and j in the SDP embedding of patch P_k. To obtain a more accurate estimate of d_{ij} we take the median of all the above estimates, and denote the updated value by

$$\widetilde{d}_{ij} = \text{median}(d_{ij}^1, d_{ij}^2, \ldots) \tag{3.21}$$

If the noise level in the originally measured distances is small, then we expect the estimates $d_{ij}^1, d_{ij}^2, \ldots$ to have small variance and take values close to the true distance. However, for higher levels of noise, many of the estimates can be very inaccurate (either highly overestimating or underestimating the true distances), and we choose the median value to approximate the true distance. Tables 3.5 and 3.6 show that MDA is indeed effective and reduces the noise level in the distances, usually by at least a few percentages. Note that the SDP improves by itself on the initial distance measurements (those that are available), and the MDA decreases their noise even further.

To take advantage of the more accurate updated distance values, we recompute the embeddings of all N patches using the cMDS algorithm, since there exists an estimate for all pairwise distances within a patch. Whenever a hard constrained distance (a "good" edge) is present, such as a bond length, we use its true distance in the cMDS embeddings, and ignore the noisy value returned by the SDP since hard constrained distances are not enforced in the SDP.

Tables 3.5 and 3.6 show the average ANE of the patches after the SDP embeddings ($\overline{ANE}_{old}$), and after the cMDS embeddings ran on the updated distances ($\overline{ANE}_{new}$). Note that the new patch embeddings are significantly more accurate, in some cases the ANE decreasing by as much as 25%. We also experimented with an iterative version of this denoising algorithm, where at each round we run the cMDS and recompute the median of the updated distances. However, subsequent iterations of this procedure did not improve the accuracy furthermore.

			Existing distances			Missing distances	
η	$\overline{ANE}_{old}$	$\overline{ANE}_{new}$	$\bar{\eta}$	$\bar{\eta}_{SDP}$	$\bar{\eta}_{MDA}$	$\bar{\eta}_{SDP}$	$\bar{\eta}_{MDA}$
10%	0.04	0.03	5.06	3.75	3.42	3.37	3.13
20%	0.10	0.078	10.28	8.22	7.5	8.91	8.12
30%	0.18	0.14	15.95	14.09	13.03	17.99	16.83
40%	0.27	0.22	21.74	20.66	19.31	29.09	27.72
50%	0.37	0.32	29.11	28.76	27.5	42.14	40.92

Table 3.5: Denoising distances at various levels of noise, for the UNITCUBE data set. $\overline{ANE}_{old}$ and $\overline{ANE}_{new}$ denote the average ANE of all patches after the SDP embedding respectively after the cMDS embedding with the distances denoised by MDA. $\bar{\eta}$ is the empirical noise level (averaged over all noisy distances), $\bar{\eta}_{SDP}$ is the noise level after the SDP embedding, $\bar{\eta}_{MDA}$ is the noise level after running MDA. We show the noise levels individually for both the existing and missing distances. Note that $\bar{\eta} \neq \eta$ since the former denotes the average absolute value deviation from the true distance as a percentage of the true distance, while the latter is the parameter in the uniform distribution in (2.28) that is used to define the noise model (in fact, $\mathbb{E}[\bar{\eta}] = \frac{\eta}{2}$).

η	$\overline{ANE}_{old}$	$\overline{ANE}_{new}$	Existing distances				Missing distances	
			$\bar{\eta}$	$\bar{\eta}_{SDP}$	$\bar{\eta}_{SDP,GOOD}$	$\bar{\eta}_{MDA}$	$\bar{\eta}_{SDP}$	$\bar{\eta}_{MDA}$
0%	0.00048	0.00039	0	0.02	0.14	0	0.38	0.32
10%	0.00316	0.00266	4.91	3.42	1.79	3.13	3.2	2.99
20%	0.00590	0.00506	9.81	7.01	3.68	6.44	6.18	5.81
30%	0.00957	0.00803	14.74	10.75	5.59	9.88	9.51	8.95
40%	0.01292	0.01120	19.65	14.5	7.62	13.42	13.55	12.8
50%	0.01639	0.01462	24.57	18.1	9.82	16.93	18.24	17.35

Table 3.6: Denoising distances at various levels of noise for the ubiqutin (PDG 1d3z, [29]). $\bar{\eta}_{SDP,GOOD}$ denotes the average noise of the hard constraint ("good") distances after the SDP embeddings. Note that the current implementation of the SNL-SDP algorithm used for the patch embeddings does not allow for hard distance constraints.

In the remaining part of this section, we discuss the issue of scaling of the distances after different steps of the algorithm, and propose a simple rescaling heuristic that improves the overall reconstruction.

We denote the true distance measurements by $l_{ij} = \|p_i - p_j\|, (i,j) \in E$, and the empirical measurements by $d_{ij} = l_{ij} + \epsilon_{ij}, (i,j) \in E$, where ϵ_{ij} is random independent noise, as introduced in (2.28). We generically refer to $\tilde{d}_{ij}$ as the distance between nodes i and j, after different steps of the algorithm, as indicated by the columns of Tables 3.7 and 3.8. We denote by δ the average scaling of the distances with respect to their ground truth values, and similarly by κ the average empirical noise of the distance measurements

$$\delta = \text{mean}\left(\frac{l_{ij}}{\tilde{d}_{ij}}, (i,j) \in E\right), \qquad \kappa = \text{mean}\left(\left|\frac{\tilde{d}_{ij} - l_{ij}}{l_{ij}}\right|, (i,j) \in E\right)$$

The first column of the tables contains the scaling and noise values of the initial distance measurements, taken as input by ASAP. The fact that κ is approximately half the value of the noise level η stems from the fact that the measurements are uniformly distributed in $[(1-\eta)l_{ij}, (1+\eta)l_{ij}]$. Note that the initial scaling is quite significant, and this is a consequence of the same noise model and the geometric graph assumption. Distances that are scaled down by the noise will still be available to 3D-ASAP, while distances that are scaled up and become larger than the sensing radius will be ignored. Therefore, on average, the empirical distances are scaled down and δ is greater than 1.

The second column gives the scaling and noise values after the MDA algorithm followed by cMDS to recompute the patches. Note that after this denoising heuristic, while the scaling δ remains very similar to the original values, the noise κ decreases considerably especially for the lower levels of noise.

The third column computes the scaling, noise and ANE values after the least squares step for estimating the translations, where by ANE we mean the average normalized error introduced in (2.29). Note that both the scaling and the noise levels significantly increase after integrating all patches in a globally consistent framework. To correct the scaling issue, we scale up all the available distances (thus taking into account only the edges of the initial

measurement graph) by δ^*, computed as

$$\delta^* = \text{mean}\left(\frac{d_{ij}}{\tilde{d}_{ij}^{LS}}, (i,j) \in E\right)$$

where d_{ij} denotes the initial distances, and $\tilde{d}_{ij}^{LS}$ the distances after the least-square step. Note that, as a consequence, the ANE error of the reconstruction decreases significantly.

Finally, we refine the solution by running gradient descent, on the initial distances d_{ij} scaled up by δ^*, which further improves the scaling, noise and ANE values.

	Original		MDA-cMDS		Least squares			Rescaled		Gradient Descent		
$\eta\%$	δ	$\kappa\%$	δ	$\kappa\%$	δ	$\kappa\%$	ANE	δ^*	ANE	δ	$\kappa\%$	ANE
10	1.01	5.1	1.01	3.4	1.02	3.6	0.05	1.01	0.05	1.00	3.0	0.04
20	1.05	10.3	1.04	7.5	1.08	8.8	0.12	1.04	0.10	1.02	6.1	0.07
30	1.12	16.0	1.11	13.0	1.21	17.0	0.25	1.09	0.20	1.04	9.9	0.16
40	1.22	21.8	1.21	19.3	1.43	27.8	0.36	1.21	0.26	1.04	13.9	0.19
45	1.29	26.0	1.28	23.8	1.65	35.6	0.45	1.32	0.30	1.02	17.8	0.26
50	1.38	29.2	1.36	27.5	1.92	43.8	0.53	1.47	0.35	0.99	22.4	0.32

Table 3.7: Average scaling and noise of the existing edges (and where appropriate the ANE of the reconstruction) for the UNITCUBE graph, at various steps of the algorithm: after the original measurements, after the SDP-MDA-cMDS denoising step, after the least squares in Step 3, after rescaling of the available distances by the empirical scaling factor δ^*, and after running gradient descent on the rescaled (originally available) distances.

	Original		MDA-cMDS		Least squares			Rescaled		Gradient Descent		
$\eta\%$	δ	$\kappa\%$	δ	$\kappa\%$	δ	$\kappa\%$	ANE	δ^*	ANE	δ	$\kappa\%$	ANE
10	1.00	0.03	1.00	0.02	1.02	0.04	0.01	1.02	4e-3	0.99	0.01	3e-3
20	1.01	0.06	1.00	0.04	1.04	0.07	0.01	1.03	0.01	0.99	0.02	0.01
30	1.02	0.08	1.01	0.06	1.06	0.10	0.02	1.05	0.01	0.99	0.03	0.01
40	1.03	0.11	1.01	0.08	1.10	0.14	0.02	1.09	0.02	0.99	0.05	0.01
50	1.06	0.14	1.02	0.10	1.18	0.19	0.04	1.16	0.03	0.98	0.08	0.01

Table 3.8: Average scaling and noise of all edges (good+NOE) of the ubiqutin (PDG 1d3z) (and where appropriate the ANE of the reconstruction) at various steps of the algorithm: after the original measurements, after the SDP-MDA-cMDS denoising step, after the least squares in Step 3, after rescaling of the available distances by the empirical scaling factor δ^*, and after running gradient descent on the rescaled NOE distances.

3.8 Experimental results

We have implemented our 3D-ASAP algorithm and compared its performance with other methods across a variety of measurement graphs, varying parameters such as the number of nodes, average degree (sensing radius) and level of noise. The experimental setup is similar to the one used in Section 2.7 for 2D-ASAP, where we used multiplicative and uniform noise, as detailed in equation 2.28.

The sizes of the graphs we experimented with range from 200 to 1000 nodes taking different shapes, with average degrees in the range $14-26$, and noise levels up to 50%. Across all our simulations, we consider the geometric graph model, meaning that all pairs of sensors within range ρ are connected. We denote the true coordinates of all sensors by the $2\times n$ matrix $P = (p_1 \ \cdots \ p_n)$, and the estimated coordinates by the matrix $\hat{P} = (\hat{p}_1 \ \cdots \ \hat{p}_n)$. To measure the localization error of our algorithm we first factor out the optimal rigid transformation between the true embedding P and our reconstruction $\hat{P}$, and then compute the average normalized error (ANE), a scale invariant measure for error introduced for 2D-ASAP in Section 2.29.

The experimental results in the case of noiseless data (i.e. incomplete set of exact distances) should already give the reader an appreciation of the difficulty of the problem. As mentioned before, the graph realization problem is NP-hard, and the most we can aim for is an approximate solution. The main advantage of introducing the notion of a weakly uniquely localizable graph and using it for breaking the original problem into smaller subproblems, can be seen in the experimental results for noiseless data. Note that across all experiments, except for the PACM graph reconstructions, we are able to compute localizations which are at least one order of magnitude more accurate than what those computed by DISCO or SNL-SDP. Note that all algorithms are followed by a refinement procedure, in particular steepest descent with backtracking line search.

η	3D-ASAP	DISCO	SNL-SDP	MVU
0%	5e-4	8e-3	2e-3	2e-6
10%	0.04	0.06	0.04	0.07
20%	0.07	0.10	0.09	0.12
30%	0.16	0.17	0.17	0.26
40%	0.19	0.29	0.29	0.50
45%	0.26	0.34	0.35	0.48
50%	0.32	0.42	0.43	0.61

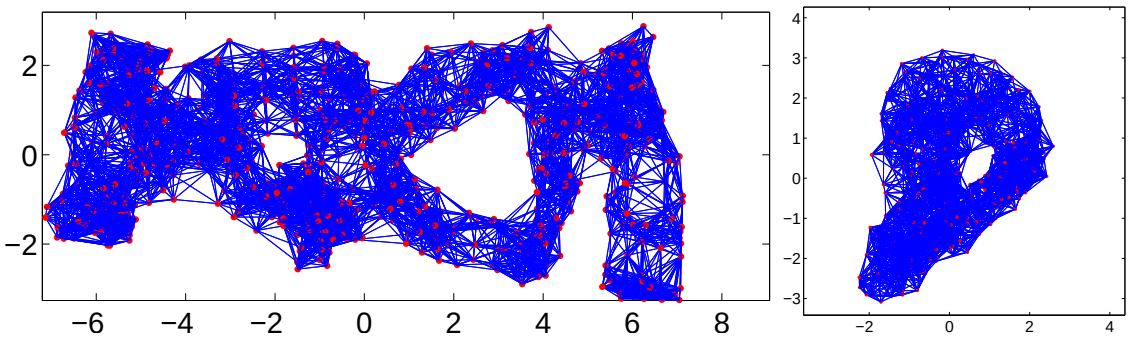
Table 3.9: Reconstruction errors (measured in ANE) for the UNITCUBE graphs with $n = 212$ vertices, sensing radius $\rho = 0.3$ and average degrees $deg = 17 - 25$.

η	3D-ASAP	DISCO
0 %	0.0001	0.0024
10 %	0.0031	0.0029
20 %	0.0052	0.0058
30 %	0.0069	0.0078
35 %	0.0130	0.0104
40 %	0.0094	0.0107
45 %	0.0168	0.0207
50 %	0.0146	0.0151

Table 3.10: Reconstruction errors (measured in ANE) for the 1d3z molecule graph with $n = 1009$ vertices, sensing radius $\rho = 5$ angstrom and average degree $deg = 14$.

As expected, FAST-MVU performs at its best when the data is a set of random points uniformly distributed in the unit cube. In this case, the top three eigenvectors of the Lapla-

cian matrix used by FAST-MVU provide an excellent approximation of the original coordinates. Indeed, as the experimental results in Table 3.9 show, the FAST-MVU algorithm gives the best precision across all algorithms, returning an $ANE = 2e-6$. However, in the case of noisy data, the performance of FAST-MVU degrades significantly, making FAST-MVU the least robust to noise algorithm tested on the UNITCUBE graph. Furthermore, FAST-MVU performs extremely poor on more complicated topologies, even in the absence of noise, as it can be seen in the reconstructions of the PACM and BRIDGE-DONUT graphs shown in Figure 3.12. For comparison, the original underlying graphs are shown in Figures 3.13 and 3.14.



(a) PACM graph with noise $\eta = 0\%$

(b) DONUT graph with noise $\eta = 0\%$

Figure 3.12: Reconstructions of the PACM and BRIDGE-DONUT graphs using the FAST-MVU algorithm, in the case of noiseless data (compare to the original measurement graphs shown in Figures 3.13 and 3.14.)

The second data set used in our experiments is a synthetic example of the molecule problem, and represents the ubiqutin protein represented by PDB entry 1d3z [29], with 1288 atoms, 686 of which are hydrogen atoms. However, after removing the nodes of degree less than 4, $n = 1009$ atoms remain. For this data set, the available distances can be divided into groups: the first group contains distances inferred from known bond length and torsion angles ("good" edges) which are kept accurate across different levels of noise, and the second group contains distances (NOE edges) which are perturbed with noise η at each stage of the experiment. On average, each node is incident with 6 good edges and 8 NOE edges, thus the average degree of the entire measurement graph is $deg = 14$. In addition, besides knowledge of good and NOE edges, we also use information from molecular fragments in the form of subgroups of atoms (molecular fragments) whose coordinates are readily available. We incorporate 280 such molecular fragments, whose average size is close to 5 atoms. To exploit such information, we slightly modify the approach to break up the measurement graph and synchronize the patches. We build patches from molecular fragments by selecting a WUL subgraph from each extended molecular fragment, where by an extended molecular fragment we mean the graph resulting from a molecular fragment and all its 1-hop neighbors. Once we extract patches from all extended molecular fragments, we consider all remaining nodes (singletons) that are not contained in any of the patches obtained so far. Depending on the noise level, the number of such singleton nodes

is between 10 and 15. Finally, we extract WUL patches from the 1-hop neighborhoods of the singleton nodes, following the approach of Section 3.5. Since we know a priori the localization, and in particular the reflection of each molecular fragment, and the reflection of a molecular fragment induces a reflection on the patch that contains it, we have readily available information on the reflection of all patches that contain molecular fragments. In other words, we are solving a synchronization problem over $\mathbb{Z}_2$ when molecular fragment information is available. Using the synchronization method introduced in Section 4.4.1, we compute the reflection of the remaining patches. In terms of the reconstruction error, 3D-ASAP and DISCO return comparable results, slightly in favor of our algorithm. For noiseless NOE distances, we compute a localization that is one order of magnitude more accurate than the solution returned by DISCO.

Another graph that we tested is the BRIDGE-DONUT graph shown in Figure 3.14. It has $n = 500$ nodes, sensing radius $\rho = 0.92$, and average degree in the range $18 - 25$, depending on the noise level. Table 3.11 contains the reconstruction errors for various levels of noise showing that 3D-ASAP consistently returns more accurate solutions than DISCO. For the BRIDGE-DONUT graph, we included in our numerical simulations the spectral partitioning 3D-SP-ASAP algorithm, which performed just as well as the 3D-ASAP algorithm (while significantly reducing the running time as shown in Tables 3.14 and 3.13), and consistently outperformed the DISCO algorihtm at all levels of noise. Note that 3D-SP-ASAP returns very poor localizations when using only $k = 8$ partitions; in this case the extended partitions have large size (up to 150 nodes) and SNL-SDP fails to embed them accurately, even at small levels of noise, as shown in Table 3.11. By increasing the number of partitions from $k = 8$ to $k = 25$ and thus decreasing the size of each partition (and also of the extended partitions), the running time of 3D-SP-ASAP increases slightly from 140 to 186 seconds (at 35% noise), but we are now able to match the accuracy of 3D-ASAP which requires almost ten times more running time (1770 seconds).

Finally, for the PACM graphs in Figure 3.13, the network takes the shape of the letters P, A, C, M that form a connected graph on $n = 800$ vertices. The sensing radius is $\rho = 1.2$ and the average degree in the range $deg \approx 21 - 26$. This graph was particularly useful in testing the sensitivity of the algorithm to the topology of the network. In Table 3.12, we show the reconstruction errors for various levels of noise DISCO returns better results for $\eta = 0\%, 10\%, 40\%$, but less accurate for $\eta = 20\%, 30\%, 35\%$. We believe that DISCO returns results comparable to 3D-ASAP (unlike the previous three graphs) because the topology of the PACM graph favors the graph decomposition used by DISCO. Note that at $\eta = 0\%$ noise, the 3D-ASAP reconstruction differs from the original embedding mainly in the right handside of the letter M, which is loosely connected to the rest of the letter, and also parts of its underlying graph are very sparse, thus rendering the SDP localization algorithms less accurate. This can also be seen in the spectral partitioning of the PACM graph shown in Figure 3.10, where for k (number of clusters) as low as 3 or 4, the right side of letter M is picked up as an individual cluster. At higher levels of noise 3D-SP-ASAP proves to be more accurate than DISCO, and it returns results comparable with ASAP.

η	3D-ASAP	3D-SP-ASAP$_{k=8}$	3D-SP-ASAP$_{k=25}$	DISCO
0%	6e-4	4e-4	2e-4	4e-3
10%	0.04	0.59	0.03	0.03
20%	0.05	0.48	0.04	0.07
30%	0.11	0.52	0.14	0.30
35%	0.12	0.33	0.15	0.21
40%	0.24	0.42	0.19	0.27
45%	0.28	0.71	0.32	0.35
50%	0.23	0.35	0.35	0.40

Table 3.11: Reconstruction errors (measured in ANE) for the BRIDGE-DONUT graph with $n = 500$ vertices, sensing radius $\rho = 0.92$ and average degrees $deg = 18 - 25$. We used k to denote the number of partitions of the vertex set. For $\eta = 0$ we embed each of the k patches (extended partitions) using FULL-SDP, while for $\eta > 0$ we used the SNL-SDP algorithm. At $\eta = 50\%$, 3D-SP-ASAP$_{k=8}$ localizes only 435 out of the 500 nodes.

