UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Virtual Object Reconstruction from Paired-Point Distances

Emile Visser
21595240

Report submitted in partial fulfilment of the requirements of the module Project (E) 448 for the degree Baccalaureus in Engineering in the Department of Electrical and Electronic Engineering at Stellenbosch University.

Supervisor: Dr J. du Preez

November 2021

Acknowledgements

I would like to thank my dog, Muffin. I also would like to thank the inventor of the incubator; without him/her, I would not be here. Finally, I would like to thank Dr Herman Kamper for this amazing report template.

UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Plagiaatverklaring / *Plagiarism Declaration*

1. Plagiaat is die oorneem en gebruik van die idees, materiaal en ander intellektuele eiendom van ander persone asof dit jou eie werk is.
   Plagiarism is the use of ideas, material and other intellectual property of another's work and to present is as my own.

2. Ek erken dat die pleeg van plagiaat 'n strafbare oortreding is aangesien dit 'n vorm van diefstal is.
   I agree that plagiarism is a punishable offence because it constitutes theft.

3. Ek verstaan ook dat direkte vertalings plagiaat is.
   I also understand that direct translations are plagiarism.

4. Dienooreenkomstig is alle aanhalings en bydraes vanuit enige bron (ingesluit die internet) volledig verwys (erken). Ek erken dat die woordelikse aanhaal van teks sonder aanhalingstekens (selfs al word die bron volledig erken) plagiaat is.
   Accordingly all quotations and contributions from any source whatsoever (including the internet) have been cited fully. I understand that the reproduction of text without quotation marks (even when the source is cited) is plagiarism

5. Ek verklaar dat die werk in hierdie skryfstuk vervat, behalwe waar anders aangedui, my eie oorspronklike werk is en dat ek dit nie vantevore in die geheel of gedeeltelik ingehandig het vir bepunting in hierdie module/werkstuk of 'n ander module/werkstuk nie.
   I declare that the work contained in this assignment, except where otherwise stated, is my original work and that I have not previously (in its entirety or in part) submitted it for grading in this module/assignment or another module/assignment.

Studentenommer / *Student number*	Handtekening / *Signature*
Voorletters en van / *Initials and surname*	Datum / *Date*

Abstract

With the advent of autonomous agents such as vehicles, drones, etc., the need has arisen for a way to translate objects from the physical to virtual space. Modern computer vision algorithms can identify points on an object and calculate the distances between them, but have no way to represent these distances as an internal, virtual object. The aim of this research paper is to develop two algorithms that can solve the reconstruction problem from paired-point distances, even in the presence of noise and record errors. The first of these two algorithms is a deterministic algorithm based on the theory of linear algebra and, in particular, the Euclidean Distance Matrix (EDM), Multidimensional Scaling (MDS) and Semidefinite Programming (SDP). In contrast, the second algorithm is developed probabilistically, using the theory of Probabilistic Graphical Models (PGM) and the Unscented Transform (UT). This report contains both the respective theoretical backgrounds and the implementations used for the testing of the two algorithms.

ADD RESULTS

ADD CONCLUSION

Contents

List of Figures

List of Tables

Nomenclature

Variables and functions

n	Number of points in original set.
k	Dimension of points in original set.
$\mathbf{x}_i$	Original point with index i.
$\mathbf{X}$	Original set of points.
$\hat{\mathbf{X}}$	Reconstructed set of points.
$\mathbf{D}$	Fully defined Euclidean Distance Matrix of $\mathbf{X}$.
$\tilde{\mathbf{D}}$	Partially defined Euclidean Distance Matrix of $\mathbf{X}$.
$\mathbf{W}$	Mask matrix of partially defined Euclidean Distance Matix $\tilde{\mathbf{D}}$.
$\mathbf{E}$	Set of measurements in partially defined Euclidean Distance Matix $\tilde{\mathbf{D}}$.
$\hat{\mathbf{E}}$	Set of measurements from reconstructed set $\hat{\mathbf{X}}$ that correspond to the original set $\mathbf{E}$.
$\hat{\mathbf{D}}$	Reconstructed full Euclidean Distance Matrix of $\tilde{\mathbf{D}}$.
$\mathbf{G}$	Gram matrix.
$\mathbf{C}$	Geometric centering matrix.
$\mathcal{G}(\mathbf{D})$	Function returning Gram matrix of Euclidean Distance Matrix $\mathbf{D}$.
$\mathcal{K}(\mathbf{G})$	Function returning Euclidean Distance Matrix of Gram matrix $\mathbf{G}$.
$\mathbf{I}_n$	Identity matrix of size $n \times n$.
$\mathbf{1}_n$	Vector of ones of size $n \times 1$.
$\left\| \mathbf{x} \right\|_2$	L^2 norm of vector $\mathbf{x}$.
$\left\| \mathbf{X} \right\|_F$	Frobenius norm of matrix $\mathbf{X}$.
$\mathbf{X} \cdot \mathbf{Y}$	Dot product of matrices $\mathbf{X}$ and $\mathbf{Y}$.
$\mathbf{X} \times \mathbf{Y}$	Cross product of matrices $\mathbf{X}$ and $\mathbf{Y}$.
$\mathbf{X} \circ \mathbf{Y}$	Hadamard (element-wise) product of matrices $\mathbf{X}$ and $\mathbf{Y}$.
$\mathbf{X}^\top$	Transpose of matrix $\mathbf{X}$.
$\mathbf{X}^{-1}$	Inverse of matrix $\mathbf{X}$.
$\mathbf{X}^+$	Moore-Penrose inverse of matrix $\mathbf{X}$.