η	3D-ASAP	3D-SP-ASAP$_{k=40}$	DISCO
0%	0.05	0.05	0.02
10%	0.08	0.12	0.07
20%	0.07	0.16	0.18
30%	0.27	0.15	0.45
35%	0.18	0.32	0.28
40%	0.48	0.20	0.26

Table 3.12: Reconstruction errors (measured in ANE) for the PACM graph with $n = 800$ vertices, sensing radius $\rho = 1.2$ and average degrees $deg = 21 - 26$. For 3D-SP-ASAP we used $k = 40$ partitions. Note that even for the noiseless case the error is not negligible due to incorrect embeddings of subgraphs that are contained in the right leg of the letter M.

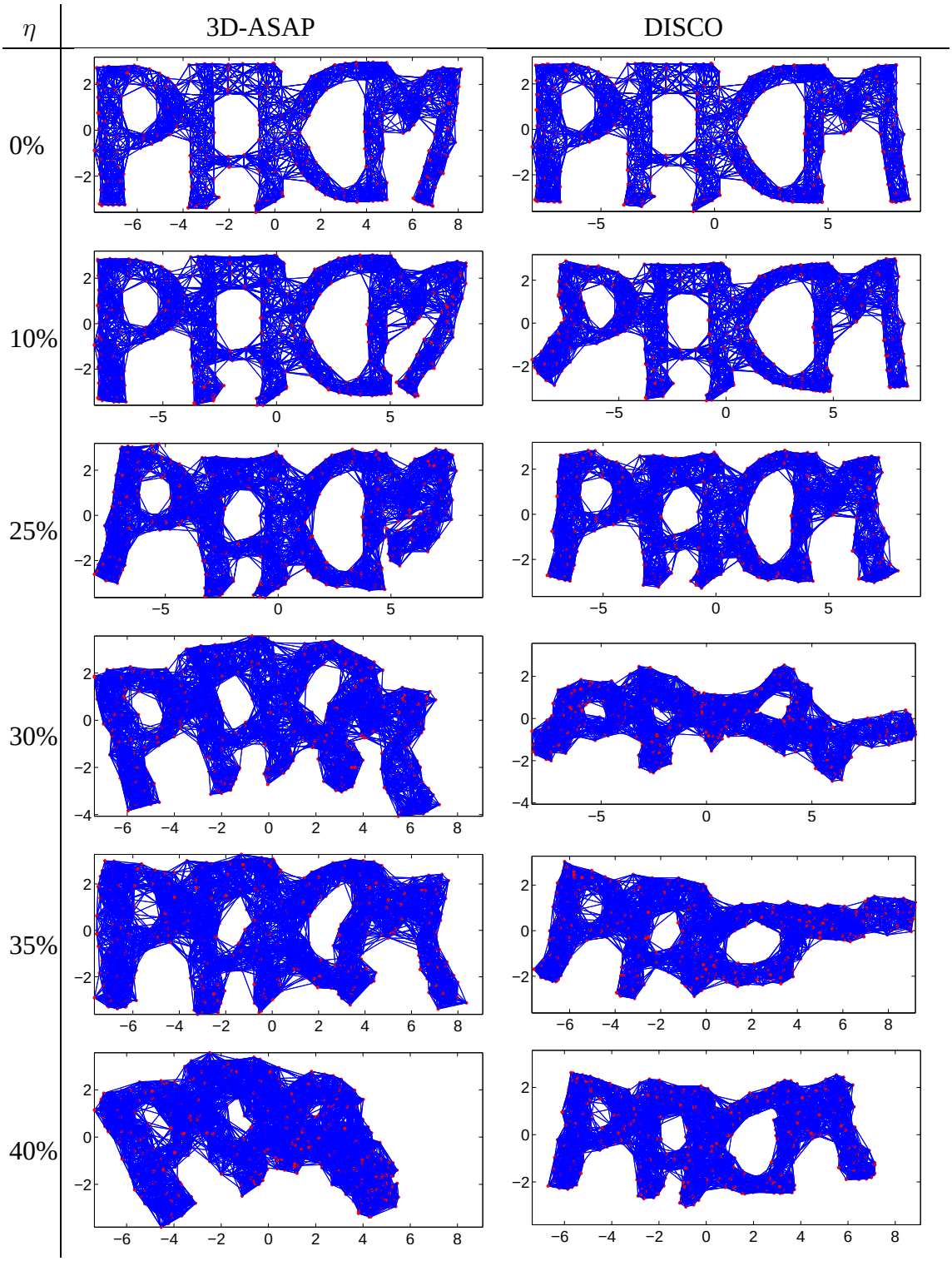


Figure 3.13: Reconstructions of the PACM graph with $n = 800$ nodes, sensing radius $\rho = 1.2$ and $\eta = 0\%, 10\%, 25\%, 30\%, 35\%, 40\%$.

Table 3.14 shows the running time of the various steps of the 3D-ASAP algorithm corresponding to our not particularly optimized MATLAB implementation. Our experimental platform was a PC machine equipped with an Intel(R) Core(TM)2 Duo CPU E8500 @ 3.16GHz 4 GB RAM. Notice that all steps are amenable to a distributed implementation, thus a parallelized implementation would significantly reduce the running times. Note the running time of 3D-ASAP is significantly larger than that of 3D-SP-ASAP and DISCO, due to the large number of patches (linear in the size of the network) that need to be localized. 3D-SP-ASAP addresses this issue, and reduces the running time from 474 to 108 seconds (for $\eta = 0\%$), and from 1770 to 186 seconds (for $\eta = 35\%$). Note that all steps of the algorithm scale linearly in the size of the network, except for the eigenvector computation, which is nearly-linear. We refer the reader to Section 7 of [32] for a complexity analysis of each step of 2D-ASAP, and remark that it is very similar to the complexity of 3D-ASAP.

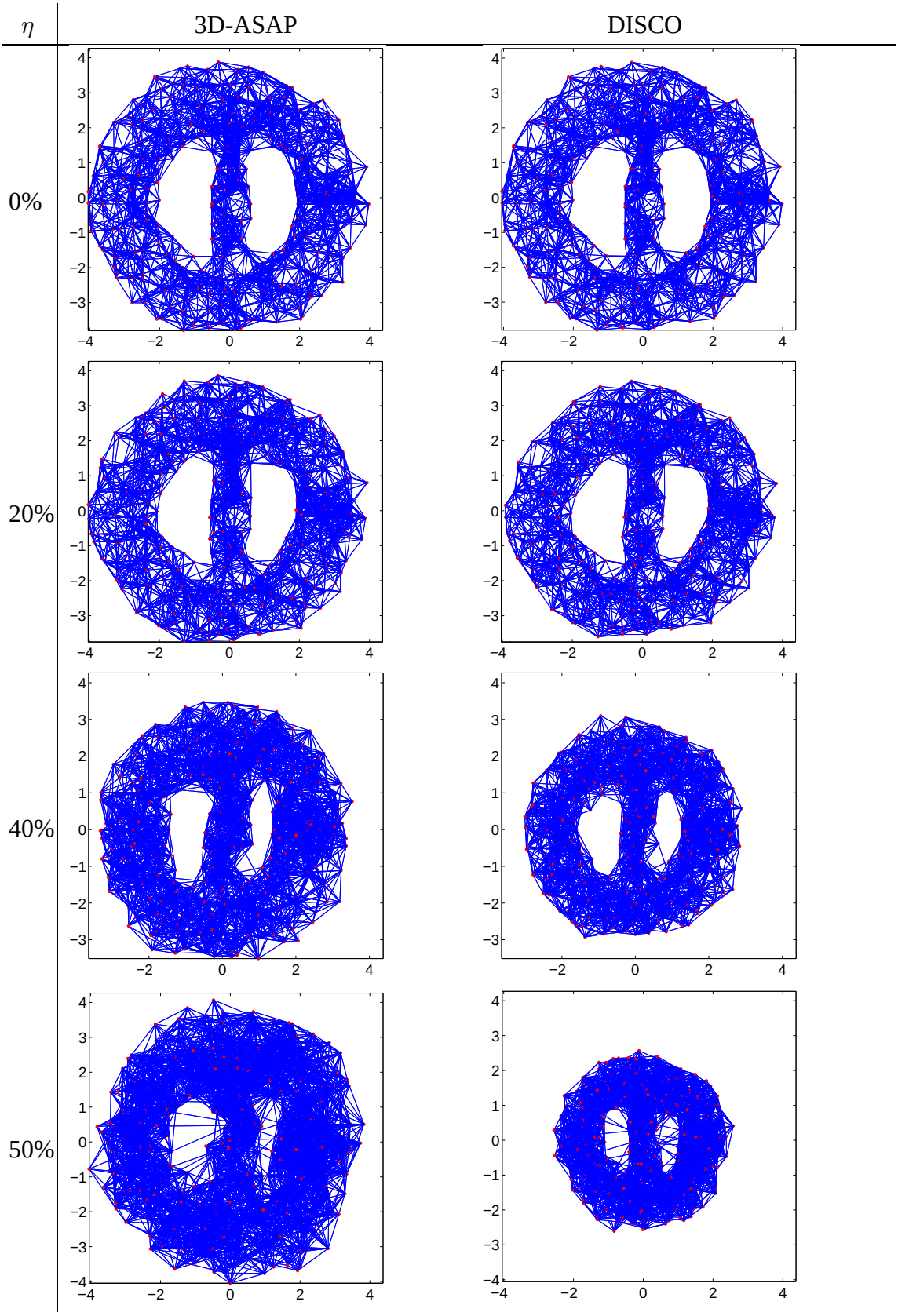


Figure 3.14: Reconstructions of the BRIDGE-DONUT graph with $n = 500$ nodes, sensing radius $\rho = 0.92$ and $\eta = 0\%, 20\%, 40\%, 50\%$.

Number of partitions k	$k = 8$		$k = 25$
Noise level η	0%	35%	35%
Spectral partitioning	0.3	0.3	0.7
Finding WUL subgraphs	48	81	89
Embedding FULL-SDP	53	(82)	(89)
Embedding SNL-SDP	(26)	50	86
Step 1 ($\mathbb{Z}_2 \times$ S0(3))	1	1	1
Step 2 (Least squares)	6	8	9
Total	108	140	186

Table 3.13: Running times (in seconds) of the 3D-SP-ASAP algorithm for the BRIDGE-DONUT graph with $n = 500$ nodes, $\eta = 0\%, 35\%$, $deg = 18, 21$, and $k = 8, 25$ partitions. For $\eta = 0\%$ we embed the patches using FULL-SDP, while for $\eta = 35\%$ we use SNL-SDP since the regularization term improves the localization.

	$\eta = 0\%$	$\eta = 35\%$
Break G into patches	59	90
Finding WUL subgraphs	210	233
Embedding FULL-SDP	154	(252)
Embedding SNL-SDP	(966)	1368
Denoising patches	30	39
Step 1 ($\mathbb{Z}_2 \times$ S0(3))	15	19
Step 2 (Least squares)	6	21
3D-ASAP	474	1770
DISCO	196	197
3D-SP-ASAP	108	186

Table 3.14: Running times (in seconds) of the 3D-ASAP algorithm for the BRIDGE-DONUT graph with $n = 500$ nodes, $\eta = 0\%, 35\%$, $deg = 18, 21$, $N = 533, 541$ patches, and average patch sizes $17.8, 20.3$. For $\eta = 0\%$ we embed the patches using FULL-SDP, while for $\eta = 35\%$ we use SNL-SDP since the regularization term improves the localization. For $\eta = 35\%$, during the least squares step we also use the scaling heuristic, and run the gradient descent algorithm several times, hence the increase in the running time from 6 to 21 seconds.

3.9 Summary and discussion

In this chapter, we introduced 3D-ASAP (As-Synchronized-As-Possible), a novel divide and conquer, non-incremental non-iterative anchor-free algorithm for solving (ab-initio) the molecule problem. In addition, we also proposed 3D-SP-ASAP, a faster version of 3D-ASAP, which uses a spectral partitioning algorithm as a preprocessing step for dividing the initial graph into smaller subgraphs. Our extensive numerical simulations show that 3D-ASAP and 3D-SP-ASAP are very robust to high levels of noise in the measured distances and to sparse connectivity in the measurement graph.

We build on the approach used in 2D-ASAP to accommodate for the additional challenges posed by rigidity theory in $\mathbb{R}^3$ as opposed to $\mathbb{R}^2$. In particular, we extract patches that are not only globally rigid, but also weakly uniquely localizable, a notion that is based on the recent unique localizability of So and Ye [96]. In addition, we also increase the robustness to noise of the algorithm by using a median-based denoising algorithm in the preprocessing step, by combining into one step the methods for computing the reflections and rotations, thus doing synchronization over O(3)= $\mathbb{Z}_2 \times$ SO(3) rather than individually over $\mathbb{Z}_2$ followed by SO(3), and finally by incorporating a scaling correction in the final step where we compute the translations of each patch by solving an overdetermined linear system by least squares. Another feature of 3D-ASAP is being able to incorporate readily available structural information on various parts of the network.

Furthermore, in terms of robustness to noise, 3D-ASAP compares favorably to some of the current state-of-the-art graph localization algorithms. The 3D-ASAP algorithm follows the same "divide and conquer" philosophy that is behind our previous 2D-ASAP algorithm, and starts with local coordinate computations based only on the 1-hop neighborhood information (of a single node, or set of nodes whose coordinates are known a priori), but unlike previous incremental methods, it synchronizes all such local information in a noise robust global optimization process using an efficient eigenvector computation. In the preprocessing step of doing the local computations, a median-based denoising algorithm improves the accuracy of the patch reconstructions, previously computed by the SDP localization algorithm.

Across almost all graphs that we have tested, 3D-ASAP constantly gives better results in terms of the averaged normalized error of the reconstruction. Except for the PACM graph, whose topology greatly favors the divide and conquer approach used by DISCO, 3D-ASAP returns reconstructions that are often significantly more accurate in the presence of large noise. Furthermore, for the case of noiseless distance measurements, the notion of weakly uniquely localizable graphs that we introduced, leads to reconstructions that are an order of magnitude more accurate than DISCO and SNL-SDP.

The geometric graph assumption, which comes up naturally in many problems of practical interest, is essential to the performance of the 3D-ASAP algorithm as it favors the existence of globally rigid or WUL patches of relatively large size. When the geometric graph assumption does not hold, the 1-hop neighborhood of a node may be extremely sparse, and thus breaking up such a sparse star graph leads to many small patches (i.e., most of them may contain only a few nodes), with only a few of them having a large enough pairwise intersection. Since small patches lead to small patch intersections, it would therefore be difficult for 3D-ASAP to align patches correctly and compute a robust final solution.

Note that in the case of random Erdős-Rényi graphs we expect the SDP methods, or even the low-rank matrix completion approaches, to work well. In other words, while these methods rely on randomness, our 3D-ASAP algorithm benefits from structure the most.

For the molecule problem, while 3D-ASAP can benefit from any existing molecular fragments, there is still information that it does not take advantage of, and which can further improve the performance of the algorithm. The most important information missing from our 3D-ASAP formulation are the residual dipolar couplings (RDC) measurements that give angle information ($cos^2(\theta)$) with respect to a global orientation. Another possible approach is to consider an energy based formulation that captures the interaction between pairs of atoms (e.g. Lennard-Jones potential), and use this information to better localize the patches.

Another information one may use to further increase the robustness to noise is the distinction between the "good" and "noisy" edges. There are two parts of the algorithm that can benefit from such information. First, in the preprocessing step for localizing the patches one may enforce the "good" distances as hard constraints in the SDP formulation. Second, in the step that synchronizes the translations using least squares, one may choose to give more weight to equations involving the "good" edges, keeping in mind however that such equations are not noise free since the direction of an edge may be noisy as a result of steps 1 and 2, even if the distances are accurate. However, note that 3D-ASAP does use the "good" edges as hard constraints in the gradient descent refinement at the end of step 3.

One other possible future direction is combining the reflection, rotation and translation steps into a single step, thus doing synchronization over the Euclidean group. Our current approach in 3D-ASAP takes one step in this direction, and combines the reflection and rotations steps by doing synchronization over the Orthogonal group O(3). However, incorporating the translations step imposes additional challenges due to the non-compactness of the group Euc(3), rendering the eigenvector method no longer applicable directly.

In general, there exist very few theoretical guaranties for graph localization algorithms, especially in the presence of noise. A natural extension of this paper is a theoretical analysis of ASAP, including performance guarantees in terms of robustness to noise for a variety of graph models and noise models. However, a complete analysis of the noise propagation through the pipeline of Figure 3.2 is out of our reach at the moment, and first calls for theoretical guarantees for the SDP formulations used for localizing the patches in the preprocessing step. Recent work in this direction is due to [64], whose main result provides a theoretical characterization of the robustness properties of an SDP-based algorithm, for random geometric graphs and uniformly bounded adversarial noise.

While the experimental results returned by 3D-ASAP are encouraging when compared to other graph realization algorithms, we still believe that there is room for improvement, and expect that further combining the approaches of 3D-ASAP and DISCO would increase the robustness to noise even more. The disadvantage of 3D-ASAP is that it sometimes uses patches of very small size, which in the noisy scenario, can be poorly aligned with neighboring patches because of small, possibly inaccurate set of overlapping nodes. One of the advantages of DISCO is that it uses larger patches, which leads to larger overlappings and more robust alignments. At the same time, the number of patches that need to be localized by an SDP algorithm is small in the case of DISCO (2^h where h is the height of the tree in the graph decomposition), thus reducing the computational cost of the algorithm. However,

DISCO does not take advantage, at a global level, of pairwise alignment information that may involve more than two patches, while the eigenvector synchronization algorithm of 3D-ASAP incorporates such local information in a globally consistent framework. Straddling the boundary between SDP computational feasibility and robustness to noise, as well as finding the "right" method of dividing the initial problem are future research directions. For a given patch of size k, is it better to run an SDP localization algorithm on the whole graph, or to first split the graph into two or more subgraphs, localize each one and then merge the solutions to recover the initial whole patch? An analysis of this question, both from a computational point of view and with respect to robustness to noise, would reveal more insight into creating a hybrid algorithm that combines the best aspects of 3D-ASAP and DISCO. The 3D-SP-ASAP algorithm is one step in the direction, and was able to address some of these challenges.

Chapter 4

The group synchronization problem

This chapter is a self contained analysis of the group synchronization problem and its variations. Finding group elements from noisy measurements of their ratios is also known as the synchronization problem [66, 41]. The eigenvector method was introduced in [92] for solving the synchronization problem over the group SO(2) of planar rotations, which is one of the building blocks for our ASAP algorithms. Namely, we reduced the graph realization problem to three consecutive synchronization problems that overall solve the synchronization problem over Euc(d). Intuitively, we used the eigenvector method for the compact part of the group, when we synchronize over the groups $\mathbb{Z}_2$ and SO(2) as in 2D-ASAP, or over O(3) as in 3D-ASAP, to recover the reflections and rotations of all patches. We then use the least-squares method for the non-compact part, to estimate the translations. Throughout this chapter we use the notation SYNC($\mathcal{G}$) to denote the synchronization problem over the group $\mathcal{G}$, where $\mathcal{G} \in \{\mathbb{Z}_2, \mathrm{SO(d)}, \mathrm{O(d)}\}$.

The organization of this chapter is as follows:

1. Section 4.1 is an introduction to the group synchronization problem, and summarizes the different instances of the problem and its usefulness in the ASAP algorithms.

2. In Section 4.2, we motivate the robustness to noise of the eigenvector synchronization method for using tools from spectral graph theory.

3. In Section 4.3 we focus on SYNC(O(3)) and show that, in the noise free case, the top three eigenvectors of the incomplete matrix of pairwise group measurements perfectly recover the unknown group elements.

4. Section 4.4 gives an analysis of different approaches to the synchronization problem over $\mathbb{Z}_2$ with anchor information, which is useful for incorporating molecular fragment information when estimating the reflections of the remaining patches.

4.1 Group synchronization problem

In general, the synchronization problem can be applied in such settings where the underlying problem exhibits a group structure and one has available noisy measurements of ratios of group elements. We have already seen in the previous chapters several such instances of the group synchronization problem. The eigenvector and SDP-based methods for solving one such instance were originally introduced by Singer in [92], and pertains to the angular synchronization problem, SYNC(SO(2)). There, one is asked to estimate N unknown angles $\theta_1, \ldots, \theta_N \in [0, 2\pi)$ given M noisy measurements δ_{ij} of their offsets $\theta_i - \theta_j \mod 2\pi$. The difficulty of the problem is amplified on one hand by the amount of noise in the offset measurements, and on the other hand by the fact that $M << \binom{N}{2}$, i.e., only a small subset of all possible offsets are measured. In general, one may consider any group $\mathcal{G}$ other than SO(2), for which there are available noisy measurements g_{ij} of ratios between group elements

$$g_{ij} = g_i g_j^{-1}, g_i, g_j \in \mathcal{G}$$

As long as the group $\mathcal{G}$ is compact and has a real or complex representation, one may construct a real or Hermitian matrix (which may be a matrix of matrices) where the element in the position $\{ij\}$ is the matrix representation of the measurement g_{ij} (possibly a matrix of size 1×1), or the zero matrix if there is no direct measurement for the ratio of g_i and g_j. For example, the rotation group SO(3) has a real representation using 3×3 rotation matrices, and the rotation group SO(2) has a complex representation as points on unit circle $e^{\imath\theta_i} = \cos\theta_i + \imath \sin\theta_i$. One may now make use of the top eigenvectors of this matrix to estimate the unknown group elements. Alternatively, one may use this matrix to formulate an SDP program and extract the unknown group elements. The set E of pairs $\{ij\}$ for which a ratio of group elements is available can be realized as the edge set of a graph $G^P = (V^P, E^P)$, $|V^P| = N, |E^P| = M$ with vertices corresponding to the group elements $g_1, \ldots, g_N$ and edges corresponding to the available measurements $g_{ij} = g_i g_j^{-1}$. Note that we use the superscript to denote the patch graph, and keep the notation consistent with the previous chapter. Two vertices i and j of the graph G^P are connected, i.e., $\{ij\} \in E^P$, if and only if their corresponding patches P_i and P_j have enough points in common and can be pairwise aligned.

SYNC(SO(2)). The setting in which synchronization shows up in the ASAP algorithms is the following. Assume one has n overlapping subgraphs (that we refer to as patches) of the same graph, together with an Euclidean embedding of each of the n patches into $\mathbb{R}^2$. For simplicity, assume that each such patch differs from its original embedding in one global framework by only a rotation transformation. The task now is to estimate the rotation associated to each patch, that would align all such patches in a globally consistent framework (up to a global rotation), based only on information given by the pairwise alignment of a small subset of pairs of patches. In other words, to each patch one may associate an element $r_i \in$ SO(2), $i = 1, \ldots, N$ that we represent as a point on the unit circle in the complex plane $r_i = e^{\imath\theta_i} = \cos\theta_i + \imath \sin\theta_i$. Using alignment methods such as those introduced in Section 2.5, we estimate the angle θ_{ij} between two overlapping patches, i.e., the angle by which one needs to rotate patch P_i to align it with patch P_j. When the aligned patches contain corrupted distance measurements, θ_{ij} is a noisy measurement of their offset $\theta_i - \theta_j$

$\mod 2\pi$. Next, we build the $N \times N$ sparse symmetric matrix $R = (r_{ij})$ whose elements are either 0 or points on the unit circle in the complex plane:

$$r_{ij} = \begin{cases} e^{\imath\theta_{ij}} & \text{if } (i,j) \in E^P \\ 0 & \text{if } (i,j) \notin E^P \end{cases}. \tag{4.1}$$

SYNC(O(3)). The setup of the synchronization problem over the group O(3) used in 3D-ASAP is very similar to its counterpart used in 2D-ASAP, summarized above. One difference is that, in 2D-ASAP, we first estimated the reflections, and based on that, we further estimated the rotations. However, it makes sense to combine the two steps, and perhaps further increase the robustness to noise of the algorithm. By doing this, information contained in the pairwise rotation matrices helps in better estimating the reflections, and vice-versa, information on the pairwise reflection between patches helps in improving the estimated rotations.

We denote the orthogonal transformation of patch P_i by $h_i \in$ O(3), which is defined up to a global orthogonal rotation and reflection. The alignment of every pair of patches P_i and P_j whose intersection is sufficiently large, provides a perhaps noisy measurement h_{ij} (a 3×3 orthogonal matrix) for the ratio $h_i h_j^{-1}$. Next, we build the following $3N \times 3N$ sparse symmetric matrix $H = (h_{ij})$, where h_{ij} is the a 3×3 orthogonal matrix that aligns patches P_i and P_j and $O_{3\times 3}$ denotes the 3×3 zero matrix

$$H_{ij} = \begin{cases} h_{ij} & (i,j) \in E^P \\ O_{3\times 3} & (i,j) \notin E^P \end{cases} \tag{4.2}$$

In Section 4.3 we show that, in the noise free case, the top three eigenvectors of a normalized version of the matrix H perfectly recover the unknown group elements $h_1, \ldots, h_N$.

SYNC($\mathbb{Z}_2$). Synchronization over the group $\mathbb{Z}_2$ is useful for estimating the reflection of the patches in the ASAP algorithms. We use the top eigenvector of the following $N \times N$ sparse symmetric matrix $Z = (z_{ij})$

$$z_{ij} = \begin{cases} 1 & \text{aligning } P_i \text{ with } P_j \text{ did not require reflection} \\ -1 & \text{aligning } P_i \text{ with } P_j \text{ required reflection of one of them} \\ 0 & (i,j) \notin E^P \text{ } (P_i \text{ and } P_j \text{ cannot be aligned}) \end{cases} \tag{4.3}$$

In addition, in ongoing work, we apply the SYNC($\mathbb{Z}_2$) problem and its variations to the U.S. Congress data set of roll call voting patterns in the U.S. Senate across time, to identify the two existing communities, i.e. the Democratic and Republican parties [31]. There, $z_{ij} = 1$ if two senators cast similar votes, and $z_{ij} = -1$ otherwise.

4.2 Analysis of synchronization over SO(2) and $\mathbb{Z}_2$

We denote by $\mathcal{Z}$, $\mathcal{R}$, respectively $\mathcal{H}$ the normalized versions of the measurement matrices Z, R, respectively H, introduces in the previous section. As detailed in the previous two chapters, we use the top eigenvectors of these matrices to recover the global reflections

and global rotation angles of all patches. In this section, we analyze the algorithm from a spectral graph theory point of view, that allows us to explain the success of the algorithm even in the presence of corrupted measurements. Here we focus on the $N \times N$ matrices $\mathcal{Z}$ and $\mathcal{R}$, while in the next section we show a similar analysis for the $3N \times 3N$ tensor product matrix $\mathcal{H}$.

We first analyze the first two steps of 2D-ASAP, hence the problems SYNC($\mathbb{Z}_2$) and SYNC(SO(2)), when the distance measurements are exact and the matrices Z and R contain no errors on the relative reflections and rotations of all overlapping pairs of patches. Denoting by Υ the $N \times N$ diagonal matrix with ± 1 on its diagonal representing the correct reflections z_i, i.e. $\Upsilon_{ii} = z_i$, we can write the matrix $Z = (z_{ij})$ as

$$Z = \Upsilon A^P \Upsilon^{-1}, \tag{4.4}$$

where $A^P = (a_{ij}^P)$ is the adjacency matrix of the patch graph G^P, given by

$$a_{ij}^P = \begin{cases} 1 & \text{if } (i,j) \in E^P \\ 0 & \text{if } (i,j) \notin E^P \end{cases}, \tag{4.5}$$

because in the noise-free case $z_{ij} = z_i z_j^{-1}$ for $(i,j) \in E^P$. Similarly, we represent the matrix $R = (r_{ij}) = (e^{\imath\theta_{ij}})$ as

$$R = \Theta A^P \Theta^{-1}, \tag{4.6}$$

where Θ is an $N \times N$ diagonal matrix with $\Theta_{ii} = e^{\imath\theta_i}$, because in the noise-free case $e^{\imath\theta_{ij}} = e^{\imath(\theta_i - \theta_j)}$ for $(i,j) \in E^P$. The normalized matrices $\mathcal{Z}$ and $\mathcal{R}$ can now be written as

$$\mathcal{Z} = \Upsilon(D^{-1}A^P)\Upsilon^{-1}. \tag{4.7}$$

and

$$\mathcal{R} = \Theta(D^{-1}A^P)\Theta^{-1}. \tag{4.8}$$

Hence, $\mathcal{Z}$, $\mathcal{R}$ and $D^{-1}A^P$ all have the same eigenvalues. Since the normalized discrete graph Laplacian $\mathcal{L}$ is defined as

$$\mathcal{L} = I - D^{-1}A^P, \tag{4.9}$$

it follows that in the noise-free case, the eigenvalues of $I - \mathcal{Z}$ and $I - \mathcal{R}$ are the same as the eigenvalues of $\mathcal{L}$. These eigenvalues are all non-negative, since $\mathcal{L}$ is similar to the positive semidefinite matrix $I - D^{-1/2}A^P D^{-1/2}$, whose non-negativity follows from the identity

$$x^T(I - D^{-1/2}A^P D^{-1/2})x = \sum_{(i,j)\in E^P} \left(\frac{x_i}{\sqrt{deg(i)}} - \frac{x_j}{\sqrt{deg(j)}} \right)^2 \geq 0.$$

In other words,

$$1 - \lambda_i^{\mathcal{Z}} = 1 - \lambda_i^{\mathcal{R}} = \lambda_i^{\mathcal{L}} \geq 0, \quad i = 1, 2, \ldots, N, \tag{4.10}$$

where the eigenvalues of $\mathcal{L}$ are ordered in increasing order, i.e., $\lambda_1^{\mathcal{L}} \leq \lambda_2^{\mathcal{L}} \leq \cdots \leq \lambda_N^{\mathcal{L}}$, and the corresponding eigenvectors $v_i^{\mathcal{L}}$ satisfy $\mathcal{L}v_i^{\mathcal{L}} = \lambda_i^{\mathcal{L}} v_i^{\mathcal{L}}$. Furthermore, the sets of eigenvectors are related by

$$v_i^{\mathcal{Z}} = \Upsilon v_i^{\mathcal{L}}, \quad v_i^{\mathcal{R}} = \Theta v_i^{\mathcal{L}}.$$

If the patch graph G^P is connected, then the eigenvalue $\lambda_1^{\mathcal{L}} = 0$ is simple and its corresponding eigenvector $v_1^{\mathcal{L}}$ is the all-ones vector $\mathbf{1} = (1, 1, \ldots, 1)^T$. Therefore,

$$v_1^{\mathcal{Z}} = \Upsilon\mathbf{1}, \quad v_1^{\mathcal{R}} = \Theta\mathbf{1}, \tag{4.11}$$

and, in particular,

$$v_1^{\mathcal{Z}}(i) = z_i, \quad v_1^{\mathcal{R}}(i) = e^{\imath\theta_i}, \quad i = 1, 2, \ldots, N. \tag{4.12}$$

This implies that in the noise-free case, the ASAP algorithm perfectly recovers the reflections and rotations as shown in Figures 2.4(a) and 2.6(a).

Notice that the top eigenvectors v_1^Z and v_1^R of the non-normalized matrices Z and R are related to the top eigenvector v_1^A of the adjacency matrix A^P via

$$v_1^Z = \Upsilon v_1^A, \quad v_1^R = \Theta v_1^A. \tag{4.13}$$

Connectivity of the patch graph together with the Perron-Frobenius theorem imply that all entries of v_1^A are positive, that is, $v_1^A(i) > 0$, which in principle suffices to ensure that our rounding procedures (2.8) and (2.11) give the correct reflections and rotations. However, unlike the constant entries of the all-ones vector, the entries of v_1^A can vary in their magnitude. In fact, when the patch graph is not regular (i.e., when the vertex degrees are not constant), it often happens that v_1^A has only a few large entries and all other entries are significantly smaller, rendering numerical difficulties in the computation of v_1^A, as indicated in Figures 2.4 and 2.6. Moreover, in the noisy case, such small entries are likely to change their sign (or phase), making the top eigenvector of Z (or R) sensitive to noise.

In order to understand why the entries of v_1^A can vary so much, we first examine the matrix $D^{-1}A^P$ and view it as a Markov transition probability matrix of a discrete random walk on the patch graph, whose top all-ones eigenvector expresses the fact that the steady state density is uniform. Denoting the maximum vertex degree of the patch graph by $Deg = \max_i deg(i)$, the matrix $\frac{1}{Deg}A^P$ corresponds to a random walk with absorption, i.e., it is possible to artificially add an extra terminal state to which the random walker jumps from node i with probability $1 - \frac{deg(i)}{Deg}$. Due to this absorption, the steady state distribution is trivially concentrated at the terminal state, but the approach to this steady state is governed by v_1^A. We therefore expect vertices located away from absorption sites to have larger values in v_1^A. For example, consider the path graph on N vertices with edges given by $(i, i+1)$ for $i = 1, \ldots, N-1$. For the path graph, $deg(i) = 2$ for $i = 2, \ldots, N-1$, while $deg(1) = deg(N) = 1$. The discrete random walk matrix $\frac{1}{2}A^P$ is a discretization of the continuous diffusion process on the interval $[0, 1]$ with absorption at the endpoints (homogenous Dirichlet boundary conditions), from which it can be deduced that in the limit $N \to \infty$ the top eigenvector is approximately given by $v_1^A(i) \sim \sin\left(\frac{\pi i}{N+1}\right)$ for $i = 1, \ldots, N$. This agrees with our intuition that values near the center are larger than near the boundaries, but also demonstrates the possible numerical instabilities, as the ratio between these values can be as large as $O(N)$. Figure 4.1 demonstrates the variability of the node degrees of patches in the patch graph for the US cities graph, rendering the importance of the normalization.

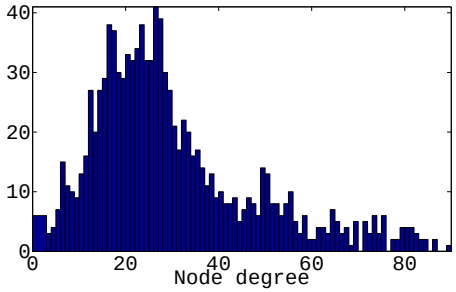


Figure 4.1: Histogram of the node degrees of patches in the patch graph G^P for the US cities graph with $\rho = 0.032$ and $\eta = 20\%$.

At this point, we understand why the top eigenvectors of the normalized matrices $\mathcal{Z}$ and $\mathcal{R}$ give superior results compared to the top eigenvectors of the non-normalized matrices Z and R. Since the (non-normalized) Laplacian L of the patch graph

$$L = D - A^P$$

also has the all-ones vector as an eigenvector (with smallest eigenvalue), another possible good way of estimating the reflections and rotations is by using the smallest eigenvectors of

$$L^Z = D - Z,$$

and

$$L^R = D - R.$$

We have seen that, in practice, the non-normalized Laplacian method (L) also performs well, giving results that are comparable to those of the normalized Laplacian method ($\mathcal{L}$).

We now turn to briefly discuss the analysis of the noisy distances scenario, dealing first with Step 1 for the reflections. For noisy data, the measurement z_{ij} of the reflection between patches P_i and P_j may be incorrect. That is, while the value of z_{ij} should really be $z_i z_j^{-1}$, the alignment of the patches may give the false value $-z_i z_j^{-1}$. The effect of false measurements changes the matrices Z and $\mathcal{Z}$ from (4.4) and (4.7) to

$$Z = \Upsilon(A^P + \Delta)\Upsilon^{-1}, \tag{4.14}$$

and

$$\mathcal{Z} = \Upsilon(D^{-1}A^P + D^{-1}\Delta)\Upsilon^{-1}, \tag{4.15}$$

where $\Delta = (\delta_{ij})$ is the $N \times N$ symmetric error matrix

$$\delta_{ij} = \begin{cases} -2 & \text{if } (i,j) \in E^P \text{ and } z_{ij} \neq z_i z_j^{-1} \text{ (bad alignment)} \\ 0 & \text{if } (i,j) \in E^P \text{ and } z_{ij} = z_i z_j^{-1} \text{ (good alignment)} \\ 0 & \text{if } (i,j) \notin E^P \end{cases}. \tag{4.16}$$

While the all-ones vector $\mathbf{1}$ is the top eigenvector of $D^{-1}A^P$, it is no longer the top eigenvector of the perturbed matrix $D^{-1}A^P + D^{-1}\Delta$. If the perturbation $D^{-1}\Delta$ is small (e.g., in

terms of its spectral norm), then we can expect the top eigenvector to be sufficiently close to $\mathbf{1}$. In particular, many of the eigenvector entries are expected to remain positive, meaning that the reflections corresponding to these entries will be estimated correctly. A similar perturbation approach can also be applied to Step 2 for the rotations. The precise matrix perturbation analysis that quantifies the "sign stability" of the top eigenvector is beyond the scope of this thesis, however, and is postponed for future work.

From the implementation perspective, it is important to note that the eigenvector method can be implemented in a distributed manner. The top eigenvector of the matrix $\mathcal{Z}$ can be efficiently computed by the power iteration method that starts from a randomly chosen vector b_0 and iterates $b_{n+1} = \frac{\mathcal{Z}b_n}{\|\mathcal{Z}b_n\|}$. Each iteration requires just a matrix-vector multiplication that takes only $O(M)$ operations, where $M = |E^P|$ is the number of edges in the patch graph G^P. The power iteration method has the advantage that it can be implemented in a distributed way with every sensor making local computations and communications with nearby sensors. The number of iterations required decreases as the spectral gap increases.

The iterations of the power method for computing the top eigenvector can also be viewed as integration of consistency relations along cycles in the patch graph G^P. To see this, consider for example a length k cycle $P_1, P_2, \ldots, P_k$ where, $(P_i, P_{i+1}) \in E^P$ for $i = 1, 2, \ldots, k-1$ and $(P_k, P_1) \in E^P$. In the noise-free case, the reflection measurements are given by $z_{ij} = z_i z_j^{-1}$, hence they must satisfy the consistency relation

$$z_{12} z_{23} \cdots z_{k-1,1} z_{k,1} = 1. \tag{4.17}$$

Similarly, the noise-free rotation measurements $r_{ij} = e^{\imath(\theta_i - \theta_j)}$ also satisfy a similar consistency relation

$$r_{12} r_{23} \cdots r_{k-1,1} r_{k,1} = 1. \tag{4.18}$$

In the iterations of the power method, the matrix $\mathcal{Z}$ (and similarly $\mathcal{R}$) gets multiplied by itself and the effect of this is twofold. First, the eigenvector method integrates the information in the consistency relations along cycles in the patch graph G^P, and second, it propagates information to far-away patches that cannot be aligned directly. This gives yet another insight to understanding why the eigenvector method is robust to noise.