$\det(\mathbf{X})$	Determinant of matrix $\mathbf{X}$.
$\mathrm{tr}(\mathbf{X})$	Trace of matrix $\mathbf{X}$.
$\mathrm{rank}(\mathbf{X})$	Rank of matrix $\mathbf{X}$.
$\mathrm{EVD}(\mathbf{X})$	Eigenvalue Decomposition of matrix $\mathbf{X}$.
$\mathbb{R}^{x\times y}$	Space of real matrices of size $x \times y$.
$\mathbb{R}^n_+$	Space of positive semidefinite matrices of size $n \times n$.
$\mathbb{S}^n_{\mathrm{c}}$	Space of symmetric, geometrically centered matrices of size $n \times n$.
$\mathbb{EDM}^n$	Space of Euclidean Distance Matrices of size $n \times n$.
$O(n)$	Bachmann–Landau notation that describes the order of function n.
$\mathrm{card}(\mathcal{A})$	Cardinality/Size of set $\mathcal{A}$.
a_j	Element j of sequence $(a_i)_{i=1}^{\mathrm{card}(\mathcal{A})}$ of set $\mathcal{A}$.
$\mathrm{sort}_{\mathrm{asc}}(\mathcal{A} \mid \mathcal{B})$	Sort set $\mathbf{A}$ based on values of set $\mathbf{B}$ ($\mathrm{card}(\mathcal{A}) = \mathrm{card}(\mathcal{B})$).
Pa^G_X	Parents of random variable X in graph G.
$\boldsymbol{\mu}$	Statistical mean vector.
$\boldsymbol{\Sigma}$	Statistical covariance matrix.
$\mathbf{X} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	Multivariate Gaussian PDF of random variable $\mathbf{X}$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$.

$$\mathrm{sgn}(x) \overset{def}{=} \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases}$$

Acronyms and abbreviations

CPD	Conditional Probability Distribution
DAG	Directed Acyclic Graph
EDM	Euclidean Distance Matrix
EKF	Extended Kalman Filter
MAP	Maximum a Posteriori
MDS	Multidimensional Scaling
MRF	Markov Random Field
PCoA	Principal Coordinates Analysis
PGM	Probabilistic Graphical Model
RIP	Running Intersection Property
RMSE	Root Mean Squared Error
SDP	Semidefinite Programming
SVD	Singular Value Decomposition
UKF	Unscented Kalman Filter
UT	Unscented Transform

Chapter 1

Introduction

With the advent of autonomous agents such as vehicles and drones, the need has arisen for a way to translate observed objects from the physical to virtual space. Using these virtual objects, the autonomous agent can create an internal approximation of its surroundings that can used for pathfinding, anomaly detection, etc.

Modern, state of the art computer vision algorithms use feature extraction algorithms to identify points on an arbitrary object. Alongside cameras in stereo, the approximate distances between these points can be calculated using homography. Using these distance measurements between pairs of identified features and little to no prior knowledge of the shape of the object, a virtual representation of the object must be constructed.

While this problem is relatively trivial in the case with complete, exact and correct measurements, in practice these measurements are subject to the performance of the feature extraction and homography algorithms, which may introduce noise, identify incorrect feature pairs or return an incomplete dataset.

This report seeks to address this problem by creating algorithms that are resilient against these errors while reconstructing the 3-D object from the paired distance measurements.

1.1. Objectives

The objectives for this report are given below:

- Construct an algorithm based on standard deterministic methods to solve the reconstruction from paired-point distances problem. This algorithm will provide a performance baseline to compare the next algorithm against.
- Develop an algorithm based on probabilistic methods to solve the reconstruction from paired-point distances problem.
- Develop a method to determine anomalous measurements in the paired-point distances set.
- Test the two algorithms with varying parameters (network interconnectivity, noise, distance error, etc.).

- Draw a conclusion on the relative performance of these two algorithms.

1.2. Outcomes

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

1.3. Overview

In summary, this report aims to solve the reconstruction from paired-point distances problem using algorithms that are based on deterministic- and probabilistic methods respectively.

The deterministic algorithm is developed, using the standard approach to this problem to serve as a performance comparison baseline, from the theory of Euclidean distance matrices (Sec. 2.1) to represent the observed distances, semidefinite programming (Sec. 2.3) to calculate the missing entries in this matrix and multidimensional scaling (Sec. 2.1) to map the reconstructed points from the completed matrix.

As an alternative solution to the baseline algorithm, the probabilistic algorithm is based on the theory of probabilistic graphical models (Sec. 3.1). In this model, belief regarding the positions of the reconstructed points is represented as probability distributions with these beliefs propagated across the model and readjusted using the unscented transform (3.2) to conform to the observed distances.

In order to improve reconstruction performance, distance measurements that have been corrupted are identified and removed using a heuristic (Sec. 4.1) based on the relative change between the original measurements on the distances between the reconstructed points.

The implementation of these algorithms are given in Sec. 5.2 though 5.4.

The testing framework for this report consists of a reference set of points on the surface of a virtual cube. From this set of points a dataset of measurements can be generated according to a set of parameters such as the number of connections per point, noise and percentage of records corrupted (Sec. 5.1). Quantification of the reconstruction error is done by calculating the relative change between the Euclidean distance matrix of the reference- and reconstructed set of points (Sec. 5.6).

The performance of the algorithms when subjected to varying dataset parameters is given in Ch. 6 and from these results, conclusions are drawn regarding the relative performance of the algorithms.

Chapter 2

Deterministic Reconstruction Methods

> *Mentat computation remained finite. You couldn't say something boundless within the boundaries of any language. Mentat abilities had their uses, though.*
>
> — Paul Atreides, *Dune Messiah* [1]

This chapter is focused on solving the reconstruction problem deterministically (as opposed to the probabilistic methods of Ch. 3).

This chapter stands on the foundation of the Euclidean distance matrix (EDM) of Sec. 2.1, a matrix that describes the squared distance between every possible pair of points in the system. Using this matrix, multidimensional scaling (MDS) of Sec. 2.2 can be applied to reconstruct the least squares error map onto an arbitrary Cartesian space, in the physical case three-dimensional, up to translation, rotation and reflection.

MDS, however, requires a fully defined EDM, i.e. must contain distance information for every point to every other point. Therefore semidefinite programming (SDP), detailed in Sec. 2.3 is used to reconstruct the full EDM from a partially defined EDM.

Using a subset of known points, the MDS reconstruction can be rotated/reflected to align with the known points using the Householder matrix and Kabsch Algorithm (given in Sec. 2.4).