4.3 Analysis of synchronization over O(3)

Let us now show that, in the noise free case, the top three eigenvectors of $\mathcal{H}$ perfectly recover the unknown group elements. We denote by h_i the 3×3 matrix corresponding to the i^{th} submatrix in the $3 \times N$ matrix $[v_1^{\mathcal{H}}, v_2^{\mathcal{H}}, v_3^{\mathcal{H}}]$. In the noise free case, h_i is an orthogonal matrix and represents the solution which aligns patch P_i in the global coordinate system, up to a global orthogonal transformation. To see this, we first let h denote the $3N \times 3$ matrix formed by concatenating the true orthogonal transformation matrices $h_1, \ldots, h_N$. Note that when the patch graph G^P is complete, H is a rank 3 matrix since $H = hh^T$, and its top three eigenvectors are given by the columns of h

$$Hh = hh^T h = hNI_3 = Nh. \tag{4.19}$$

In the general case when G^P is a sparse connected graph, note that

$$Hh = Dh, \text{ hence } D^{-1}Hh = \mathcal{H}h = h, \tag{4.20}$$

and thus the three columns of h are each eigenvectors of matrix $\mathcal{H}$, associated to the same eigenvalue $\lambda = 1$ of multiplicity 3. It remains to show this is the largest eigenvalue of $\mathcal{H}$. We recall that the adjacency matrix of G^P is A^P, and denote by $\mathcal{A}^P$ the $3N \times 3N$ matrix built by replacing each entry of value 1 in A^P by the identity matrix I_3, i.e., $\mathcal{A}^P = A^P \otimes I_3$ where $\otimes$ denotes the tensor product of two matrices. As a consequence, the eigenvalues of $\mathcal{A}^P$ are just the direct products of the eigenvalues of I_3 and A^P, and the corresponding eigenvectors of $\mathcal{A}^P$ are the tensor products of the eigenvectors of I and A^P. Furthermore, if we let Δ denote the $N \times N$ diagonal matrix with $\Delta_{ii} = deg(i)$, for $i = 1, \ldots, N$, it holds true that

$$D^{-1}\mathcal{A}^P = (\Delta^{-1}A^P) \otimes I_3, \tag{4.21}$$

and thus the eigenvalues of $D^{-1}\mathcal{A}^P$ are the same as the eigenvalues of $\Delta^{-1}A^P$, each with multiplicity 3. In addition, if Υ denotes the $3N \times 3N$ matrix with diagonal blocks h_i, $i = 1, \ldots, N$, then the normalized alignment matrix $\mathcal{H}$ can be written as

$$\mathcal{H} = \Upsilon D^{-1}\mathcal{A}^P\Upsilon^{-1}, \tag{4.22}$$

and thus $\mathcal{H}$ and $D^{-1}\mathcal{A}^P$ have the same eigenvalues, which are also the eigenvalues of $\Delta^{-1}A^P$, each with multiplicity 3. Whenever it is understood from the context, we will omit from now on the remark about the multiplicity 3. Since the normalized discrete graph Laplacian $\mathcal{L}$ is defined as

$$\mathcal{L} = I - \Delta^{-1}A^P, \tag{4.23}$$

it follows that in the noise-free case, the eigenvalues of $I - \mathcal{H}$ are the same as the eigenvalues of $\mathcal{L}$. These eigenvalues are all non-negative, since $\mathcal{L}$ is similar to the positive semidefinite matrix $I - \Delta^{-1/2}A^P\Delta^{-1/2}$, whose non-negativity follows from the identity

$$x^T(I - \Delta^{-1/2}A^P\Delta^{-1/2})x = \sum_{(i,j)\in E^P} \left(\frac{x_i}{\sqrt{deg(i)}} - \frac{x_j}{\sqrt{deg(j)}} \right)^2 \geq 0.$$

In other words,

$$1 - \lambda^{\mathcal{H}}_{3i-2} = 1 - \lambda^{\mathcal{H}}_{3i-1} = 1 - \lambda^{\mathcal{H}}_{3i} = \lambda^{\mathcal{L}}_i \geq 0, \quad i = 1, 2, \ldots, N, \tag{4.24}$$

where the eigenvalues of $\mathcal{L}$ are ordered in increasing order, i.e., $\lambda^{\mathcal{L}}_1 \leq \lambda^{\mathcal{L}}_2 \leq \cdots \leq \lambda^{\mathcal{L}}_N$. If the patch graph G^P is connected, then the eigenvalue $\lambda^{\mathcal{L}}_1 = 0$ is simple (thus $\lambda^{\mathcal{L}}_2 > \lambda^{\mathcal{L}}_1$) and its corresponding eigenvector $v^{\mathcal{L}}_1$ is the all-ones vector $\mathbf{1} = (1, 1, \ldots, 1)^T$. Therefore, the largest eigenvalue of $\mathcal{H}$ equals 1 and has multiplicity 3, i.e., $\lambda^{\mathcal{H}}_1 = \lambda^{\mathcal{H}}_2 = \lambda^{\mathcal{H}}_3 = 1$, and $\lambda^{\mathcal{H}}_4 > \lambda^{\mathcal{H}}_3$. This concludes our proof that, in the noise free case, the top three eigenvectors of $\mathcal{H}$ perfectly recover the true solution $h_1, \ldots, h_N \in$ O(3), up to a global orthogonal transformation.

4.4 Synchronization over $\mathbb{Z}_2$ with molecular fragments

In the molecule problem, there are molecular fragments whose local configuration is known in advance up to an element of the special Euclidean group SE(3) rather than the Euclidean group E(3). In other words, these are small structures that need to be translated and rotated with respect to the global coordinate system, but no reflection is required. As a result, for their corresponding patches we know the corresponding group element in $\mathbb{Z}_2$. This motivates us to consider the problem of synchronization over $\mathbb{Z}_2$ when "molecular fragment" information is available, and refer to it from now on as SYNC($\mathbb{Z}_2$). We propose and compare four methods for solving SYNC($\mathbb{Z}_2$): two relaxations to a quadratically constrained quadratic program (QCQP), and two semidefinite programming (SDP) formulations.

Mathematically, the synchronization problem over the group $\mathbb{Z}_2 = \{\pm 1\}$ can be stated as follows: given k group elements (anchors) $\mathcal{A} = \{a_1, \ldots, a_k\}$ and a set of (possibly) incomplete (noisy) pairwise group measurements $z_{ij} = a_i x_j^{-1}$ and $z_{ij} = x_i x_j^{-1}$, $(i,j) \in E(G)$, find the unknown group elements (sensors) $\mathcal{S} = \{x_1, \ldots, x_l\}$. We may sometimes abuse notation and denote all nodes by $x = \{x_1, \ldots, x_l, x_{l+1}, \ldots, x_N\}$, where $N = l + k$, with the understanding that the last k elements denote the anchors. Whenever we say that indices $i \in \mathcal{S}$ and $j \in \mathcal{A}$, it should be understood that we refer to sensor $x_i \in \mathcal{S}$, respectively anchor $a_j \in \mathcal{A}$. In the molecule problem, k denotes the number of patches whose reflection is known a priori , and the goal is to estimate the reflection of the remaining $l = N - k$ patches.

In the absence of anchors (when $k = 0$ and $N = l$), the synchronization problem over $\mathbb{Z}_2$ was considered in [32], following the approach for angular synchronization introduced in [92]. The goal is to estimate N unknown group elements $z_1, \ldots, z_N \in \mathbb{Z}_2$, whose pairwise group relations are captured in a sparse $N \times N$ matrix $Z = (z_{ij})$ where

$$z_{ij} = \begin{cases} z_i z_j^{-1} & \text{if the measurement is correct} \\ -z_i z_j^{-1} & \text{if the measurement is incorrect} \end{cases} \tag{4.25}$$

The set E^P of pairs (i,j) for which a pairwise measurement exists (correct or incorrect) can be realized as the edge set of the patch graph $G^P = (V^P, E^P)$, where vertices correspond to patches, and edges to the measurements. We denote the original solution by $z_1, \ldots, z_N$ and our approximated solution by $x_1, \ldots, x_N$. Our task is to estimate $x_1, \ldots, x_N$ such that we satisfy as many pairwise group relations as possible. To that end, we start by considering the problem of maximizing the following quadratic form

$$\max_{x_1,\ldots,x_N \in \mathbb{Z}_2^N} \sum_{i,j=1}^{N} x_i Z_{ij} x_j = \max_{x_1,\ldots,x_N \in \mathbb{Z}_2^N} x^T Z x, \tag{4.26}$$

whose maximum, in the noise-free case, is attained when $x = z$. Note that for the correct set of reflections $z_1, \ldots, z_N$, each correct group measurement $z_{ij} = z_i z_j^{-1}$ contributes $z_i Z_{ij} z_j = +1$ to the sum in (4.26), while each incorrect measurement will add a -1 to the summation. Our task now is to solve the quadratic integer optimization problem in (4.26), which is unfortunately a non-convex optimization problem. Since NP-hard problems, such as the maximal clique problem, can be formulated as an integer quadratic problem, the

maximization in (4.26) is itself NP-hard. We therefore make use of the following relaxation

$$\max_{\sum_{i=1}^{N}|x_i|^2=N} \sum_{i,j=1}^{N} x_i Z_{ij} x_j = \max_{\|x\|^2=N} x^T Z x \tag{4.27}$$

whose maximum is achieved when $x = v_1$, where v_1 is the normalized top eigenvector of Z, satisfying $Zv_1 = \lambda_1 v_1$ and $\|v_1\|^2 = N$, with λ_1 being the largest eigenvalue. Thus an approximate solution to the maximization problem in (4.26) is given by the top eigenvector of the symmetric matrix Z. In practice, for increased robustness to noise, we use the following alternative relaxation

$$\max_{x^T D x = 2m} x^T Z x, \tag{4.28}$$

where $D_{ii} = \sum_{j=1}^{N} |Z_{ij}|$, m denotes the number of edges, and hence $2m$ is the sum of the node degrees. The solution to the maximization problem in (4.28) is given by the top eigenvector of the normalized matrix $D^{-1}Z$. We refer the reader to [92, 32] for an analysis of the eigenvector method and its connection with the normalized discrete graph Laplacian.

4.4.1 Synchronization by relaxing a QCQP

When anchor information is available, meaning that we know a priori some of the group elements which we refer to as anchors, we follow a similar approach to equations (4.26, 4.27) that motivated the eigenvector synchronization method. Similarly, we are interested in finding the set of unknown elements that maximize the number of satisfied pairwise measurements, but this time, under the additional constraints imposed by the anchors, i.e. $x_i = a_i, i \in \mathcal{A}$. Unfortunately, maximizing the quadratic form $x^T Z x$ under the anchor constraints, is no longer an eigenvector problem. Our approach is to combine under the same objective function both the contribution of the sensor-sensor pairwise measurements (as a quadratic term) and the contribution of the anchors-sensor pairwise measurements (as a linear term). To that end, we start by formulating the synchronization problem as a least squares problem, by minimizing the following quadratic form

$$\begin{aligned}
\min_x \sum_{(i,j)\in E} (x_i - Z_{ij} x_j)^2 &= \min_x \sum_{(i,j)\in E} z_i^2 + Z_{ij}^2 x_j^2 - 2 Z_{ij} x_i x_j \\
&= \min_x \sum_{(i,j)\in E} x_i^2 + x_j^2 - 2 Z_{ij} x_i x_j \\
&= \min_x \sum_{i=1}^{n} d_i x_i^2 - \sum_{(i,j)\in E} 2 Z_{ij} x_i x_j \\
&= \min_x x^T D x - x^T Z x \\
&= \min_x x^T (D - Z) x
\end{aligned} \tag{4.29}$$

To account for the existence of anchors, we first write the matrices Z and D in the following block format

$$Z = \begin{bmatrix} S & U \\ U^T & V \end{bmatrix}, \qquad D = \begin{bmatrix} D_S & 0 \\ 0 & D_V \end{bmatrix}$$

where $S_{l\times l}$, $U_{l\times k}$ and $V_{k\times k}$ denote the sensor-sensor, sensor-anchor respectively anchor-anchor measurements, and D is a diagonal matrix with $D_{ii} = \sum_{j=1}^{N} |Z_{ij}|$. Note that V is a matrix with all nonzero entries, since the measurement between any two anchors is readily available. Similarly, we write (with a slight abuse of notation) the solution vector in the form $x = [s\ a]^T$, where s denotes the signs of the sensor nodes, and a the signs of the anchor nodes. The quadratic function minimized in (4.29) can now be written in the following form

$$\begin{bmatrix} s^T & a^T \end{bmatrix} \begin{bmatrix} D_S - S & -U \\ -U^T & D_V - V \end{bmatrix} \begin{bmatrix} s \\ a \end{bmatrix} = s^T(D_S - S)s - 2s^TUa + a^T(D_V - V)a \quad (4.30)$$

The vector $(Ua)_{l\times 1}$ can be interpreted as the anchor contribution in the estimation of the sensors. Note that in the case when the sensor-anchors measurements should be trusted more than the sensor-sensor measurements, the anchor contribution can be weighted accordingly, and equation (4.30) becomes $z^T(D_S - S)z - 2\gamma z^TUa + a^T(D_V - V)a$, for a given weight γ. Since $a^T(D_V - V)a$ is a (nonnegative) constant, we are interested in minimizing the integer quadratic form

$$\underset{z\in\mathbb{Z}_2^l}{\text{minimize}}\ z^T(D_S - S)z - 2z^TUa.$$

Unfortunately, solving such a problem is NP-hard, and thus we introduce the following relaxation to a quadratically constrained quadratic program (QCQP)

$$\begin{aligned} \underset{z=(z_1,\ldots,z_l)}{\text{minimize}} \quad & z^T(D_S - S)z - 2z^TUa \\ \text{subject to} \quad & z^Tz = l \end{aligned} \quad (4.31)$$

We proceed by considering the Lagrangian function

$$\phi(z,\lambda) = z^T(D_S - S)z - 2z^TUa + \lambda(z^Tz - l) \quad (4.32)$$

where λ is the Lagrange multiplier. Differentiating (4.32) with respect to z, we are led to the following equation

$$2(D_S - S)z - 2Ua + 2\lambda z = 0$$

which can be written as

$$(D_S - S + \lambda I)z = Ua \quad (4.33)$$

Using the fact that $z^Tz = l$, (4.33) becomes

$$(Ua)^T(D_S - S + \lambda I)^{-2}(Ua) = l \quad (4.34)$$

which we solve for λ. We refer the reader to [8] for a more detailed analysis of eigenvalue solutions to such quadratically constrained optimization problems. Finally, the solution to the minimization problem formulated in (4.31) is given by

$$z^* = (D_S - S + \lambda I)^{-1}(Ua) \quad (4.35)$$

For simplicity, we choose to solve the nonlinear matrix equality (4.34) in MATLAB using the $lsqnonlin$ command, whose success in finding λ is contingent upon a good initialization. To that end, we consider the eigendecomposition of the symmetric positive definite matrix

$$D_S - S = \sum_{i=1}^{l} \lambda_i \phi_i \phi_i^T,$$

where $(D - S)\phi_i = \lambda_i \phi_i$, for $i = 1, \ldots, l$, and expand the vectors Ua and z in the form $Ua = \sum_{i=1}^{l} \beta_i \phi_i$ and $z = \sum_{i=1}^{l} \alpha_i \phi_i$. Rewriting equation (4.33), $(D_S - S)z + \lambda z = Ua$, in terms of the above expansion yields

$$\sum_{i=1}^{l} \lambda_i \alpha_i \phi_i + \lambda \sum_{i=1}^{l} \alpha_i \phi_i = \sum_{i=1}^{l} \beta_i \phi_i. \tag{4.36}$$

Thus, for every $i = 1, \ldots, l$ it must hold true that

$$\lambda_i \alpha_i + \lambda \alpha_i = \beta_i, \tag{4.37}$$

i.e. $\alpha_i = \frac{\beta_i}{\lambda + \lambda_i}$. Since $z^T z = l$ and $\sum_{i=1}^{l} \alpha_i^2 = l$, the Lagrange multiplier λ must satisfy

$$\sum_{i=1}^{l} \frac{\beta_i^2}{(\lambda + \lambda_i)^2} = l \tag{4.38}$$

The condition that $\lambda + \lambda_i > 0$ ensures that the solution search for λ lies outside the set of singularities of (4.38). Thus $\lambda > -\lambda_0$, where λ_0 is the smallest eigenvalue of the matrix $D_S - S$. Note that the row sums of $D_S - S$ are non-negative since the diagonal entries in D_S are the degree of the sensor nodes in the entire graph, taking into account the sensor-anchor edges as well, not just the sensor-sensor edges. Furthermore, $D_S - S$ is a positive semidefinite matrix (thus $\lambda_0 \geq 0$), as it can be seen from the following identity

$$x^T(D_S - S)x \;=\; \sum_{i=1}^{l} x_i^2 d_i - \sum_{(i,j)\in E_{SS}} 2x_i x_j S_{ij} \geq \sum_{(i,j)\in E_{SS}, S_{ij}=\pm 1} (x_i - S_{ij}x_j)^2 \geq 0$$

where E_{SS} denotes the set of sensor-sensor edges.

We also consider the following formulation similar to (4.31), but we replace the constraint $z^T z = l$ with $z^T D_S z = \Delta$, where $\Delta = \sum_{i=1}^{l} d_i$ is the sum of the degrees of all sensor nodes. Note that the following change of variable $\bar{z} = D_S^{1/2} z$ brings the new optimization problem to a form similar to (4.31)

$$\begin{aligned} \underset{\bar{z}}{\text{minimize}} \quad & \bar{z}^T D_S^{-1/2}(D_S - S)D_S^{-1/2}\bar{z} - 2\bar{z}^T D_S^{-1/2} Ua \\ \text{subject to} \quad & \bar{z}^T \bar{z} = \Delta \end{aligned} \tag{4.39}$$

with the corresponding Lagrangian $\bar{\lambda}$ satisfying $\bar{\lambda} > -\bar{\lambda}_0$, where $\bar{\lambda}_0$ is the smallest eigenvalue of the matrix $D_S^{-1/2}(D_S - S)D_S^{-1/2}$.

4.4.2 Synchronization by SDP

An alternative approach to solving SYNC($\mathbb{Z}_2$) is by using semidefinite programming. The objective function in (4.26) can be written as

$$\sum_{i,j=1}^{N} x_i Z_{ij} x_j = Trace(Z\Upsilon) \tag{4.40}$$

where Υ is the $N \times N$ symmetric rank-one matrix with ± 1 entries

$$\Upsilon_{ij} = \begin{cases} x_i x_j^{-1} & \text{if } i,j \in \mathcal{S} \\ x_i a_j^{-1} & \text{if } i \in \mathcal{S}, j \in \mathcal{A} \\ a_i a_j^{-1} & \text{if } i,j \in \mathcal{A} \end{cases} \tag{4.41}$$

Note that Υ has ones on its diagonal $\Upsilon_{ii} = 1, \forall i = 1, \ldots, N$, and the anchor information gives another layer of hard constraints. The SDP relaxation of (4.26) in the presence of anchors becomes

$$\begin{aligned} \underset{\Upsilon \in \mathbb{R}^{N \times N}}{\text{maximize}} \quad & Trace(Z\Upsilon) \\ \text{subject to} \quad & \Upsilon_{ii} = 1, i = 1, \ldots, N \\ & \Upsilon_{ij} = a_i a_j^{-1}, \quad \text{if } i,j \in \mathcal{A} \\ & \Upsilon \succeq 0 \end{aligned} \tag{4.42}$$

where the maximization is taken over all semidefinite positive real-valued matrices $\Upsilon \succeq 0$. While Υ as defined in (4.42) has rank one, the solution of the SDP is not necessarily of rank one. We therefore compute the top eigenvector of that matrix, and estimate $x_1, \ldots, x_s$ based on the sign of its first s entries.

Alternatively, to reduce the number of unknowns in (4.42) from $N = l + k$ to l, one may consider the following relaxation

$$\begin{aligned} \underset{\Upsilon \in \mathbb{R}^{l \times l}; x \in \mathbb{R}^l}{\text{maximize}} \quad & Trace(S\Upsilon) + 2x^T U a \\ \text{subject to} \quad & \Upsilon_{ii} = 1, \forall i = 1, \ldots, l \\ & \begin{bmatrix} \Upsilon & x \\ x^T & 1 \end{bmatrix} \succeq 0 \end{aligned} \tag{4.43}$$

Ideally, we would like to enforce the constraint $\Upsilon = xx^T$, which guarantees that Υ is indeed a rank-one solution. However, since rank constrains do not lead to convex optimization problems, we relax this constraint via Schur's lemma to $\Upsilon \succeq xx^T$. This last matrix inequality is equivalent [20] to the last constraint in the SDP formulation in (4.43). Finally, we obtain estimators $\hat{z}_1, \ldots, \hat{z}_l$ for the sensors by setting $\hat{z}_i = \text{sign}(x_i), \forall i = 1, \ldots, l$.

4.4.3 Comparison of algorithms for SYNC($\mathbb{Z}_2$)

Figure 4.2 compares the performance of the algorithms we proposed for synchronization in the presence of molecular fragment information. The adjacency graph of available pairwise

measurements is an Erdős-Rényi graph $G(N, p)$ with $N = 75$ and $p = 0.2$ (i.e., a graph with N nodes, where each edge is present with probability p, independent of the other edges). We show numerical experiments for four scenarios, where we vary the number of anchors $k = \{5, 15, 30, 50\}$ chosen uniformly at random from the N nodes. As the number of anchors increases, compared to the number of sensors $s = N - k$, the performance of the four algorithms is essentially similar. Only when the number of anchor nodes is small ($k = 5$), the SDP-Y formulation shows superior results, together with SDP-XY and QCQP with constraint $z^T Dz = \Delta$, while the QCQP with constraint $z^T z = s$ performs less well. In practice, one would choose the QCQP formulation with constraint $z^T Dz = \Delta$, since the SDP based methods are computationally expensive as the size of the problem increases. This was also our method of choice for computing the reflection of the patches in the localization of the ubiqutin (PDG 1d3z), when molecular fragment information was available and the reflection of many of the patches was known a priori .

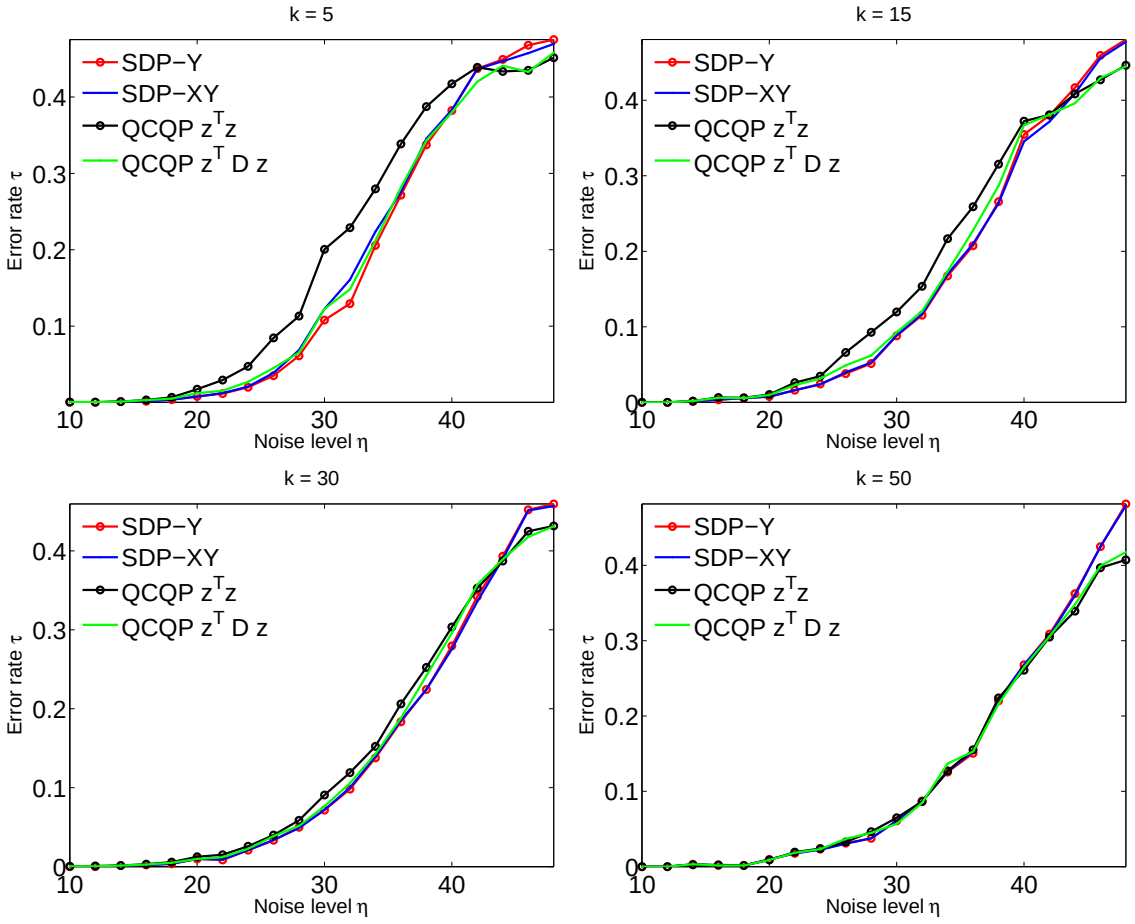


Figure 4.2: Comparison, in terms of robustness to noise, of the four algorithms we proposed for SYNC($\mathbb{Z}_2$): the two QCQP formulations using the two different constraints: $z^T z = s$ and $z^T D z = \Delta$ as they appear in equations (4.31) and (4.39), the two SDP-based formulations SDP-Y and SDP-XY as formulated in (4.42) and (4.43). The adjacency graph of available pairwise measurements is an Erdős-Rényi graph $G(N, p)$ with $N = 75$ and $p = 0.2$. Also, k denotes the number of anchors, chosen uniformly at random from the N nodes. Results are averaged over 50 runs.

Chapter 5

Low-rank matrix completion

Can the missing entries of an incomplete real valued matrix be recovered? Clearly, a matrix can be completed in an infinite number of ways by replacing the missing entries with arbitrary values. In order for the completion question to be of any value we must restrict the matrix to belong to a certain class of matrices. A popular class of matrices are the matrices of limited rank and the problem of completing a low-rank matrix from a subset of its entries has received a great deal of attention lately. The completion problem comes up naturally in a variety of settings. One of these is the *Netflix* problem [78], where users submit rankings for only a small subset of movies, and one would like to infer their preference of unrated movies. The data matrix of all user-ratings may be approximately low-rank because it is believed that only a few factors contribute to an individual's preferences. The completion problem also arises in computer vision, in the problem of inferring three-dimensional structure from motion [102], as well as in many other data analysis, machine learning [97], control [76] and other problems that are modeled by a factor model. Numerous completion algorithms have been proposed over the years, see e.g., [22, 37, 38, 62, 98, and references therein]. Many of the algorithms relax the non-convex rank constraint by the convex set of positive semidefinite matrices and solve a convex optimization problem using semidefinite programming (SDP) [105]. Recently, using techniques from compressed sensing [24, 35], Candès and Recht [23] proved that if the pattern of missing entries is random then the minimization of the convex nuclear norm (the ℓ_1 norm of the singular values vector) finds (with high probability) the exact completion of most $n\alpha \times n$ matrices of rank d as long as the number of observed entries m satisfies $m \geq C(\alpha)dn^{1.2}\log n$, where $C(\alpha)$ is some function. Even more recently, Keshavan, Oh, and Montanari [68, 69] improved the bound to $C(\alpha)dn\log n$ and also provided an efficient completion algorithm.

These fascinating recent results do not provide, however, a solution to the more practical case in which the pattern of missing entries is non-random. Given a specific pattern of missing entries, it would be extremely desirable to have an algorithm that can determine the uniqueness of a rank-d matrix completion. Prior to running any of the numerous existing completion algorithms such as SDP it is important for the analyst to know if such a completion is indeed unique.

Building on ideas from rigidity theory (see, e.g., [83]) we propose an efficient randomized algorithm that determines whether or not it is possible to uniquely complete an incomplete matrix to a matrix of specified rank d. Our proposed algorithm does not attempt to complete the matrix but only determines if a unique completion is possible. We introduce a new matrix, which we call *the completion matrix* that serves as the analogue of the rigidity matrix in rigidity theory. The rank of the completion matrix determines a property which we call infinitesimal completion. Whenever the completion matrix is large and sparse its rank can be efficiently determined using iterative methods such as LSQR [79]. As in rigidity theory, we will also make the distinction between *local* completion and *global* completion. The analogy between rigidity and completion is quite striking, and we believe that many of the results in rigidity theory can be usefully translated to the completion setup. Our randomized algorithm for testing local completion is based on a similar randomized algorithm for testing local rigidity that was suggested by Hendrickson [55].

While the local rigidity theory translates smoothly into the local completion setup, we find the global theory to be more subtle. A full characterization of globally rigid frameworks exists due to Connelly [27] who proved sufficient conditions for global rigidity and conjectured necessary conditions that were recently proved by Gortler, Healy, and Thurston [44]. Here we conjecture sufficient conditions for global completion, but we were not able to come up with suitable necessary conditions. Based on our conjectured sufficient conditions, we propose a randomized algorithm that tests for sufficient conditions but not for necessary conditions for global completion. As a result, applying our algorithms for testing local and global completion in conjunction classifies missing entry patterns into three classes. The first class contains matrices that did not pass the local completion test and are clearly not globally completable. The second class contains matrices that passed both the local completion and global completion tests, and are therefore guaranteed to be globally completable. The third class contains matrices that passed the local completion test but did not pass the global completion test, and since the latter only checks for sufficient conditions, we cannot determine if such matrices are globally completable or not.

The organization of this chapter is as follows.

1. In Section 5.1 we analyze the low-rank matrix completion problem for the particular case of positive semidefinite Gram matrices and present algorithms for testing local and global completion of such matrices.

2. In Section 5.2 the analysis is generalized to the more common completion problem of general low-rank rectangular matrices and corresponding algorithms are provided.

3. Section 5.3 is concerned with the combinatorial characterization of entry patterns that can be either locally completable or globally completable. In particular, we present a simple combinatorial characterization for rank-1 matrices and comment on the rank-2 and rank-d ($d \geq 3$) cases.

4. In Section 5.4 we detail the results of extensive numerical simulations in which we tested the performance of our algorithms while verifying the theoretical bounds of [23, 68, 69] for matrices with random missing patterns.

5. Finally, Section 5.5 is a summary and discussion.

5.1 Gram matrices

We start by analyzing the completion problem of low-rank positive semidefinite Gram matrices with missing entries. We make extensive use of the terminology and results summarized in Chapter 1 which the reader is advised to consult whenever felt needed.

For a collection of n vectors $p_1, p_2, \ldots, p_n \in \mathbb{R}^d$ there corresponds an $n \times n$ Gram matrix J of rank (at most) d whose entries are given by the inner products

$$J_{ij} = p_i^T p_j, \quad i, j = 1, \ldots, n. \tag{5.1}$$

An alternative way of writing J is through its Cholesky decomposition $J = P^T P$, where P is a $d \times n$ matrix given by $P = \begin{bmatrix} p_1 & p_2 & \cdots & p_n \end{bmatrix}$, from which it is clear that $\mathrm{rank}(J) \leq d$. If J is fully observed (no missing entries) then the Cholesky decomposition of J reveals P up to a $d \times d$ orthogonal matrix O ($OO^T = I$), as $J = P^T P = (OP)^T (OP)$. Now, suppose that only a few of the entries of J are observed by the analyst. The symmetry of the matrix J implies that the set of observed entries defines an undirected graph $G = (V, E)$ with n vertices where $(i, j) \in E$ is an edge iff the entry J_{ij} is observed. The graph may include self loop edges of the form (i, i) corresponding to observed diagonal elements J_{ii}. For an incomplete Gram matrix J with an observed pattern that is given by the graph G, we would like to know if it is possible to *uniquely* complete the missing entries of J so that the resulting completed matrix is of rank d.

For example, consider the three planar points

$$p_1 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \; p_2 = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \; p_3 = \begin{pmatrix} 2 \\ 3 \end{pmatrix} : \; P = \begin{pmatrix} 0 & 1 & 2 \\ 1 & 2 & 3 \end{pmatrix}, \tag{5.2}$$

whose corresponding 3-by-3 Gram matrix J is of rank 2

$$J = P^T P = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 8 \\ 3 & 8 & 13 \end{pmatrix}. \tag{5.3}$$

The following three different missing entry patterns demonstrate that a matrix may either have a unique completion, a finite number of possible completions, or an infinite number of possible completions:

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 8 \\ 3 & 8 & ? \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & ? \\ 2 & 5 & 8 \\ ? & 8 & 13 \end{pmatrix}, \quad \begin{pmatrix} 1 & 2 & 3 \\ 2 & ? & 8 \\ 3 & 8 & ? \end{pmatrix}. \tag{5.4}$$

For the left matrix, the missing diagonal element is uniquely determined by the fact that $\det J = 0$. For the middle matrix, the vanishing determinant constraint is a quadratic equation in the missing entry and there are *two* different possible completions (the reader may check that $J_{13} = J_{31} = 3.4$ is a valid completion). For the right matrix, the vanishing determinant constraint is a single equation for the two unknown diagonal elements, and so there is an infinite number of possible completions. We want to go beyond 3-by-3 matrices and develop techniques for the analysis of much larger matrices with arbitrary patterns of missing entries.

5.1.1 The completion matrix and local completion

We now adapt some of the the rigidity theory tools introduced in Chapter 1 to the matrix completion problem. We start by considering motions $p_i(t)$ that preserve the inner products $J_{ij} = p_i^T p_j$ for all $(i,j) \in E$. Differentiating (5.1) with respect to t yields the set of $m = |E|$ linear equations for the unknown velocities

$$p_i^T \dot{p}_j + p_j^T \dot{p}_i = 0, \quad \text{for all } (i,j) \in E. \tag{5.5}$$

This linear system can be brought together as $C_G(p)\dot{p} = 0$, where $C_G(p)$ is an $m \times dn$ coefficient matrix which we call the *completion matrix*. The completion matrix is sparse as it has only $2d$ non-zero elements per row. The locations of the non-zero entries are solely determined by the graph G, while their values are determined by the particular realization p. The solution space of (5.5) is at least $d(d-1)/2$ dimensional, due to orthogonal transformations that preserve inner products. Indeed, substituting into (5.5) the ansatz $\dot{p}_i = Ap_i$ (for all $i = 1, \ldots, n$) with A being a constant $d \times d$ matrix yields $p_i^T(A + A^T)p_j = 0$. Therefore, any choice of a skew-symmetric matrix $A = -A^T$ leads to a possible solution. We refer to these as the trivial infinitesimal motions. The number of trivial degrees of freedom in the completion problem is $d(d-1)/2$ which differs from its rigidity theory counterpart, because translations preserve distances but not inner products.

The rank of the completion matrix will help us determine if a unique completion of the Gram matrix J is possible. Indeed, if

$$\dim \operatorname{null}(C_G(p)) > d(d-1)/2, \tag{5.6}$$

that is, if the dimensionality of the null space of $C_G(p)$ is greater than $d(d-1)/2$, then there exist non-trivial solutions to (5.5). In other words, there are non-trivial infinitesimal motions that preserve the inner products. The rank of the completion matrix thus determines if the framework is *infinitesimally completable*. The counterpart of Asimow-Roth theorem would imply that there exists a non-trivial transformation that preserves the inner products and that the matrix is not *generically locally completable*. The rank of the completion matrix equals the size of its maximal non-zero minor. The minors are polynomials with integer coefficients of the coordinates of the configuration p. Such polynomials are either zero for all reals or they define an algebraic variety of singular configurations for which they are zero and are non-zero on the complement, which is an open dense subset of $\mathbb{R}^{dn}$. For that reason, almost all completion matrices for a given graph have the same rank and generic local completion is a property of the graph itself, and we do not need any advance knowledge of the realization of the matrix that we are trying to complete. Instead, we can simply construct a completion matrix from a randomly chosen realization. With probability one, the dimensionality of the null space of the randomized completion matrix will be the same as that of the completion matrix of the true realization, and this rank will determine if the Gram matrix is generically locally completable or not. The resulting randomized algorithm for testing local completion is along the same lines of the randomized algorithms for testing local rigidity [44, 55]:

In order for Algorithm 2 to be feasible for large-scale problems, the approach sketched above requires a fast method to determine the existence of a non-trivial infinitesimal motion, which is equivalent to checking that the rank of the completion matrix satisfies $\operatorname{rank}(C_G(p)) <$

Algorithm 2 Local completion of $n \times n$ rank-d Gram matrices

Require: Graph $G = (V, E)$ with n vertices and m edges corresponding to known matrix entries (self loops are possible).

1: Randomize a realization $p_1, \ldots, p_n$ in $\mathbb{R}^d$.
2: Construct the sparse completion matrix $C_G(p)$ of size $m \times dn$.
3: Check if there is a non-trivial infinitesimal motion $\dot{p}$ satisfying $C_G(p)\dot{p} = 0$.
4: If a non-trivial infinitesimal motion exists then G cannot be locally completable, otherwise G is locally completable.

$dn - d(d-1)/2$. This is not a straightforward check from the numerical linear algebra point of view. Note that the rank of a matrix can be determined only up to some numerical tolerance (like machine precision), because the matrix may have arbitrarily small non-zero singular values. The full singular value decomposition (SVD) is the most reliable way to compute the rank of the completion matrix (e.g., using MATLAB's rank function), but is also the most time consuming and is computationally prohibitive for large matrices. The completion matrix is often sparse, in which case, sparse LU, sparse QR and rank-revealing factorizations [47, 33, and references within] are much more efficient. However, due to non-zero fill-ins, such methods quickly run out of memory for large scale problems. Our numerical experimentation with Gotsman and Toledo's sparse LU MATLAB function nulls [47] and Davis' SuiteSparseQR MATLAB library and spqr function [33] encountered some memory issues for $n \geq 6000$. Therefore, for large-scale sparse completion matrices we use iterative methods that converge fast and do not have special storage requirements. The iterative procedure has several steps:

1. Use different choices of skew-symmetric $d \times d$ matrices A to construct $d(d-1)/2$ linearly independent trivial infinitesimal motions (recall that trivial motions are given by $\dot{p}_i = Ap_i$, $i = 1, \ldots, n$), and store the trivial motions in a $dn \times d(d-1)/2$ matrix T.

2. Compute the QR factorization of $T = QR$ (see, e.g., [43]) such that the columns of the $dn \times d(d-1)/2$ matrix Q form an orthogonal basis for the subspace of trivial motions, i.e., the two column spaces are the same $\mathrm{col}(Q) = \mathrm{col}(T)$ and $Q^TQ = I_{d(d-1)/2}$.

3. Randomize a unit size vector $b \in \mathbb{R}^{dn}$ in the orthogonal subspace of trivial motions, $b \in \mathrm{col}(Q)^\perp$. This is performed by randomizing a vector $v \in \mathbb{R}^{dn}$ with i.i.d standard Gaussian entries ($v_i \sim \mathcal{N}(0,1)$), projecting v onto the orthogonal subspace using $w = (I - QQ^T)v$ (the matrix QQ^T is never formed), and normalizing $b = w/\|w\|$. It is easy to check that $Q^Tb = 0$ and that b has the desired normal distribution.

4. Attempt solving the linear system $C_G(p)^Tx = b$ for the unknown x using an iterative method such as LSQR [79] that minimizes the sum of squares residual. The linear system may or may not have a solution. Numerically, a tolerance parameter tol must be supplied to the LSQR procedure. Set the tolerance parameter to be $tol = \varepsilon n^{-1/2}$, with a small ε, e.g., $\varepsilon = 10^{-4}$. If the residual cannot be made smaller then tol then

conclude that the linear system has no solution. In such a case $b \notin \mathrm{col}(C_G(p)^T)$ and by the fundamental theorem of linear algebra it follows that the projection of b onto $\mathrm{null}(C_G(p))$ is non-zero. Since b is orthogonal to all trivial motions, it follows that $\mathrm{null}(C_G(p))$ contains a non-trivial infinitesimal motion and the matrix is not locally completable. On the other hand, if the linear system $C_G(p)x = b$ has a solution in the sense that the residual is smaller than $\varepsilon n^{-1/2}$ then the projection of b on $\mathrm{null}(C_G(p))$ is smaller than $\varepsilon n^{-1/2}$. If a non-trivial infinitesimal motion $\dot{p}$ exists, then the projection of b onto it is normally distributed with zero mean and variance n^{-1}, that is $b^T\dot{p}/\|\dot{p}\| \sim \mathcal{N}(0, n^{-1})$. Therefore, with probability at least $1 - \dfrac{2\varepsilon}{\sqrt{2\pi}}$ all infinitesimal motions are trivial.