A flowchart describing this method is given in Fig. 2.1 below:

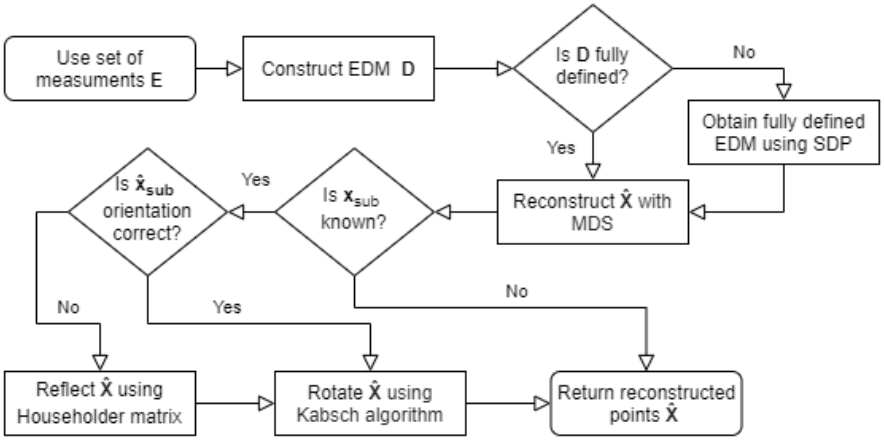


Figure 2.1: Deterministic Reconstruction Method Flowchart

2.1. Euclidean Distance Matrix

From Ch. 5 of *Convex Optimization & Euclidean Distance Geometry* [2], a Euclidean distance matrix (EDM) is defined as a $\mathbb{R}^n_+$ matrix on the set of n, k dimensional points $\mathbf{X} = \{\mathbf{x}_i, i = 1...n\}$ in $\mathbb{R}^{1\times k}$ where element d_{ij} in the matrix is defined as the square of the L^2 norm (which is strictly smooth and convex as opposed to the L^1 norm which is non-smooth and convex [3]) of the difference between points x_i and x_j with noise ϵ:

$$d_{ij} = \left\|x_i - x_j\right\|_2^2 + \epsilon \quad \text{where} \quad i, j \in \{1...n\} \tag{2.1}$$

For example, in the case where $n = 3$:

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & d_{13} \\ d_{21} & d_{22} & d_{23} \\ d_{31} & d_{32} & d_{33} \end{bmatrix} = \begin{bmatrix} 0 & d_{12} & d_{13} \\ d_{21} & 0 & d_{23} \\ d_{31} & d_{32} & 0 \end{bmatrix} \quad \text{since} \quad d_{ij} = 0 \quad \text{where} \quad i = j \tag{2.2}$$

A useful property of the EDM is that the symmetric Gram matrix $\mathbf{G}$ obtained by applying the geometric centering matrix $\mathbf{C}$, is positive semidefinite [4]. This property is defined below and is used in Sec. 2.3.

$$\mathbf{C} \overset{def}{=} \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top \tag{2.3}$$

$$\mathbf{G} = -\frac{1}{2}\mathbf{CDC} \succeq 0 \tag{2.4}$$

The reverse of this property is also true, being that if $-\frac{1}{2}\mathbf{CDC} \succeq 0$, then $\mathbf{D}$ is an EDM.

The EDM $\mathbf{D}$ is unique up to rigid transformations: translation, reflection or rotation of the original set $\mathbf{X}$. However, this means that this orientation information is lost when constructing the EDM and cannot be recovered during MDS in Sec. 2.2.

Information regarding k (dimension of the original points in $\mathbf{X}$) is also lost when taking the L^2 norm, however this should not be a problem since in most practical applications this will be known beforehand or can be assumed to be 2- or 3-dimensional, since it will be used to reconstruct physical objects that are at most 3-dimensional.

2.2. Multidimensional Scaling

Multidimensional scaling (MDS), first proposed in 1952 by W.C. Torgerson [5] and also known as principal coordinates analysis (PCoA), is an algorithm that maps a set of points to an x-dimensional space so that their between-point distances are preserved as much as

possible with a given EDM. Stated simply, MDS translates an EDM back into a set of points of arbitrary dimension.

This algorithm, as outlined in *An Introduction to MDS* [6], is given in Al. 2.1.

Algorithm 2.1: Classical MDS

Require: EDM $\mathbf{D}$ *of size* $n \times n$
Require: k *as required dimensions of map*
$\mathbf{C} \leftarrow \mathbf{I}_n - \frac{1}{n}\mathbf{1}_n\mathbf{1}_n^\top$ ▷ Get centering matrix
$\mathbf{G} \leftarrow -\frac{1}{2}\mathbf{CDC}$ ▷ Get Gram matrix
$\mathbf{U}, \{\lambda_{1\ldots n}\} \leftarrow \text{EVD}(\mathbf{G})$ ▷ Eigenvalue decomposition of Gram matrix
$\mathbf{\Lambda}_k \leftarrow \text{diag}(\lambda_1, \lambda_2 ... \lambda_k)$ ▷ Get diagonal matrix with k largest eigenvalues
$\mathbf{U}_k \leftarrow (U_1 \,|\, U_2 \,|\, ... |\, U_k)$ ▷ Concatenate k largest eigenvectors
$\hat{\mathbf{X}} \leftarrow \mathbf{U}_k\sqrt{\mathbf{\Lambda}_k}$ ▷ Get projected points
return $\hat{\mathbf{X}}$

The matrix $\hat{X}$ returned by this algorithm is the matrix that minimizes the loss function known as 'Stress', given below [7]:

$$\text{Stress}_{\mathbf{D}}(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, ..., \hat{\mathbf{x}}_n) = \sqrt{\sum_{i \neq j = 1 \ldots n} (\mathbf{D}_{ij} - \left\|\hat{\mathbf{x}}_i - \hat{\mathbf{x}}_j\right\|_2^2)^2} \tag{2.5}$$

This loss function can be interpreted as Root Mean Squared Error (RMSE) of the differences between the original and reconstructed EDM. Since the algorithm discards all but the largest eigenvalues, it is concluded that MDS is relatively robust to Gaussian noise in the EDM. This property is also attested in a paper by Peterfreund and Gavish [8].

While MDS is very efficient when working with full EDM's (distance between every point and every other point is known), in practice this is an almost impossible requirement.

In the first case, the number of distances that must be known (read: measured) can be stated as a k-combinations problem with n as the number of points in the system.

$$\binom{n}{2} = \frac{n!}{2!(n-2)!} = \frac{1}{2}(n)(n-1) = \frac{1}{2}n^2 - \frac{1}{2}n \quad \therefore O(n^2) \tag{2.6}$$

Stated otherwise, the number of measurements needed to fully define the EDM of a system of points increases with quadratic order, making it intractable for large systems.