The LSQR procedure consists of applying the sparse completion matrix C_G and its transpose C_G^T to vectors with no special need for storage. The number of non-zero entries in the completion matrix is $2dm$ which is also the computational cost of applying it to vectors. The number of LSQR iterations depends on the non-zero singular values and in particular the ratio of the largest singular value and the smallest non-zero singular value (condition number). Arbitrarily small singular values may cause our iterative algorithm with its preset tolerance to fail. In practice, however, at least for moderate values of n, the full SVD revealed that such small singular values are rare.

We conclude this section by general remarks on the iterative procedure described above. First, note that the same iterative procedure can be used to determine local rigidity of bar and joint frameworks, and perhaps it can also be useful in other applications where existence of non-trivial null space vector is sought to be determined. Second, iterative methods can be often accelerated by a proper choice of a preconditioner matrix. This leads to the interesting question of designing a suitable preconditioner for completion and rigidity matrices, which we defer for future investigation.

5.1.2 Global completion and stress matrices

Generically local completion of a framework means that the realization cannot be continuously deformed while satisfying the inner product constraints. However, as the middle matrix in example (5.4) shows, local completion does not exclude the possibility of having a non-trivial discontinuous deformation that satisfies the inner product constraints, where by non-trivial we mean that the deformation is not an orthogonal transformation. We say that the framework is *globally completable* if the only deformations that preserve the inner products are the trivial orthogonal transformations (rotations and reflections). While local completion allows for a finite number of different completions, global completion is a stronger property that certifies that completion is unique.

A *completion stress* ω for a framework is an assignment of weights ω_{ij} on the edges of the graph such that for every vertex $i \in V$

$$\sum_{j:\,(i,j)\in E} \omega_{ij} p_j = 0. \tag{5.7}$$

Equivalently, a completion stress ω is a vector in the left null space of the completion matrix $C_G(p)$, i.e., $C_G(p)^T\omega = 0$. A stress matrix Ω is a symmetric $n \times n$ matrix obtained by the following rearrangement of the completion stress vector entries: $\Omega_{ij} = w_{ij}$ for $(i,j) \in E$, and $\Omega_{ij} = 0$ for $(i,j) \notin E$.

It follows from (5.7) that if ω is a stress for the framework of $p_1, \ldots, p_n$ then it is also a stress for the framework of $Ap_1, \ldots, Ap_n$, where A is any $d \times d$ linear transformation. In other words, the d coordinate vectors and their linear combinations are in the null space of the stress matrix Ω and $\dim \mathrm{null}(\Omega) \geq d$.

In the rigidity case, Theorem 1.2.2 by Connelly gives a sufficient condition for generic global rigidity, while the recently proved Theorem 1.2.3 by Gortler, Healy, and Thurston provides the necessary condition. Based on Theorems 1.2.2 and 1.2.3 a randomized algorithm for testing global rigidity was suggested in [44].

However, unlike the local rigidity theory that goes through without too much difficulty to the completion case, the global theory turns out to be more subtle. Perhaps the main difference between the two cases is that while prescribing the inner product between two points does not prevent them from getting arbitrarily far apart, it is obviously not the case when fixing the distance between the points. Despite this difference, it seems that the Theorem 1.2.2 by Connelly can be naturally translated to give sufficient conditions for global completion. Since the proof of this claim is outside the scope of this paper, it will not be given here, and the reader may want to consider the last statement as a conjecture. As for necessity, we did not find an obvious counterpart of Theorem 1.2.3 by Gortler, Healy and Thurston that would give necessary conditions for global completion. In fact, it is possible to come up with examples of generically globally completable frameworks with stresses whose null space is of dimension greater than d. For example, the first missing entry pattern in (5.4) (the 3×3 matrix with a single missing diagonal entry) was noted to be globally completable for $d = 2$ (by the vanishing determinant consideration), but it has no non-trivial stresses (the stress must satisfy $w_{3,3} = 0$ due to the location of the missing entry, and from linear independence of p_1 and p_2 in $\mathbb{R}^2$ it follows that $w_{3,1} = w_{3,2} = 0$, and by similar considerations all the other entries of the stress must vanish). The dimension of the null space of the zero stress matrix is 3, which is strictly greater than $d = 2$.

Having these considerations in mind, we propose Algorithm 3, which is a randomized algorithm for testing sufficiency conditions for global completion. That is, the output of Algorithm 3 is one of two possibilities: 1) G is generically globally completable, or 2) global completion of G is undecided. Algorithm 3 cannot be used to conclude that a matrix is not globally completable.

Algorithm 3 also uses iterative methods in order to insure its scalability to large scale matrices. In stage (4) of Algorithm 3 we use LSQR, which when initialized with a random starting vector converges to a random left null space solution, rather than to the zero vector. Here we use LSQR with a very small tolerance as we are guaranteed the existence of a non-trivial stress.

In stage (6) we again apply LSQR, this time similarly to the way it is applied in the local completion case (Algorithm 2). Specifically, we first find a random vector b which is perpendicular to the subspace of coordinate vectors and then we try to solve $\Omega x = b$. Here Ω is symmetric (compared to C_G). For moderate scale problems (e.g., $n \leq 5000$) we use sparse QR (SPQR) to compute the rank of Ω as it runs faster than LSQR, but cannot handle

Algorithm 3 Global completion of $n \times n$ rank-d Gram matrices

Require: Graph $G = (V, E)$ with n vertices and m edges corresponding to known matrix entries (self loops are possible).

1: Check local completion using Algorithm 2. Proceed only if framework is locally completable.
2: Randomize a realization $p_1, \ldots, p_n$ in $\mathbb{R}^d$.
3: Construct the sparse completion matrix $C_G(p)$ of size $m \times dn$.
4: Compute a random completion stress vector ω in the left null space of $C_G(p)$ satisfying $C_G(p)^T\omega = 0$.
5: Rearrange ω into a completion stress matrix Ω.
6: Check if the null space of Ω contains vectors which are not linear combinations of the d coordinate vectors.
7: If no such other null space vectors exist (i.e., if $\dim \mathrm{null}(\Omega) = d$) then G is globally completable; otherwise global completion of G is undecided.

much larger values of n due to memory problems (fill-ins).

Before moving to the completion problem of general rectangular matrices, we comment on two other aspects of the Gram matrix case. First, note that not every partially observed matrix can be completed into a positive semidefinite matrix. For example, if all diagonal entries are given and are set to be negative (e.g., $J_{ii} = -1$ for $i = 1, \ldots, n$) then the matrix cannot be completed to a semidefinite positive matrix, because its trace is negative so at least one of its eigenvalues must be negative. If the matrix can be completed to some positive semidefinite matrix and the number of observed entries $m \leq \binom{r+2}{2} - 1$, then from [9, Theorem 1.1] it follows that the matrix can be completed to a matrix of rank r (or lower). Second, when the diagonal entries of the Gram matrix are known, then both local and global completion problems are equivalent to the standard rigidity and global rigidity problem. The graph is essentially the cone on the graph [28] determined by the off-diagonal entries, since the diagonal elements give the distances of the vertices from the origin (the extra vertex), while the off-diagonal inner products are translated into distances using the diagonal entries. The generic rigidity and generic global rigidity of a graph in dimension d is equivalent to the generic rigidity and generic global rigidity of the cone on the graph in dimension $d + 1$.

5.2 General rectangular low rank matrices

A general $n_1 \times n_2$ rank-d matrix X can be written as $X = UV^T$ where U is $n_1 \times d$ and V is $n_2 \times d$ given by $U = \begin{bmatrix} u_1 & u_2 & \cdots & u_{n_1} \end{bmatrix}^T$ and $V = \begin{bmatrix} v_1 & v_2 & \cdots & v_{n_2} \end{bmatrix}^T$, where the $n_1 + n_2$ vectors $u_1, \ldots, u_{n_1}, v_1, \ldots, v_{n_2}$ are in $\mathbb{R}^d$. Entries of X are inner products of these vectors

$$X_{ij} = u_i^T v_j. \tag{5.8}$$

The observed entries X_{ij} define a bipartite graph $G = (V, E)$ with $n_1 + n_2$ vertices as we never observe inner products of the form $u_i^T u_{i'}$ or $v_j^T v_{j'}$. Differentiation of (5.8) with

respect to t yields the set of linear equations

$$u_i^T \dot{v}_j + v_j^T \dot{u}_i = 0, \quad (i,j) \in E \tag{5.9}$$

for the unknown velocities $\dot{u}_i$ and $\dot{v}_j$. The corresponding completion matrix $C_G(u,v)$ has m rows and $d(n_1+n_2)$ columns, but only $2d$ non-zero elements per row.

The decomposition $X = UV^T$ is not unique: if W is an invertible $d \times d$ matrix then $X = U_W V_W^T$ with $U_W = UW$ and $V_W = V(W^{-1})^T$. It follows that there are d^2 degrees of freedom in choosing W, because the general linear group $GL(d,\mathbb{R})$ of $d \times d$ invertible matrices is a Lie group over $\mathbb{R}$ of dimension d^2. Indeed, substituting in (5.9) the ansatz $\dot{u}_i = Au_i$ (for $i = 1,\ldots,n_1$) and $\dot{v}_j = Bv_j$ (for $j = 1,\ldots,n_2$) gives $u_i^T(B + A^T)v_j = 0$, which is trivially satisfied whenever $B = -A^T$. These are the trivial infinitesimal motions and they span a subspace of dimension d^2.

Similarly to local rigidity and local completion of Gram matrices, a condition for local completion for general rectangular matrices is $\dim \operatorname{null}(C_G(u,v)) = d^2$. If the dimension of the null space of the completion matrix is greater than d^2 then there are non-trivial deformations that preserve all observed elements of the matrix, where a trivial deformation is any invertible linear transformation of $\mathbb{R}^d$. The local completion testing algorithm for general rectangular matrices is given in Algorithm 4.

Algorithm 4 Local completion of $n_1 \times n_2$ rank-d matrices

Require: Bipartite graph $G = (V,E)$ with $n_1 + n_2$ vertices and m edges corresponding to known matrix entries.

1: Randomize a realization $u_1,\ldots,u_{n_1}$ and $v_1,\ldots,v_{n_2}$ in $\mathbb{R}^d$.
2: Construct the sparse completion matrix $C_G(u,v)$ of size $m \times d(n_1+n_2)$.
3: Check if there is a non-trivial infinitesimal motion $\dot{p}$ satisfying $C_G(p)\dot{p} = 0$, where there are d^2 trivial motions.
4: If a non-trivial infinitesimal motion exists then G cannot be locally completable, otherwise G is locally completable.

The implementation of the steps in Algorithm 4 is very similar to the implementation of the steps in Algorithm 2 for testing local completion of Gram matrices.

As in the Gram case, local completion does not imply global completion. A stress ω for the general rectangular matrix case is an assignment of weights ω_{ij} on the edges of the bipartite graph that satisfy

$$\sum_{j:\,(i,j)\in E} \omega_{ij} v_j = 0, \quad \text{for all } i = 1,\ldots,n_1, \tag{5.10}$$

and

$$\sum_{i:\,(i,j)\in E} \omega_{ij} u_i = 0, \quad \text{for all } j = 1,\ldots,n_2. \tag{5.11}$$

The stress ω is in the left null space of the completion matrix, i.e., $C_G(u,v)^T\omega = 0$.

The stress weights can be viewed as the entries of an $n_1 \times n_2$ matrix $\tilde{\Omega}$ that satisfies $\tilde{\Omega}V = 0$ and $\tilde{\Omega}^T U = 0$, therefore

$$\operatorname{rank}\tilde{\Omega} = \operatorname{rank}\tilde{\Omega}^T \leq \min(n_1,n_2) - d. \tag{5.12}$$

The stress matrix is the $(n_1 + n_2) \times (n_1 + n_2)$ symmetric matrix Ω defined via

$$\Omega = \begin{pmatrix} 0 & \tilde{\Omega} \\ \tilde{\Omega}^T & 0 \end{pmatrix}, \tag{5.13}$$

and from (5.12) it follows that $\operatorname{rank} \Omega \leq 2 \min(n_1, n_2) - 2d$, or equivalently

$$\dim \operatorname{null}(\Omega) \geq n_1 + n_2 - (2 \min(n_1, n_2) - 2d) = 2d + |n_1 - n_2|. \tag{5.14}$$

The notion of a shared stress kernel was recently defined in [44, Section 4] as the intersection of all stress kernels, and it was used to prove a stronger version of Connelly's sufficient condition for global rigidity (Theorem 1.2.2). Shared stress kernels seem to play a crucial role in the theory of global completion of rectangular matrices. In particular, it is conjectured that a sufficient condition for global completion of rectangular matrices is that the dimension of the shared stress kernel is $2d$, which is in general smaller than $2d + |n_1 - n_2|$ that appears in (5.14). As with the Gram matrix case, we have not been able to conjecture necessary conditions for global completion of rectangular matrices. In particular, having a shared stress kernel of dimension $2d$ is not a necessary condition, as there exist counterexamples of globally completable rectangular matrices with a shared stress kernel of dimension strictly greater than $2d$. As an example[1], consider a 4×4 general matrix with the following pattern of missing entries (marked by ?)

$$X = \begin{pmatrix} * & ? & * & * \\ ? & * & * & * \\ ? & * & * & * \\ * & * & * & * \end{pmatrix}. \tag{5.15}$$

This matrix is globally completable as a rank-2 matrix (the three missing entries can be completed using the vanishing determinant consideration: the minor $M_{2,1}$ gives the missing value $X_{1,2}$, $M_{3,2}$ gives $X_{2,1}$, and $M_{2,2}$ gives $X_{3,1}$). The matrix $\tilde{\Omega}$ must be a rank-1 matrix of the form

$$\tilde{\Omega} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix} = ab^T, \tag{5.16}$$

with $a = (0, a_2, a_3, a_4)^T$ and $b = (0, b_2, b_3, b_4)^T$. Indeed, the missing entries $X_{2,1}$ and $X_{3,1}$ imply that $w_{2,1} = w_{3,1} = 0$, and then the linear independence of u_1 and u_4 in $\mathbb{R}^2$ yields $w_{1,1} = w_{4,1} = 0$; similarly $w_{1,2} = 0$ due to the missing entry $X_{1,2}$, and $w_{1,3} = w_{1,4} = 0$ follows from the linear independence of v_3 and v_4. The vectors a and b are determined uniquely, up to scale, by the linear dependencies of the vectors u_2, u_3, u_4 and v_2, v_3, v_4, such that $b^T V = 0$ and $a^T U = 0$. From $\operatorname{rank}(\Omega) = 1$ it follows that $\operatorname{rank}(\tilde{\Omega}) = 2$ and $\dim \operatorname{null}(\Omega) = 6 > 4 = 2d$.

Following the randomized algorithms for global completion of Gram matrices, Algorithm 5 is a randomized algorithm for testing sufficient conditions for global completion of rectangular matrices. The shared stress kernel check is made by picking l different random

Algorithm 5 Global completion of $n_1 \times n_2$ rank-d matrices

Require: Bipartite graph $G = (V, E)$ with $n_1 + n_2$ vertices and m edges corresponding to known matrix entries.

1: Check local completion using Algorithm 4. Proceed only if framework is locally completable.
2: Randomize a realization $u_1, \ldots, u_{n_1}$ and $v_1, \ldots, v_{n_2}$ in $\mathbb{R}^d$.
3: Construct the sparse completion matrix $C_G(u, v)$ of size $m \times d(n_1 + n_2)$.
4: Compute $l \geq 1$ random completion stress vectors $\omega_1, \ldots, \omega_l$ in the left null space of $C_G(u, v)$ satisfying $C_G(u, v)^T \omega_i = 0$ ($i = 1, \ldots, l$).
5: Rearrange $\omega_1, \ldots, \omega_l$ into $(n_1 + n_2) \times (n_1 + n_2)$ symmetric completion stress matrices $\Omega_1, \ldots, \Omega_l$ and stack them into a single matrix Ω of size $l(n_1 + n_2) \times (n_1 + n_2)$ given by $\Omega = (\Omega_1 \Omega_2 \cdots \Omega_l)^T$.
6: Check if the null space of Ω contains a vector outside the column space of $\begin{pmatrix} U & 0 \\ 0 & V \end{pmatrix}$ (equivalently, check if the null space of Ω contains more than $2d$ linearly independent vectors).
7: If $\dim \operatorname{null}(\Omega) = 2d$ then G is globally completable, otherwise ($\dim \operatorname{null}(\Omega) > 2d$) global completion of G is undecided.

stresses (the parameter l should be set by the user) and stacking the different stress matrices in one matrix whose kernel is evaluated.

We remark that Bolker and Roth [18] derived stress matrices of the same form as the stress matrices in (5.13) in their characterization of rigid (complete) bipartite graphs.

5.3 Combinatorial approach for local and global completion

In the previous sections we observed that the rank of the completion matrices can determine generically local rank-d completion properties of a given graph. These observations led to practical algorithms for property testing, but perhaps there is a simple combinatorial characterization of locally and/or globally completable graphs? In rigidity theory, locally and globally rigid graphs have a simple combinatorial characterization in one and two dimensions (see Theorems 1.1.2 and 1.2.1 by Laman and Hendrickson). In higher dimensions, however, such combinatorial characterizations are still missing, and only necessary conditions are available. We therefore believe that exact combinatorial characterization of local and global completion of rank-d matrices for $d \geq 3$ is currently out of reach, and focus our attention to rank-1 and rank-2 matrices and to finding only necessary conditions for $d \geq 3$. We begin by observing some properties of locally completable graphs.

[1] The example is due to Dylan Thurston

Proposition 5.3.1. *In any Gram locally completable rank-d matrix J of size $n \times n$ with $n \geq d$, one can delete entries until the resulting matrix is locally completable and has exactly $dn - \frac{d(d-1)}{2}$ entries.*

Proof. The completion matrix of the system of equations (5.5) $p_i^T \dot{p}_j + p_j^T \dot{p}_i = 0$ for all $(i,j) \in E$ must have rank $dn - \frac{d(d-1)}{2}$, and thus one can drop linearly dependent equations until only $dn - \frac{d(d-1)}{2}$ equations remain. □

Proposition 5.3.2. *Any Gram locally completable rank-d matrix J with $|E(G_J)| = dn - \frac{d(d-1)}{2}$ has the property that for any $n' \times n'$ submatrix J' of J, where $d \leq n' \leq n$, it holds true that $|E(G_{J'})| \leq dn' - \frac{d(d-1)}{2}$.*

Proof. For every $(i,j) \in E(G_J)$ there corresponds an equation $p_i^T \dot{p}_j + p_j^T \dot{p}_i = 0$. Given our interpretation of local completion and Proposition 5.3.1, these equations are linearly independent. If for some $n' \times n'$ submatrix J' of J, with $n' \geq d+1$ we would have that the number of edges $|E(G_{J'})| > dn' - \frac{d(d-1)}{2}$, then by the same Proposition 5.3.1 there would be dependence among the corresponding $dn' - \frac{d(d-1)}{2}$ equations, which is a contradiction. □

The result above implies that every locally completable Gram matrix contains an underlying graph that spans all vertices which is $(d, \frac{d(d-1)}{2})$-tight sparse. It is natural to ask whether any $(d, \frac{d(d-1)}{2})$-tight sparse graph is Gram locally completable. The answer to this question turns out to be false even in one dimension, that is, not every $(1,0)$-tight sparse graph is Gram locally completable for $d = 1$.

Proposition 5.3.3. *A graph G is minimally Gram locally completable in one dimension if and only if each connected component of G is a tree plus one edge that contains an odd cycle.*

Proof. First, suppose $G = (V, E)$ is minimally Gram locally completable for $d = 1$. Proposition 5.3.1 implies that G has no redundant edges so $|E(G)| = |V(G)|$. Moreover, Proposition 5.3.2 implies that each connected component H of G satisfies $|E(H)| = |V(H)|$, from which it follows that each connected component is a tree plus one edge. We show

that the cycle formed by the extra edge must be of odd length. Let L be the length of the cycle and w.l.o.g let its vertices be $\{1, \ldots, L\}$ with the corresponding framework points $p_1, \ldots, p_L$. The linear system (5.5) attached to the cycle edges is given by

$$p_i \dot{p}_{i+1} + p_{i+1} \dot{p}_i = 0, \ i = 1, \ldots, L, \tag{5.17}$$

(with the convention that $L + 1$ is 1). The system (5.17) results in the coefficient matrix

$$C_L = \begin{pmatrix} p_2 & p_1 & 0 & 0 & 0 & 0 \\ 0 & p_3 & p_2 & 0 & 0 & 0 \\ 0 & 0 & \ddots & \ddots & 0 & 0 \\ 0 & 0 & 0 & p_{L-1} & p_{L-2} & 0 \\ 0 & 0 & 0 & 0 & p_L & p_{L-1} \\ p_L & 0 & 0 & 0 & 0 & p_1 \end{pmatrix}$$

Expanding by the first column, we get that

$$\det C_L = p_2 \prod_{i \neq 2} p_i + (-1)^{L+1} p_L \prod_{i \neq L} p_i = \left[1 + (-1)^{L+1}\right] \prod_{i=1}^{L} p_i.$$

For L even the determinant vanishes, from which it follows that there is a linear dependence (redundancy) among the equations and G cannot be minimally locally completable. Therefore, the length of the cycle must be odd.

Conversely, if each connected component of G is a tree plus one edge that forms an odd cycle, then from $\det C_L \neq 0$ it follows that the only solution to (5.17) is the trivial one: $\dot{p}_1 = \dot{p}_2 = \ldots = \dot{p}_L = 0$. From the connectivity of the connected component we get that all velocities must vanish, that is, the only solution to $C_G(p)\dot{p} = 0$ is $\dot{p} = 0$ and G is locally completable. Since each connected component is a tree plus one edge it follows that $|E(G)| = |V(G)|$ which means that G is also minimally locally completable. □

We remark that Propositions 5.3.1 and 5.3.2 implicitly define a matroid on the complete graph of n vertices with a loop at each vertex. The independent sets of the matroid correspond to sets of linearly independent rows of the completion matrix, and the rank of the matroid is $dn - \frac{d(d-1)}{2}$. As a quick illustration of Proposition 5.3.3, consider the complete

graph on two vertices. Here the (matroid) bases are either two loops (two diagonal entries), or one loop and the edge between the vertices (one diagonal entry and the off diagonal entry). Either way, the cycle is of odd length.

We now turn to the combinatorial characterization of global completion for Gram matrices in one dimension. Note that the inner products accessible to the analyst are simply products of the form $p_i p_j = J_{ij}$ for $(i,j) \in E$. Taking the logarithm of the modulus gives the linear system

$$q_i + q_j = \log |J_{ij}|, \quad (i,j) \in E, \tag{5.18}$$

where $q_i = \log |p_i|$. This linear system has much resemblance with the linear system (5.5) of $C_G(p)\dot{p} = 0$. Indeed, by substituting $p_1 = p_2 = \ldots = p_n = 1$ in $C_G(p)$ the two coefficient matrices of (5.5) and (5.18) are the same. The only difference between to the systems is that (5.18) is non-homogenous. Therefore, by following the steps of the proof of Proposition 5.3.3 it follows that if G is minimally locally completable then the system (5.18) has a unique solution. That is, if each connected component of G is a tree plus one edge that forms an odd cycle, then all the $|p_i|$'s are uniquely determined. This leaves only the signs of the $p_i's$ undetermined. Clearly, if G has more than one connected component then the relative sign between coordinates of different components cannot be determined. Therefore, a globally completable graph must be connected. On the other hand, if G is a single tree plus an edge that forms an odd cycle then by similar considerations all bit signs $b_i = (1 - \operatorname{sign} p_i)/2 \in \{0,1\}$ are determined up to overall negation (reflection), because they satisfy a similar linear system over Z_2

$$b_i \oplus b_j = (1 - \operatorname{sign}(J_{ij}))/2, \quad (i,j) \in E. \tag{5.19}$$

This is summarized in the following proposition:

Proposition 5.3.4. *A graph is minimally Gram globally completable in one dimension if and only if it is a tree plus an edge that forms an odd cycle.*

We now give the combinatorial characterization of local and global completion for the case of a general rectangular rank-1 matrices. The proofs are very similar to the Gram case and are therefore omitted. The only differences being the number of degrees of freedom (d^2 instead of $d(d-1)/2$, which for $d = 1$ are 1 and 0, respectively) and the fact that the underlying graph is bipartite.

Proposition 5.3.5. *A rectangular graph is minimally locally completable in one dimension if and only if it is a forest (a disjoint union of trees, or equivalently, a $(1,1)$-tight sparse bipartite graph).*

Proposition 5.3.6. *A rectangular graph is minimally globally completable in one dimension if and only if it is a tree.*

We note that the conditions for local and global completion in one dimension are much weaker than the condition for local and global rigidity in one dimension, which are connectedness and 2-connectedness, respectively. Laman theorem for two-dimensional rigidity leads us to speculate that the combinatorial characterization of Gram and rectangular rank-2 matrices will perhaps involve $(2,1)$-sparse graphs and bipartite $(2,4)$-sparse graphs. However, we currently postpone the investigation of this interesting question.

5.4 Numerical Simulations

To illustrate the applicability of the above algorithms for testing low rank matrix completion in practice, we present the outcomes of several numerical experiments for both Gram and general rectangular rank-d matrices. Let us first investigate the case when J is a Gram matrix of size $n \times n$. We apply the local completion Algorithm 2 to test the local completion property of random graphs where each edge (including self loops) is chosen independently with probability β, such that the expected number of edges is $\mathbb{E}[m] = \binom{n+1}{2}\beta$. The conditional probability $f(n,d,\beta)$ for an $n \times n$ rank-d Gram matrix to be locally completable

$$f(n,d,\beta) = \Pr\{G \text{ is Gram } n \times n \text{ rank-}d \text{ locally completable} \,|\, \beta\} \tag{5.20}$$

is a monotonic function of β. We define the threshold value $\beta^*(n,d)$ as the value of β for which $f(n,d,\beta^*) = 1/2$. The theoretical bound of Candès and Recht [23] implies $\beta^*(n,d) \sim C(d)n^{-0.8}\log n$ (CR), while that of Keshavan, Oh, and Montanari [69, 68] gives $\beta^*(n,d) \sim C(d)n^{-1}\log n$ (KOM) for large n.

We now describe the numerical simulation and estimation procedure for the threshold values. The goal is to infer the asymptotic behavior of $\beta^*(n,d)$ for large values of n. For small values of n the running time of the completion Algorithm 2 is not an issue and it is possible to perform an exhaustive search for the threshold value. However, for large values of n (e.g., $n = 20000$) an exhaustive search is too time consuming.

The way we choose to accelerate the search is by using logistic regression. Suppose that after the $(i-1)$'th iteration we were able to estimate the threshold value for $n_{i-1} \times n_{i-1}$ matrices, and now we want to find the threshold for larger matrices of size $n_i \times n_i$ with $n_i > n_{i-1}$. From the previous threshold estimate $\beta^*(n_{i-1},d)$ we can easily guess a reasonable upper bound β_i^u such that with probability one $\beta^*(n_i,d) \le \beta_i^u$ (e.g., $\beta_i^u = \beta^*(n_{i-1},d)$, because β^* is decreasing with n). Starting from the upper bound $\beta = \beta_i^u$ we decrease β in small steps until we encounter enough matrices that are not completable anymore (say until we observe 20 consecutive non-completable matrices). This gives us a rough confidence interval for β^*. We then sub-sample this interval to get a more accurate estimation of the threshold value by collecting more β's and their corresponding binary responses $y \in \{0,1\}$, where $y = 1$ if the matrix is completable, and $y = 0$ otherwise. We perform a binomial logistic regression in order to estimate numerically the approximate value of β^*. Logistic regression is a model frequently used in statistics for prediction of the probability of occurrence of an event, by fitting data to a logistic curve. The logistic function $f(\beta)$

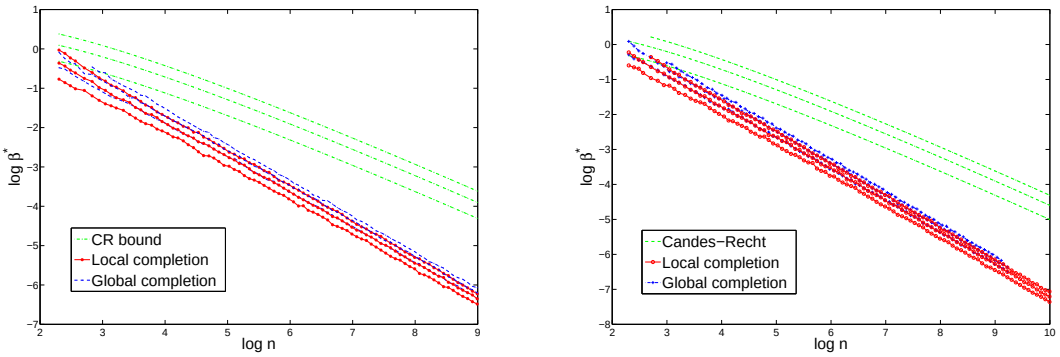


Figure 5.1: Local and global completion of rank-d Gram matrices (left) and rectangular matrices (right): the threshold β^* as a function of d and n for $d = 2, 3, 4$.

models the conditional probability (5.20) by

$$\Pr\{y = 1|\beta\} \approx f(\beta) = \frac{1}{1 + e^{-\alpha^*(\beta - \beta^*)}},$$

where β^* and α^* are parameters to be estimated from the set of samples of the explanatory variable β and the corresponding binary responses y. We typically used a total of about 120 (β, y) sample pairs for the logistic regression (80 samples to find the lower bound, and another 40 samples to improve the confidence intervals for the estimate). We performed similar searches and logistic regressions to find the threshold value for the sufficient condition for global completion of Gram matrices, as well as local and sufficient conditions for global completion of rectangular matrices.

Figure 5.1 is a log-log plot of the threshold β^* against n, for rank-d Gram matrices (left) and rectangular matrices (right), with $d = 2, 3, 4$. For "rectangular" matrices we chose for simplicity $n_1 = n_2 = n$ and run Algorithm 5 with $l = 1$. The dotted curves that appear in green are the CR bounds, the blue dashed curves represent global completion, while the remaining three curves denote local completion. It is interesting to note that both in the Gram case and in the rectangular case, the global completion curves for dimension d coincide with the local completion curves for dimension $d + 1$. This can perhaps be explained by the following theorem of Gortler, Healy, and Thurston [44, Theorem 1.17]: If a graph is generically locally rigid but not generically globally rigid in $\mathbb{R}^d$, then any generic framework can be connected to an incongruent framework by a path of frameworks in $\mathbb{R}^{d+1}$ with constant edge lengths. For example, the graph on 4 vertices with all possible edges except for one (K_4 minus an edge) is locally rigid in $\mathbb{R}^2$ but is not globally rigid in $\mathbb{R}^2$. This graph can be realized as two triangles with a common side and has two possible realizations in the plane. Though it is impossible to continuously deform the framework in the plane, it is possible to continuously deform it by leaving the plane into the three dimensional space by folding one triangle on top of the other. We may expect a similar phenomenon in the matrix completion problem: if a graph is not globally rank-d completable, then it is not locally rank-$(d + 1)$ completable.

Both the CR and the KOM bounds can be written as $\beta^* \sim C(d)n^\alpha \log n$ or upon taking the logarithm as $\log \beta^* \sim \alpha \log n + \log\log n + \log C(d)$. We therefore performed a simple linear regression in the Gram case with $d = 2$ of the form $\log \beta^*(n) = a_1 \log n + a_2 \log\log n + a_3$ to estimate a_1, a_2, a_3 (see Table 5.1). For local completion, the sampled n take values up to 100000 while for global completion up to 15000, in a geometric progression of rate 1.08. The coefficient a_2 may be expected to be 1 (the coupon collector problem) and Theorem 1.2.4 for planar rigidity [61] may even shed light on the higher order corrections. The results for different rank values $d = 2, 3, 4$ are summarized in Table 5.2. These results may be considered as a numerical evidence for the success of the KOM theoretical bound $C(d)n \log n$. The slight deviation of $a_1 + 2$ from unity can be explained by the small bias introduced by the small n values.

	Local Completion		Global Completion	
	Value	95% -conf. interval	Value	95% conf. interval
$a_1 + 2$	1.022	[0.99842, 1.0456]	0.99066	[0.94206 1.0393]
a_2	0.63052	[0.43626, 0.8247]	0.79519	[0.43446 1.1559]
a_3	0.90663	[0.69405, 1.1192]	0.99899	[0.6394 1.3586]
$a_1 + 2$	0.9773	[0.9748 0.9797]	0.9631	[0.95959 0.96668]
$a_2 = 1$	-	-	-	-
a_3	0.5039	[0.48295 0.5250]	0.7954	[0.76823 0.82258]

Table 5.1: Linear regression for local (left column) and global (right column) rank-2 Gram completion: $\log \beta^* = a_1 \log n + a_2 \log\log n + a_3$.

d	($a_1 + 2$,a_2,a_3)	($a_1 + 2$,1,a_3)
2	(1.022, 0.63052 , 0.90663)	(0.9773, 1, 0.5039)
3	(1.0423, 0.40053 , 1.394)	(0.9636, 1, 0.78538)
4	(1.0382, 0.35039 , 1.6725)	(0.95289, 1, 1.013)

Table 5.2: Linear regression for local Gram completion for $d = 2, 3, 4$.

The asymptotic behavior $\beta^* \sim C(d)dn^{-1} \log n$ implies that $\frac{\beta^* n}{d \log n}$ tends to a constant as $n \to \infty$. Figure 5.2 shows $\frac{\beta^* n}{d \log n}$ against $\log n$ for different values of d in the Gram case (left) and in the rectangular matrix case (right) from which the asymptotic behavior is clear.

Finally, in Tables 5.3 and 5.4 we list the running times in seconds for several of the numerical simulations. All computations were performed on a PC machine equipped with an Intel(R) Core(TM)2 Duo CPU 3.16GHz with 4 GB RAM. The matrices we considered were general "rectangular" n-by-n matrices of rank 2. We list the running times for different values of $\beta = \frac{m}{n^2}$ as a function of β^*, where m denotes the number of given entries in the matrix. The algorithm often runs faster as m (alternatively, β) increases, because the iterative LSQR method converges after a fewer number of iterations. Note that the values

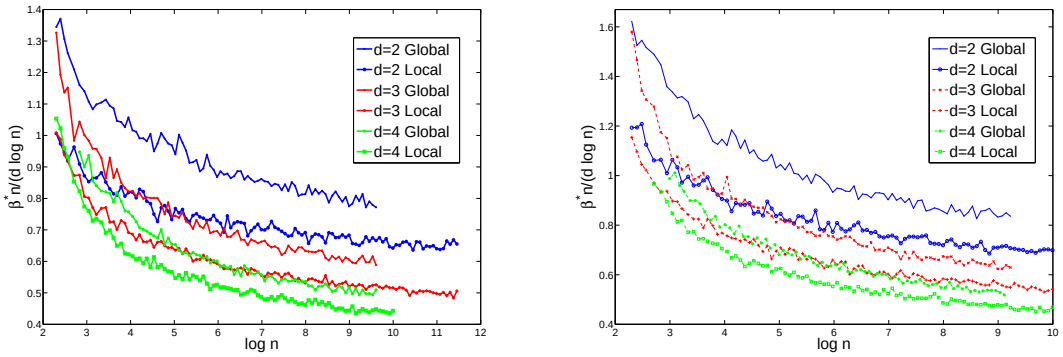


Figure 5.2: A plot of $\frac{\beta^* n}{d \log n}$ vs. $\log n$ for Gram matrices (left) and rectangular matrices (right). For large values of n we expect $\frac{\beta^* n}{d \log n}$ to go to the constant $C(d)$ of the KOM bound.

$\beta = 0.75\beta^*$ in the first column in Table 5.3 produce matrices that are not locally completable. Similarly, in Table 5.4, all values in the first column were chosen such that the matrix is locally completable, but not globally completable. In both tables, all the values $\beta = 2\beta^*$ produce matrices that are locally (and globally) completable. We do this to better illustrate the time difference between the case when the LSQR method converges after a certain relatively small number of iterations, as opposed to when the residual cannot be made smaller than the threshold.

n	β	m	T	β	m	T	β	m	T
101	$0.75\beta^*$	598	0.07	β^*	796	0.05	$2\beta^*$	1592	0.06
498	$0.75\beta^*$	3596	0.24	β^*	4795	0.32	$2\beta^*$	9590	0.25
1002	$0.75\beta^*$	7792	0.75	β^*	10390	0.85	$2\beta^*$	20781	0.72
2533	$0.75\beta^*$	21591	2.6	β^*	28788	2.3	$2\beta^*$	57575	2
5068	$0.75\beta^*$	45982	10.9	β^*	61309	7.3	$2\beta^*$	122620	10.7
10133	$0.75\beta^*$	99140	28.6	β^*	132190	12	$2\beta^*$	264370	18.2

Table 5.3: Running time T (in seconds) in the case of *local* completion of general n-by-n rectangular matrices of rank $d = 2$, for various values of m (number of entries revealed).

n	β	m	T	β	m	T	β	m	T
101	$0.8\beta^*$	800	0.17	β^*	999	0.14	$2\beta^*$	1998	0.14
498	$0.8\beta^*$	4638	1.3	β^*	5797	0.9	$2\beta^*$	11595	0.88
1002	$0.9\beta^*$	11539	2.2	β^*	12821	3.1	$2\beta^*$	25642	2.3
2533	$0.9\beta^*$	31078	12	β^*	34531	10.1	$2\beta^*$	69063	16.1
5068	$0.91\beta^*$	69020	45.2	β^*	75847	43	$2\beta^*$	151690	59.4
10133	$0.92\beta^*$	143790	179	β^*	156280	180	$2\beta^*$	312590	288

Table 5.4: Running time T (in seconds) in the case of *global* completion of general n-by-n rectangular matrices of rank $d = 2$, for various values of m (number of entries revealed).

5.5 Summary and Discussion

In this chapter we made the observation that the rank-d matrix completion problem is tightly related to rigidity theory in $\mathbb{R}^d$, with inner products replacing the role of distances. Many of the results in rigidity theory, both classical and recent, give new insights into the completion problem. In particular, we introduced the completion matrix that enables fast determination of the generic local completion property of a partially viewed matrix into a rank-d matrix. We used stresses to conjecture sufficient conditions for generic global completion. Our algorithms help to determine if a unique completion is possible without attempting to complete the entries of the matrix.

Most of the results in rigidity theory translate nicely into the completion setup. However, occasional differences between completion and rigidity lead to interesting difficulties and questions, such as what is the generalization of Laman's theorem to the completion case and what are the necessary conditions for global completion. Finally, we note that beyond its mathematical importance, rigidity theory is much useful in diverse applications such as scene analysis and localization of sensor networks, and some of the recent localization algorithms based on rigidity [91, 113] can be easily adjusted to the inner products completion setup.

Bibliography

[1] E. AB, A. R. ATKINSON, L. BANCI, I. BERTINI, S. CIOFI-BAFFONI, K. BRUNNER, T. DIERCKS, V. DOTSCH, F. ENGELKE, AND G. FOLKERS, *NMR in the SPINE structural proteomics project*, Acta Crystallogr D Biol Crystallogr, 62 (2006), pp. 1150–1161.

[2] B. D. O. ANDERSON, P. N. BELHUMEUR, T. EREN, D. K. GOLDENBERG, A. S. MORSE, AND W. WHITELEY, *Graphical properties of easily localizable networks*, Wireless Networks, 15 (2009), pp. 177–191.

[3] K. ARUN, T. HUANG, AND S. BOLSTEIN, *Least-squares fitting of two 3-D point sets*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 9 (1987), pp. 698–700.

[4] T. ASANO, P. BOSE, P. CARMI, A. MAHESHWARI, C. SHU, M. SMID, AND S. WUHRER, *A linear-space algorithm for distance preserving graph embedding*, Computational Geometry, 42 (2009), pp. 289–304.

[5] L. ASIMOW AND B. ROTH, *The rigidity of graphs*, Trans. Amer. Math. Soc., 245 (1978), pp. 279–289.