Secondly, it is not always possible to measure the distance between every pair of points in the system, e.g. if one point is visible from one perspective but not another, leading to gaps in the EDM. The EDM has no mechanism to flag which of these measurements are missing (crucially, a value of 0 would imply that the points have no distance between them). Naively discarding every point that is not fully connected to every other also discards valid measurements that contain valuable information that could be used during reconstruction and may reduce the MDS's ability to fully capture the shape being reconstructed.

In the next section, semidefinite programming (SDP) will be used to transform a semi-defined EDM into a fully defined EDM that can be used by the MDS algorithm.

2.3. Semidefinite Programming

Seemingly a misnomer, semidefinite programming (SDP), is a type of optimisation that minimizes a user defined loss function over the intersection of positive semidefinite matrices and an affine space. In other words, SDP finds a positive semidefinite matrix that both minimizes error as defined by the user and is an element of a given space.

In this section, SDP will be used to fill a fully defined EDM $\hat{\mathbf{D}}$ from a semi-defined EDM $\tilde{\mathbf{D}}$. As mentioned previously in Sec. 2.2, An EDM has no intrinsic way of flagging missing distance measurements, since a value of zero would imply that the two points defined by that entry must lie at the same position. This problem is addressed by defining a mask matrix $\mathbf{W}$ from an index set of valid measurements $\mathcal{E}$ (where a value of 1 at w_{ij} would indicate that a measurement exists between $\mathbf{x}_i$ and $\mathbf{x}_j$ in the original set $\mathbf{X}$):

$$w_{ij} \overset{def}{=} \begin{cases} 1 & (i,j) \in \mathcal{E} \\ 0 & otherwise \end{cases} \tag{2.7}$$

Recall from Eq. 2.4 that for an EDM $\mathbf{D}$, its Gram matrix $\mathbf{G}$ is positive semidefinite while also being geometrically centered. Therefore, there is a direct map from the space of possible EDM's ($\mathbb{EDM}^n$) to the intersection of possible positive semidefinite matrices $\mathbb{R}^n_+$ and symmetric, geometrically centered matrices $\mathbb{S}^n_c$. Using this correspondence, the problem can be used to cast EDM completion as a relaxed semidefinite program [9]:

$$\underset{\mathbf{G}}{\text{minimize}} \quad \left\| \mathbf{W} \circ (\tilde{\mathbf{D}} - \mathcal{K}(\mathbf{G})) \right\|_F^2 \tag{2.8}$$

$$\text{subject to} \quad \mathbf{G} \in \mathbb{R}^n_+ \cap \mathbb{S}^n_c \tag{2.9}$$

Also from [9], the constraint $\mathbf{G} \in \mathbb{S}^n_c$ implies that $\mathbf{G}$ has a nullspace (at least one of its eigenvalues must be 0) that may cause numerical problems. To solve this, some sort of invertible transformation must be applied to $\mathbf{G}$ to transform it to a lower dimensional space, removing the part responsible for the nullspace. Dokmanić *et al.* proposes the following transformation:

$$\mathcal{G}_{n-1}(\mathbf{D}) \overset{def}{=} -\frac{1}{2}\mathbf{V}^\top \mathbf{D} \mathbf{V} \quad \text{where} \quad \mathcal{G}_{n-1}(\mathbf{D}) \in R^{n-1}_+ \tag{2.10}$$

$$\mathbf{V} = \begin{bmatrix} p & p & ... & p \\ 1+q & q & ... & q \\ q & 1+q & ... & q \\ \vdots & \vdots & \ddots & \vdots \\ q & q & ... & 1+q \end{bmatrix} \quad \text{such that} \quad \mathbf{V} \in \mathbb{R}_+^{n\times(n-1)} \quad \text{and} \quad \mathbf{V}^\top\mathbf{V} = 1 \tag{2.11}$$

$$\therefore p = -\frac{1}{n+\sqrt{n}} \quad \text{and} \quad q = -\frac{1}{\sqrt{n}} \tag{2.12}$$

Additionally:

$$\mathcal{K}(\mathbf{V}\mathcal{G}_{n-1}(\mathbf{D})\mathbf{V}^\top) = \mathbf{D} \tag{2.13}$$

Therefore, $\mathbf{H} \mapsto \mathcal{K}(\mathbf{V}\mathbf{H}\mathbf{V}^\top)$ is an invertible map from $\mathbb{R}_+^{n-1}$ to $\mathbb{EDM}^n$. Using this observation, a more numerically stable SDP can be constructed:

$$\underset{\mathbf{H}}{\text{minimize}} \quad \left\|\mathbf{W} \circ (\tilde{\mathbf{D}} - \mathcal{K}(\mathbf{V}\mathbf{H}\mathbf{V}^\top))\right\|_F^2 \tag{2.14}$$

$$\text{subject to} \quad \mathbf{H} \in \mathbb{R}_+^{n-1} \tag{2.15}$$

However, as mentioned by Krislock & Wolkowicz [10], this solution allows movement of the points in higher dimensions even if there is a valid solution in k-dimensions. A heuristic for promoting lower dimensions can be found by maximising the trace of the Gram matrix, as proposed by Biswas, *et al.* [11]. This stretches out the points as much as possible, favouring smaller, affine dimensions (as if smoothing out a crumpled piece of paper). Finally, noting that $\text{tr}(\mathbf{G}) = \text{tr}(\mathbf{H})$, the SDP can be rewritten [9], including the data fidelity term proposed by Biswas, *et al.*[1]:

$$\underset{\mathbf{H}}{\text{maximise}} \quad \text{tr}(\mathbf{H}) - \lambda\left\|\mathbf{W} \circ (\tilde{\mathbf{D}} - \mathcal{K}(\mathbf{V}\mathbf{H}\mathbf{V}^\top))\right\|_F^2 \tag{2.16}$$

$$\text{subject to} \quad \mathbf{H} \in \mathbb{R}_+^{n-1} \tag{2.17}$$

A Matlab/CVX [12] [13] script of this SDP can be found in App. C. Once this matrix $\mathbf{H}$ is determined, the approximate EDM $\hat{\mathbf{D}}$ can be determined:

$$\hat{\mathbf{D}} = \mathcal{K}(\mathbf{V}\mathbf{H}\mathbf{V}^\top) \tag{2.18}$$

This EDM $\hat{\mathbf{D}}$ is guaranteed to be fully defined and can then be used by the MDS of Sec. 2.2 to create a reconstruction $\hat{\mathbf{X}}$.

[1] This value for λ is determined to be 10 experimentally in Sec. 5.2

2.4. Rotation Matrix Estimation

As mentioned in Sec. 2.1, the EDM loses information regarding the orientation and dimension of $\mathbf{X}$. While the dimension k of the reconstructed points $\hat{\mathbf{X}}$ can usually be inferred by the implementation (e.g. 3-dimensional when working with stereo cameras), the lost information allows a reconstruction that is accurate only up to translation, rotation and reflection. If the points $\mathbf{X}$ or a subset thereof are known, a transformation can be applied to $\hat{\mathbf{X}}$ to align it with $\mathbf{X}$. The next sections will illustrate this transformation.

2.4.1. Naive Inverse Matrix

If the true values are known, the naive solution would be to construct a transformation matrix that could be applied to $\hat{\mathbf{X}}$ to rotate it to the same orientation as $\mathbf{X}$. Firstly, taking the first points, $\mathbf{x}_1$ and $\hat{\mathbf{x}}_1$, arbitrarily as the origin of the new coordinate system, $\mathbf{X}$ and $\hat{\mathbf{X}}$ are translated to align with this assumption by broadcasting this subtraction to all of the points in the set:

$$\mathbf{X}_\mathrm{t} = \mathbf{X} - \mathbf{1}_n\mathbf{x}_1 \quad \text{where} \quad \mathbf{X} \in \mathbb{R}^{n\times k} \quad \text{and} \quad \mathbf{x}_1 \in \mathbb{R}^{1\times k} \tag{2.19}$$

$$\hat{\mathbf{X}}_\mathrm{t} = \hat{\mathbf{X}} - \mathbf{1}_n\hat{\mathbf{x}}_1 \quad \text{where} \quad \hat{\mathbf{X}} \in \mathbb{R}^{n\times k} \quad \text{and} \quad \hat{\mathbf{x}}_1 \in \mathbb{R}^{1\times k} \tag{2.20}$$

The transformation matrix $\mathbf{T}$ is calculated using the inverse of $\mathbf{X}_\mathrm{t}$ below:

$$\hat{\mathbf{X}}_\mathrm{t}\mathbf{T} = \mathbf{X}_\mathrm{t} \tag{2.21}$$

$$\hat{\mathbf{X}}_\mathrm{t}^{-1}\hat{\mathbf{X}}_\mathrm{t}\mathbf{T} = \hat{\mathbf{X}}_\mathrm{t}^{-1}\mathbf{X}_\mathrm{t} \tag{2.22}$$

$$\mathbf{T} = \hat{\mathbf{X}}_\mathrm{t}^{-1}\mathbf{X}_\mathrm{t} \tag{2.23}$$

With the transformation matrix $\mathbf{T}$, it is applied to $\hat{\mathbf{X}}_\mathrm{t}$ to obtain the rotated set of points:

$$\hat{\mathbf{X}}_\mathrm{t}\mathbf{T} = \hat{\mathbf{X}}_\mathrm{rot} \tag{2.24}$$

However, since $\hat{\mathbf{X}}_\mathrm{t}$ is non-square ($\hat{\mathbf{X}}_\mathrm{t} \in \mathbb{R}^{n\times d}$) it is not invertible and singular. This problem is addressed in the next section.

2.4.2. Moore-Penrose Inverse Matrix

While $\hat{\mathbf{X}}_\mathrm{t}$ does not have an inverse, the Moore-Penrose inverse $\hat{\mathbf{X}}_\mathrm{t}^+$ can be calculated that corresponds to a least squares solution of the system of linear equations represented by $\hat{\mathbf{X}}_\mathrm{t}$. This matrix is a left inverse ($\hat{\mathbf{X}}_\mathrm{t}^+\hat{\mathbf{X}}_\mathrm{t} = 1$) when rank($\hat{\mathbf{X}}$) >= k (columns are linearly independent) allowing its use in same fashion as above.

$$\mathbf{X}_\mathrm{t} = \mathbf{X} - \mathbf{1}_n\mathbf{x}_1 \quad \text{where} \quad \mathbf{X} \in \mathbb{R}^{n\times k} \quad \text{and} \quad \mathbf{x}_1 \in \mathbb{R}^{1\times k} \tag{2.25}$$

$$\hat{\mathbf{X}}_\mathrm{t} = \hat{\mathbf{X}} - \mathbf{1}_n\hat{\mathbf{x}}_1 \quad \text{where} \quad \hat{\mathbf{X}} \in \mathbb{R}^{n\times k} \quad \text{and} \quad \hat{\mathbf{x}}_1 \in \mathbb{R}^{1\times k} \tag{2.26}$$

$$\hat{\mathbf{X}}_\mathrm{t}\mathbf{T} = \mathbf{X}_\mathrm{t} \tag{2.27}$$

$$\hat{\mathbf{X}}_\mathrm{t}^+\hat{\mathbf{X}}_\mathrm{t}\mathbf{T} = \hat{\mathbf{X}}_\mathrm{t}^+\mathbf{X}_\mathrm{t} \tag{2.28}$$

$$\mathbf{T} = \hat{\mathbf{X}}_\mathrm{t}^+\mathbf{X}_\mathrm{t} \tag{2.29}$$

As before this transformation matrix is applied to $\hat{\mathbf{X}}_\mathrm{t}$ to rotate the set of points:

$$\hat{\mathbf{X}}_\mathrm{t}\mathbf{T} = \hat{\mathbf{X}}_\mathrm{rot} \tag{2.30}$$

While certainly powerful and simple, this transformation is critically not distance preserving since det($\mathbf{T}$) is not guaranteed to equal 1, leading to possible distortions when applied to the set of reconstructed points.

2.4.3. Kabsch Algorithm and Householder Matrix

A solution to the distortion problem of the Moore-Penrose Inverse would be to use the Kabsch algorithm [14], also known as a constrained orthogonal Procrustes problem subject to a unitary transformation. This algorithm returns the optimal rotation matrix that minimizes the RMSE between all or a subset of $\mathbf{X}$ and the transformed corresponding $\hat{\mathbf{X}}$.