[6] J. ASPNES, T. EREN, D. K. GOLDENBERG, A. S. MORSE, W. WHITELEY, Y. R. YANG, B. D. O. ANDERSON, AND P. N. BELHUMEUR, *A theory of network localization*, IEEE Transactions on Mobile Computing, 5 (2006), pp. 1663–1678.

[7] J. ASPNES, D. K. GOLDENBERG, AND Y. R. YANG, *On the computational complexity of sensor network localization*, in Proceedings of Algorithmic Aspects of Wireless Sensor Networks: First International Workshop (ALGOSENSORS), Lecture Notes in Computer Science, Springer-Verlag, 2004, pp. 32–44.

[8] Z. BAI AND G. H. GOLUB, *Some unusual matrix eigenvalue problems*, in Proceedings of VECPAR'98 - Third International Conference for Vector and Parallel Processing, 1999.

[9] A. BARVINOK, *A remark on the rank of positive semidefinite matrices subject to affine constraints*, Discrete Comput. Geom., 25 (2001), pp. 23–31.

[10] A. BAX AND A. GRISHAEV, *Weak alignment NMR: a hawk-eyed view of biomolecular structure*, Curr Opin Struct Biol, 15 (2005), pp. 563–570.

[11] M. BELKIN AND P. NIYOGI, *Laplacian eigenmaps for dimensionality reduction and data representation*, Neural Computation, 15 (2003), pp. 1373–1396.

[12] P. BISWAS, H. AGHAJAN, AND Y. YE, *Semidefinite programming algorithms for sensor network localization using angle of arrival information*, in Proc. 39th Annu. Asilomar Conf. Signals, Systems, and Computers, Oct. 2005, pp. 220–224.

[13] P. BISWAS, T. C. LIAN, T. C. WANG, AND Y. YE, *Semidefinite programming based algorithms for sensor network localization*, ACM Transactions on Sensor Networks, 2 (2006), pp. 188–220.

[14] P. BISWAS, T. LIANG, K. TOH, Y. YE, AND T. WANG, *Semidefinite programming approaches for sensor network localization with noisy distance measurements*, IEEE Transactions on Automation Science and Engineering, 3 (2006), pp. 360–371.

[15] P. BISWAS, K. C. TOH, AND Y. YE, *A distributed SDP approach for large scale noisy anchor-free graph realization with applications to molecular conformation*, SIAM J. Scientific Computing, 30 (2008), pp. 1251–1277.

[16] P. BISWAS AND Y. YE, *A distributed method for solving semidefinite programs arising from ad hoc wireless sensor network localization*, Technical report, Dept. of Management Sci. and Eng., Stanford University, (2003).

[17] ——, *Semidefinite programming for ad hoc wireless sensor network localization*, in Proceedings of the Third International Symposium on Information Processing in Sensor Networks, New York, 2004, ACM, pp. 46–54.

[18] E. D. BOLKER AND B. ROTH, *When is a bipartite graph a rigid framework?*, Pacific Journal of Mathematics, 90, pp. 27–44.

[19] I. BORG AND P. J. F. GROENEN, *Modern multidimensional scaling: Theory and applications*, Springer, New York, NY, 2005.

[20] S. BOYD, L. E. GHAOUI, E. FERON, AND V. BALAKRISHNAN, *Linear Matrix Inequalities in System and Control Theory*, SIAM: Society for Industrial and Applied Mathematics, 1994.

[21] J. BRUCK, J. GAO, AND A. JIANG, *Localization and routing in sensor networks by local angle information*, ACM Transactions on Sensor Networks, 5 (2009).

[22] J.-F. CAI, E. CANDÈS, AND Z. SHEN, *A singular value thresholding algorithm for matrix completion*, SIAM J. on Optimization, 20 (2008), pp. 1956–1982.

[23] E. CANDÈS AND B. RECHT, *Exact matrix completion via convex optimization*, Found. of Comput. Math., 9 (2008), pp. 717–772.

[24] E. CANDÈS AND T. TAO, *Decoding by linear programming*, in IEEE Transactions on Information Theory, vol. 51, 2005, pp. 4203–4215.

[25] R. R. COIFMAN AND S. LAFON, *Diffusion maps*, Appl. Comput. Harmon. Anal., 21 (2006), pp. 5–30.

[26] R. CONNELLY, *On generic global rigidity*, Applied Geometry and Discrete Mathematics, 4 (1991), pp. 147–155.

[27] ——, *Generic global rigidity*, Discrete Comput. Geom, 33 (2005), pp. 549–563.

[28] R. CONNELLY AND W. J. WHITELEY, *Global rigidity: The effect of coning*, Discrete and Computational Geometry, (2009). ISSN 0179-5376 (Print) 1432-0444 (Online).

[29] G. CORNILESCU, J. L. MARQUARDT, M. OTTIGER, AND A. BAX, *Validation of protein structure from anisotropic carbonyl chemical shifts in a dilute liquid crystalline phase*, Journal of The American Chemical Society, 120 (1998), pp. 6836–6837.

[30] T. F. COX AND M. A. A. COX, *Multidimensional Scaling*, Monographs on Statistics and Applied Probability 88, Chapman & Hall/CRC, Boca Raton, FL, 2001.

[31] M. CUCURINGU, *Synchronization over Z_2 and community detection in bipartite networks*. in progress.

[32] M. CUCURINGU, Y. LIPMAN, AND A. SINGER, *Sensor network localization by eigenvector synchronization over the Euclidean group*, ACM Transactions on Sensor Networks, (2011). accepted for publication.

[33] T. DAVIS, *Multifrontral multithreaded rank-revealing sparse QR factorization*, in Combinatorial Scientific Computing, U. Naumann, O. Schenk, H. D. Simon, and S. Toledo, eds., no. 09061 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2009, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

[34] J. DE LEEUW, *Applications of convex analysis to multidimensional scaling*, in Recent Developments in Statistics, J. R. Barra, F. Brodeau, G. Romierand, and B. V. Cutsem, eds., Amsterdam, 1977, North Holland Publishing Company, pp. 133–146.

[35] D. DONOHO, *Compressed sensing*, in IEEE Transactions on Information Theory, vol. 52, 2006, pp. 1289–1306.

[36] K. FAN AND A. J. HOFFMAN, *Some metric inequalities in the space of matrices*, Proceedings of the American Mathematical Society, 6 (1955), pp. 111–116.

[37] M. FAZEL, *Matrix rank minimization with applications*, Ph.D. Thesis, (2002).

[38] M. FAZEL, H. HINDI, AND S. BOYD, *Rank minimization and applications in system theory*, in Proceedings of American Control Conference, Boston, Massachusetts, 2004, pp. 3273–3278.

[39] D. G. FEDOROV AND K. KITAURA, *Extending the power of quantum chemistry to large systems with the fragment molecular orbital method*, J. Phys. Chem. A, 111 (2007), pp. 6904–6914. doi:10.1021/jp0716740. PMID 17511437.

[40] J. FRANK, *Three-Dimensional Electron Microscopy of Macromolecular Assemblies: Visualization of Biological Molecules in Their Native State*, Oxford University Press, USA, 2nd ed., 2006.

[41] A. GIRIDHAR AND P. R. KUMAR, *Distributed clock synchronization over wireless networks: Algorithms and analysis*, in 45th IEEE Conference on Decision and Control, 2006, pp. 4915–4920.

[42] H. GLUCK, *Almost all simply connected closed surfaces are rigid*, Geometric Topology, Lecture Notes in Mathematics, 438 (1975), pp. 225–239.

[43] G. GOLUB AND C. V. LOAN, *Matrix computations*, 3rd ed., Johns Hopkins University Press, Baltimore, 1996.

[44] S. J. GORTLER, A. D. HEALY, AND D. P. THURSTON, *Characterizing generic global rigidity*, AMERICAN JOURNAL OF MATHEMATICS, 4, p. 897.

[45] S. J. GORTLER, A. D. HEALY, AND D. P. THURSTON, *Characterizing generic global rigidity*, American Journal of Mathematics, 132 (2010), pp. 897–939.

[46] C. GOTSMAN AND Y. KOREN, *Distributed graph layout for sensor networks*, in Proc. Intl. Symp. Graph Drawing, 2004, pp. 273–284.

[47] C. GOTSMAN AND S. TOLEDO, *On the computation of null spaces of sparse rectangular matrices*, SIAM Journal on Matrix Analysis and Applications, 30 (2008), pp. 445–463.

[48] M. GRANT AND S. BOYD, *CVX: MATLAB software for disciplined convex programming, version 1.21*.

[49] ——, *Graph implementations for nonsmooth convex programs*, in Recent Advances in Learning and Control, V. Blondel, S. Boyd, and H. Kimura, eds., Lecture Notes in Control and Information Sciences, Springer-Verlag Limited, 2008, pp. 95–110.

[50] A. GRISHAEV AND M. LLINAS, *CLOUDS, a protocol for deriving a molecular proton density via NMR*, Proc Natl Acad Sci USA, 99 (2002), pp. 6707–6712.

[51] P. GUNTERT, *Automated NMR protein structure determination*, Prog NMR Spectrosc, 43 (2003), pp. 105–125.

[52] ——, *Automated NMR structure calculation with CYANA*, Methods Mol Biol, 278 (2004), pp. 353–378.

[53] R. HADANI AND A. SINGER, *Representation theoretic patterns in three dimensional Cryo-Electron Microscopy I - the intrinsic reconstitution algorithm*, submitted.

[54] T. F. HAVEL AND K. WUTHRICH, *An evaluation of the combined use of nuclear magnetic resonance and distance geometry for the determination of protein conformation in solution*, J Mol Biol, 182 (1985), pp. 281–294.

[55] B. HENDRICKSON, *Conditions for unique graph realizations*, SIAM J Comput, 21 (1992), pp. 65–84.

[56] ——, *The molecule problem: Exploiting structure in global optimization*, SIAM J Optimization, 5 (1995), pp. 835–857.

[57] J. P. HESPANHA, *`grPartition` a MATLAB function for graph partitioning*.

[58] J. E. HOPCROFT AND R. E. TARJAN, *Dividing a graph into triconnected components*, SIAM J. Comput., 2 (1973), pp. 135–158.

[59] B. HORN, H. HILDEN, AND S. NEGAHDARIPOUR, *Closed-form solution of absolute orientation using orthonormal matrices*, Journal of the Optical Society of America A, 5 (1988), pp. 1127–1135.

[60] B. JACKSON AND T. JORDAN, *Connected rigidity matroids and unique realization graphs*, J. Combinatorial Theory B, 94 (2005), pp. 1–29.

[61] B. JACKSON, B. SERVATIUS, AND H. SERVATIUS, *The 2-dimensional rigidity of certain families of graphs*, Journal of Graph Theory, 54 (2006), pp. 154–166.

[62] D. JACOBS, *Linear fitting with missing data: Applications to structure-from-motion and to characterizing intensity images*, in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), 1997.

[63] D. J. JACOBS AND B. HENDRICKSON, *An algorithm for two-dimensional rigidity percolation: The pebble game*, Journal of Computational Physics, 137 (1997), pp. 346–365.

[64] A. JAVANMARD AND A. MONTANARI, *Localization from incomplete noisy distance measurements*, submitted, (2011).

[65] X. JI AND H. ZHA, *Sensor positioning in wireless ad-hoc sensor networks using multidimensional scaling*, in INFOCOM, vol. 4, 2004, pp. 2652–2661.

[66] R. KARP, J. ELSON, D. ESTRIN, AND S. SHENKER, *Optimal and global time synchronization in sensornets*, tech. report, Center for Embedded Networked Sensing, University of California, Los Angeles, 2003.

[67] J. B. KELLER, *Closest unitary, orthogonal and hermitian operators to a given operator*, Mathematics Magazine, 48 (1975), pp. 192–197.

[68] R. KESHAVAN, A. MONTANARI, AND S. OH, *Learning low rank matrices from $O(n)$ entries*, in Proc. of the Allerton Conf. on Commun., Control and Computing, 2008.

[69] R. KESHAVAN, S. OH, AND A. MONTANARI, *Matrix completion from a few entries*, in ISIT, 2009.

[70] U. A. KHAN, S. KAR, AND J. M. F. MOURA, *DILAND: An algorithm for distributed sensor localization with noisy distance measurements*, IEEE Transactions on Signal Processing, 58 (2010), pp. 1940–1947.

[71] Y. KOREN, C. GOTSMAN, AND M. BEN-CHEN, *PATCHWORK: Efficient localization for sensor networks by distributed global optimization*, tech. report, 2005.

[72] G. LAMAN, *On graphs and rigidity of plane skeletal structures*, Journal of Engineering Mathematics, 4 (1970), pp. 331–340.

[73] A. LEE AND I. STREINU, *Pebble game algorithms and sparse graphs*, Discrete Mathematics, 308 (2008), pp. 1425–1437.

[74] N. H. Z. LEUNG AND K. C. TOH, *An SDP-based divide-and-conquer algorithm for large scale noisy anchor-free graph realization*, Technical report, Dept. of Management Sci. and Eng., Stanford University, 31 (2009), pp. 4351–4372.

[75] J. MAXWELL, *On the calculation of the equilibrium and stiffness of frames*, Philos. Mag., 27 (1864), pp. 294–299.

[76] M. MESBAHI AND G. PAPAVASSILOPOULOS, *On the rank minimization problem over a positive semidefinite linear matrix inequality*, in IEEE Transactions on Automatic Control, vol. 42, 1997, pp. 239–243.

[77] D. MOORE, J. LEONARD, D. RUS, AND S. TELLER, *Robust distributed network localization with noisy range measurements*, in Proceedings of the Second ACM Conference on Embedded Networked Sensor Systems, November 3-5 2004, pp. 50–61.

[78] NETFLIX, *ACM SIGKDD and netflix*. Proceedings of KDD Cup and Workshop, Apr. 2007.

[79] C. PAIGE AND M. SAUNDERS, *LSQR: An algorithm for sparse linear equations and sparse least squares*, ACM Transactions on Mathematical Software (TOMS), 8 (1982), pp. 43–71.

[80] A. G. PALMER AND F. MASSI, *Characterization of the dynamics of biomacromolecules using rotating-frame spin relaxation NMR spectroscopy*, Chem Rev, 106 (2006), pp. 1700–1719.

[81] D. PEAUCELLE, D. HENRION, Y. LABIT, AND K. TAITZ, *User's guide for SeDuMi interface 1.04*.

[82] W. RIEPING, M. HABECK, B. BARDIAUX, A. BERNARD, T. MALLIAVIN, AND M. NILGES, *ARIA2: automated NOE assignment and data integration in NMR structure calculation*, Bioinformatics, 23 (2007), pp. 381–382.

[83] B. ROTH, *Rigid and flexible frameworks*, The American Mathematical Monthly, 88 (1981), pp. 6–21.

[84] S. T. ROWEIS AND L. K. SAUL, *Nonlinear dimensionality reduction by locally linear embedding*, Science, 290 (2000), pp. 2323–2326.

[85] A. SALI, R. GLAESER, T. EARNEST, AND W. BAUMEISTER, *From words to literature in structural proteomics*, Nature, 422 (2003), pp. 216–225.

[86] J. B. SAXE, *Embeddability of weighted graphs in k-space is strongly NP-hard*, in Proc. 17th Allerton Conf. Commun. Control Comput., 1979, pp. 480–489.

[87] S. SCHAEFER, T. MCPHAIL, AND J. WARREN, *Image deformation using moving least squares*, in ACM TOG, Press, 2006, pp. 533–540.

[88] C. D. SCHWIETERS, J. J. KUSZEWSKI, N. TJANDRA, AND G. M. CLORE, *The Xplor-NIH NMR molecular structure determination package*, J Magn Reson, 160 (2003), pp. 65–73.

[89] Y. SHANG AND W. RUML, *Improved MDS-based localization*, in Proceedings of IEEE Infocom, vol. 23, Hong Kong, China, 2004, pp. 2640–2651.

[90] J. SHI AND J. MALIK, *Normalized cuts and image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 22 (2000), pp. 888–905.

[91] A. SINGER, *A remark on global positioning from local distances*, Proceedings of the National Academy of Sciences, 105 (2008), pp. 9507–9511.

[92] ——, *Angular synchronization by eigenvectors and semidefinite programming*, Appl. Comput. Harmon. Anal., 30 (2011), pp. 20–36.

[93] A. SINGER AND Y. SHKOLNISKY, *Three-dimensional structure determination from common lines in Cryo-EM by eigenvectors and semidefinite programming*, SIAM Journal on Imaging Sciences, 4 (2011), pp. 543–572.

[94] A. SINGER AND H. T. WU, *Vector Diffusion Maps and the Connection Laplacian*, Communications on Pure and Applied Mathematics (CPAM).

[95] A. M. C. SO, *A semidefinite programming approach to the graph realization problem: Theory, applications and extensions*, Ph.D. Thesis, (2007).

[96] A. M. C. SO AND Y. YE, *Theory of semidefinite programming for sensor network localization*, in Proceedings of the 17th Annual ACM–SIAM Symposium on Discrete Algorithm (SODA), 2005, pp. 405–414.

[97] N. SREBRO, *Learning with matrix factorizations*, PhD thesis, (2004).

[98] N. SREBRO AND T. JAAKKOLA, *Weighted low-rank approximations*, in Proceedings of the Twentieth International Conference on Machine Learning (ICML), 2003, pp. 720–727.

[99] J. SUN, S. BOYD, L. XIAO, AND P. DIACONIS, *The fastest mixing markov process on a graph and a connection to a maximum variance unfolding problem*, SIAM Review, 48 (2006), pp. 681–699.

[100] L. THERAN, *Rigid components of random graphs*, in Proc. of the 21st Canadian Conference on Computational Geometry, Vancouver, BC, 2009.

[101] K. TOH, P. BISWAS, AND Y. YE, *SNLSDP version 0 - a MATLAB software for sensor network localization*.

[102] C. TOMASI AND T. KANADE, *Shape and motion from image streams under orthography: a factorization method*, International Journal of Computer Vision, 9 (1992), pp. 137–154.

[103] L. N. TREFETHEN AND D. BAU, *Numerical Linear Algebra*, SIAM: Society for Industrial and Applied Mathematics, 1997.

[104] M. TUBAISHAT AND S. MADRIA, *Sensor networks: An overview*, in IEEE Potentials, vol. 22, 2003, pp. 20–23.

[105] L. VANDENBERGHE AND S. BOYD, *Semidefinite programming*, SIAM Review, 38 (1996), pp. 49–95.

[106] U. VON LUXBURG, *A tutorial on spectral clustering*, Statistics and Computing, 17 (2007), pp. 395–416.

[107] S. J. D. VRIES, M. VAN DIJK, AND A. BONVIN, *The HADDOCK web server for data-driven biomolecular docking*, Nature Protocols 5, (2010), pp. 883–897.

[108] D. WAGNER AND F. WAGNER, *Between min cut and graph bisection*, in Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science, MFCS '93, 1993, pp. 744–750.

[109] K. Q. WEINBERGER, F. SHA, Q. ZHU, AND L. K. SAUL, *Graph laplacian regularization for large-scale semidefinite programming*, in Advances in Neural Information Processing Systems (NIPS), B. Schoolkopf, J. Platt, and T. Hofmann, eds., Cambridge, MA, 2007, MIT Press.

[110] K. WUTHRICH, *NMR studies of structure and function of biological macromolecules (Nobel lecture)*, J Biomol NMR, 27 (2003), pp. 13–39.

[111] J. X. YAO, E. J. DODSON, K. S. WILSON, AND M. M. WOOLFSON, *ACORN: a review*, Acta Crystallographica Section D, 62 (2006).

[112] Y. YEMINI, *Some theoretical aspects of location-location problems*, in Proc. IEEE Sympos. Found. Comput. Sci., 1979, pp. 1–8.

[113] L. ZHANG, L. LIU, C. GOTSMAN, AND S. J. GORTLER, *An As-Rigid-As-Possible approach to sensor network localization*, ACM Transactions on Sensor Networks, 6 (2011).

[114] Z. ZHU, A. M. C. SO, AND Y. YE, *Universal rigidity: Towards accurate and efficient localization of wireless networks*, in Proc. IEEE INFOCOM, 2010.