This rotation matrix obtained with the Kabsch algorithm is also guaranteed to have a determinant of 1, meaning that there is no distortion when applied to the EDM.

Recall that the reconstruction is only accurate up to translation, rotation and reflection. Crucially, reflection. Since the Kabsch algorithm only returns a rotation matrix, the "handedness" of the reconstruction $\hat{\mathbf{X}}$ must be corrected to conform to the true, known set $\mathbf{X}$. This requires a subset ($\mathbf{X}_\mathrm{sub}$ and their reconstructed points $\hat{\mathbf{X}}_\mathrm{sub}$) of at least 3 known point pairs other than $\mathbf{x}_1$ and $\hat{\mathbf{x}}_1$, which is assumed to be the origin of the shared coordinate systems. As before, $\mathbf{x}_1$ and $\hat{\mathbf{x}}_1$ are first used to align $\mathbf{X}$ and $\hat{\mathbf{X}}$ (and by extension,

their subsets) with this assumption:

$$\mathbf{X}_{\mathrm{t}} = \mathbf{X} - \mathbf{1}_n \mathbf{x}_1 \quad \text{where} \quad \mathbf{X} \in \mathbb{R}^{n \times k} \quad \text{and} \quad \mathbf{x}_1 \in \mathbb{R}^{1 \times k} \tag{2.31}$$

$$\hat{\mathbf{X}}_{\mathrm{t}} = \hat{\mathbf{X}} - \mathbf{1}_n \hat{\mathbf{x}}_1 \quad \text{where} \quad \hat{\mathbf{X}} \in \mathbb{R}^{n \times k} \quad \text{and} \quad \hat{\mathbf{x}}_1 \in \mathbb{R}^{1 \times k} \tag{2.32}$$

Next, the normal unit vector to the plane defined by the first two points in the subsets is determined:

$$\hat{\mathbf{v}} = \left\| \hat{\mathbf{x}}_{\mathrm{sub}_1} \times \hat{\mathbf{x}}_{\mathrm{sub}_2} \right\|_2 \tag{2.33}$$

$$\mathbf{v} = \left\| \mathbf{x}_{\mathrm{sub}_1} \times \mathbf{x}_{\mathrm{sub}_2} \right\|_2 \tag{2.34}$$

Finally, the signs of the dot products between these normal unit vectors and the third point in the respective subsets is compared:

$$\mathrm{sgn}(\mathbf{v} \cdot \left\| \mathbf{x}_{\mathrm{sub}_3} \right\|_2) \stackrel{?}{=} \mathrm{sgn}(\hat{\mathbf{v}} \cdot \left\| \hat{\mathbf{x}}_{\mathrm{sub}_3} \right\|_2) \tag{2.35}$$

If the signs are equal, it can be concluded that the "handedness" of the systems are congruent and a rotation matrix can be determined directly using the Kabsch algorithm (by taking $\mathbf{P}$ as $\mathbf{I}_3$ in Eq. 2.36). If the "handedness" differs, the subset $\hat{\mathbf{x}}_{sub}$ must be reflected across the plane defined by the normal unit vector $\hat{\mathbf{v}}$. To do this, the Householder matrix [15] $\mathbf{P}$ is defined and applied to the subset and reconstructed set:

$$\mathbf{P} = \mathbf{I}_3 - 2\hat{\mathbf{v}}^\top \hat{\mathbf{v}} \tag{2.36}$$

$$\hat{\mathbf{X}}_{\mathrm{refl}_{\mathrm{sub}}} = \hat{\mathbf{X}}_{\mathrm{sub}} \mathbf{P} \tag{2.37}$$

$$\hat{\mathbf{X}}_{\mathrm{refl}} = \hat{\mathbf{X}}_{\mathrm{t}} \mathbf{P} \tag{2.38}$$

This Householder matrix has the useful property of being guaranteed to have a determinant of -1, meaning that no distortion of the EDM will occur, only a reflection of the points.

Once the handedness is confirmed, the Kabch algorithm can be applied:

$$\mathbf{H} = \hat{\mathbf{X}}_{\mathrm{refl}_{\mathrm{sub}}}^\top \mathbf{X}_{\mathrm{sub}} \tag{2.39}$$

Taking the singular value decomposition (SVD) of $\mathbf{H}$ (which can be seen as a cross-covariance matrix of $\hat{\mathbf{X}}_{\mathrm{refl}_{\mathrm{sub}}}^\top$ and $\mathbf{X}_{\mathrm{sub}}$):

$$\mathbf{H} = \mathbf{U} \Sigma \mathbf{V}^\top \tag{2.40}$$

The rotation matrix is then calculated:

$$\mathbf{T} = \mathbf{V}\mathbf{U}^{\top} \tag{2.41}$$

Finally, this rotation matrix $\mathbf{T}$ is applied to $\hat{\mathbf{X}}_{\mathrm{t}}$ to rotate the set of points:

$$\hat{\mathbf{X}}_{\mathrm{t}}\mathbf{T} = \hat{\mathbf{X}}_{\mathrm{rot}} \tag{2.42}$$

The Kabsch algorithm alongside the Householder matrix has the advantage of needing only 4 known points to work, while additional points increase the accuracy of the rotation matrix obtained. This algorithm is also guaranteed to introduce no distortions in the reconstructed points since the resultant transformation is unitary:

$$\Big|\det(\mathbf{T}) \cdot \det(\mathbf{P})\Big| = 1 \tag{2.43}$$

Chapter 3

Probabilistic Reconstruction Methods

"The way to control and direct a Mentat, Nefud, is through his information. False information—false results."

— Baron Harkonnen, *Dune* [16]

In contrast to the deterministic methods of Ch. 2, this chapter provides an alternative probabilistic solution to the reconstruction problem. This method has the advantage of using potential prior information effectively and being a simpler method to implement with less moving parts, ergo fewer sources of error.

This chapter introduces the probabilistic graphical model (PGM), a way to represent the belief of the system concerning the locations of the reconstructed points and how belief propagates across this network.

Using this model, along with prior distributions describing known positional knowledge and the unscented transform (UT) to "couple" the positional distributions based on the observed distance between them, the beliefs regarding the positions of the reconstructed points converge to values that are then returned.

A flowchart describing this method is given below:

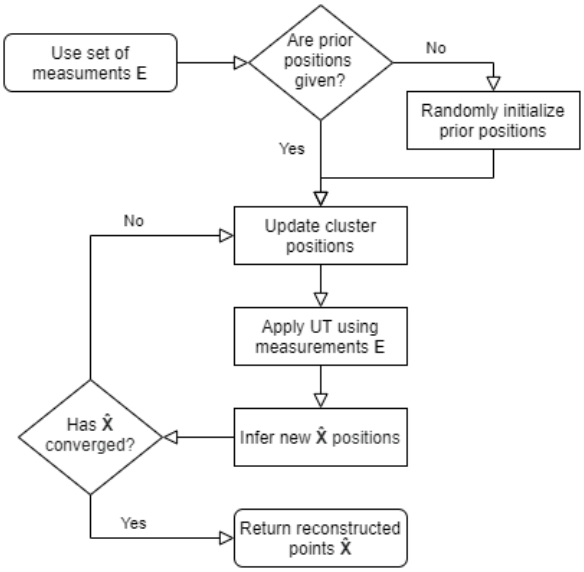


Figure 3.1: Probabilistic Reconstruction Method Flowchart

3.1. Probabilistic Graphical Model

Graphs, in the context of graph theory, are defined as a set of points (also called nodes or vertices) where in which some pairs of these points are connected by edges [17]. Edges may also be directed (i.e. edge has an orientation, e.g. from point a to point b) or undirected (i.e. edge orientation is irrelevant). A graph consisting of only directed edges is therefore known as a directed graph and, conversely, a graph with only undirected edges is known as undirected.

Probabilistic graphical models (PGM) are a specific subset of these graphs where the nodes of the graph represent dependencies/independencies in probability (or more general) distributions of random variables, showing how these random variables influence each other [18, p. 58].

3.1.1. Bayes and Markov Networks

Directed acyclic graphs (DAG) with nodes that are defined with conditional probability distributions (CPD) are known as Bayes Networks. These CPDs that define the graph G must satisfy the form $P(X \mid \mathrm{Pa}_X^G)$, stating that the conditional probability of a random variable can only be conditional on the direct parents of that node. An example can be seen in Fig. 3.2a and its factorization in Eq. 3.2.

These Bayes networks model causal influence between its nodes very intuitively and show the causal flow of influence along its directed edges.

Bayes networks, however, cannot capture the mechanics of all systems, such in the famous XOR case [19, p. 82].

To solve this problem, a more general definition of the Bayesian network is used, that of the Markov random field (MRF) or Markov network. These networks can be cyclical, do not have directed edges and use a more general definition of distributions (that need not be CPDs) called factors. These factors map the random variables over which they are defined (the scope of the factor) to values, normally constrained to be positive.

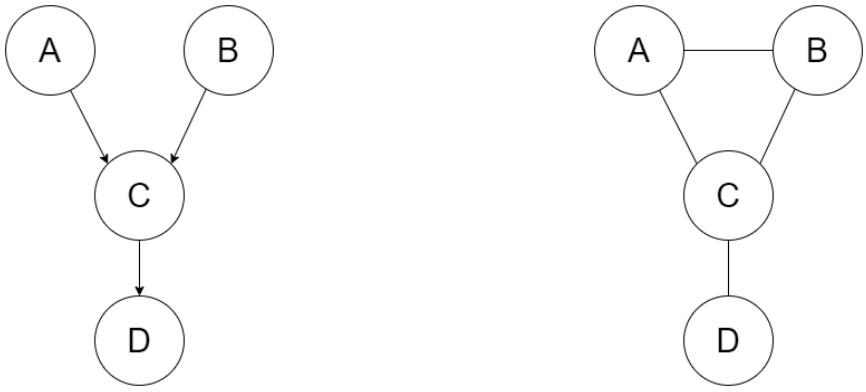
An example of a factor, $\phi(A, B)$, for $A, B \in \{0, 1\}$ is given by:

$$\phi(A,B) = \begin{array}{cc|c} A^0 & B^0 & 1 \\ A^0 & B^1 & 0 \\ A^1 & B^0 & 5 \\ A^1 & B^1 & 1000 \end{array} \tag{3.1}$$

It is important to note is the fact that these factors do not have to sum to 1, since they do not necessarily have to be probability distributions. Therefore, to convert the product of the factors that factorize a MRF (known as an unnormalized measure) to a valid, joint probability distribution, a normalizing factor of $\frac{1}{Z}$ must be added to this product, where

Z is known as the partition function, as seen in Eq. 3.3.

These two graph types can be converted to the other through a process known as moralisation[1]. This is visually shown by the two equivalent graphs of Fig. 3.2:



(a) Bayes Network **(b)** Markov Network

Figure 3.2: Equivalent Bayes and Markov Networks

The Bayes network in Fig. 3.2a is defined by the equation:

$$P(A, B, C, D) = P(A)P(B)P(C \mid A, B)P(D \mid C) \tag{3.2}$$

And its Markov network equivalent (obtained through moralisation) in Fig. 3.2b is defined by the equation:

$$\begin{aligned} P(A, B, C, D) &= \frac{1}{Z}\phi(A, B, C)\phi(C, D) \\ \text{with} \quad Z &= \sum_{A,B,C,D} \phi(A, B, C)\phi(C, D) \end{aligned} \tag{3.3}$$

For the application of reconstruction, taking the positional coordinates of the reconstructed points as random variables, an undirected MRF model is more fitting. Since distance is bidirectional, a Bayes network with directed edges would be extremely cyclical to account for this bidirectional flow of positional information between the points.

An additional problem arises from taking the positional coordinates of the reconstructed points as random variables, since these coordinates are not discrete. Therefore, the next section will detail the continuous extension of the MRF using Gaussian distributions.

As an additional note, while it is certainly possible to discretise the coordinate space for the positional random variables, to do so for any acceptable level of accuracy would be prohibitively computationally expensive given the fact that the size of a factor grows quadratically with the possible values that its random variables can take.

[1] A humorous convention, since the direct parents of a node become connected, a pseudo-"moral" thing to do.

An additional, although relative, benefit of using continuous factors is that the computational complexity to describe the network parameters drops from exponential ($O(2^n)$) to cubic ($O(n^3)$) complexity.

3.1.2. Gaussian Markov Random Fields

As mentioned previously in Sec. 3.1.1, the space of parameterisations for representing a continuous variable with discrete factors is essentially unbounded [19, p. 247].

This subsection details the extension of the Markov Random Field (MRF) to the continuous case using the alternate representation of multivariate Gaussian distributions, the information form.

The standard form of representing a multivariate distribution with a mean vector $\boldsymbol{\mu}$ and a $n \times n$ covariance matrix $\boldsymbol{\Sigma}$ is given by the following equation:

$$p(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^n |\, \boldsymbol{\Sigma} \,|}} \exp\Big[-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \Big] \tag{3.4}$$

Considering the exponent in Eq. 3.4 and taking the inverse of the covariance matrix as $\mathbf{J}$ (known as the precision or information matrix), the exponent can be rewritten:

$$-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu})^\top \mathbf{J} (\mathbf{x} - \boldsymbol{\mu}) = -\frac{1}{2}(\mathbf{x}^\top \mathbf{J}\mathbf{x} - 2\mathbf{x}^\top \mathbf{J}\boldsymbol{\mu} + \boldsymbol{\mu}^\top \mathbf{J}\boldsymbol{\mu}) \tag{3.5}$$

Noting that the last term is a constant:

$$p(\mathbf{x}) \propto \exp\Big[-\frac{1}{2}\mathbf{x}^\top \mathbf{J}\mathbf{x} + (\mathbf{J}\boldsymbol{\mu})^\top \mathbf{x} \Big] \tag{3.6}$$

This alternate representation is known as the information form, with $\mathbf{h} = \mathbf{J}\boldsymbol{\mu}$ known as the potential vector [19, p. 248].

The advantage of this form is that independence between variables can be read directly form the information matrix. This can be seen below in Th. 7.2 from Koller [19, p. 251].

Koller Theorem 7.2. *Consider a Gaussian distribution* $p(X_1, ..., X_n) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$*, and let* $\boldsymbol{J} = \boldsymbol{\Sigma}^{-1}$ *be the information matrix. Then* $\boldsymbol{J}_{i,j} = 0$ *if and only if* $p \models (X_i \perp X_j \mid \mathcal{X} - \{X_i, X_j\})$

Since that the information matrix can be used to describe the independecies in a set a continuous variables, the information matrix can now be used to capture the independencies of a Markov network (the nodes that are not connected).

Therefore, using this theorem, any Markov network of continuous variables can be represented by a multivariate Gaussian in the information form.

3.2. Unscented Transform

Introduced as part of the unscented Kalman filter (UKF)[2] by Julier and Uhlmann [20], the unscented transform approximates a non-linear transformation of a Gaussian distribution through deterministic sampling of a minimal set of sample points (known as sigma points) transformed by a non-linear function which is used to estimate a new mean and covariance of the transformed Gaussian.

From [20], the Unscented Transform of the d-dimensional random variable $\mathbf{X}$ with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ can be described by $2d+1$ weighted points $\boldsymbol{\mathcal{X}}$:

$$\boldsymbol{\mathcal{X}}_0 = \boldsymbol{\mu} \qquad W_0 = \frac{\kappa}{d+\kappa} \tag{3.7}$$

$$\boldsymbol{\mathcal{X}}_i = \boldsymbol{\mu} + \left(\sqrt{(d+\kappa)\boldsymbol{\Sigma}}\right)_i \qquad W_i = \frac{\kappa}{2(d+\kappa)} \tag{3.8}$$

$$\boldsymbol{\mathcal{X}}_{i+d} = \boldsymbol{\mu} - \left(\sqrt{(d+\kappa)\boldsymbol{\Sigma}}\right)_i \qquad W_{i+d} = \frac{\kappa}{2(d+\kappa)} \tag{3.9}$$

where $\boldsymbol{\mathcal{X}} \in \mathbb{R}^{d\times(2d+1)}$ and $\kappa \in \mathbb{R}$

W_i indicates the weight of the ith point and $\left(\sqrt{(d+\kappa)\boldsymbol{\Sigma}}\right)_i$ denotes the ith row or column of the matrix square root $\mathbf{L}$ obtained using the Cholesky decomposition:

$$\left(\sqrt{(d+\kappa)\boldsymbol{\Sigma}}\right) = \mathbf{L}\mathbf{L}^\top \tag{3.10}$$

These points can then be transformed using a potentially non-linear function:

$$\boldsymbol{\mathcal{Y}}_i = f(\boldsymbol{\mathcal{X}}_i) \tag{3.11}$$

The approximated new mean is given by the weighted average of the transformed points and the approximated new covariance matrix is given by the weighted outer product of the transformed points:

$$\tilde{\boldsymbol{\mu}} = \sum_{i=0}^{2d} W_i \boldsymbol{\mathcal{Y}}_i \tag{3.12}$$

$$\tilde{\boldsymbol{\Sigma}} = \sum_{i=0}^{2d} W_i (\boldsymbol{\mathcal{Y}}_i - \tilde{\boldsymbol{\mu}})(\boldsymbol{\mathcal{Y}}_i - \tilde{\boldsymbol{\mu}})^\top \tag{3.13}$$

This approximated mean and covariance is then used to construct the new Gaussian

[2]The apocryphal origin of this name is purported to either be Uhlmann not wanting the UKF to be named after himself or that the UKF was named in response to the "rotten"[sic] extended Kalman filter, although no extensive academic research has been done to ascertain the veracity of these claims.

distribution for the transformed random variable $\mathbf{y}$:

$$\mathbf{Y} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}) \tag{3.14}$$

Eq. 3.11 can also be used to increase the dimension of the original Gaussian. By observing this additional axis, the original distribution can be shifted according to Eq. 3.11.

This hyperplane created by observing the additional axis is demonstrated in the reduced dimensional example in Fig. 3.3 below. In this example, the random variable x, represented by a zero-mean Gaussian, gains an additional dimension y using the unscented transform given by the following simple linear transform:

$$y = x - 4 \tag{3.15}$$

Observing values for y creates the hyperplanes that intersect the transformed multivariate Gaussian as seen in Fig. 3.3. By inspecting the means of the new Gaussians defined in x, it is clear that the values of the new means are roughly 4 more than the observed values, as expected from Eq. 3.15.

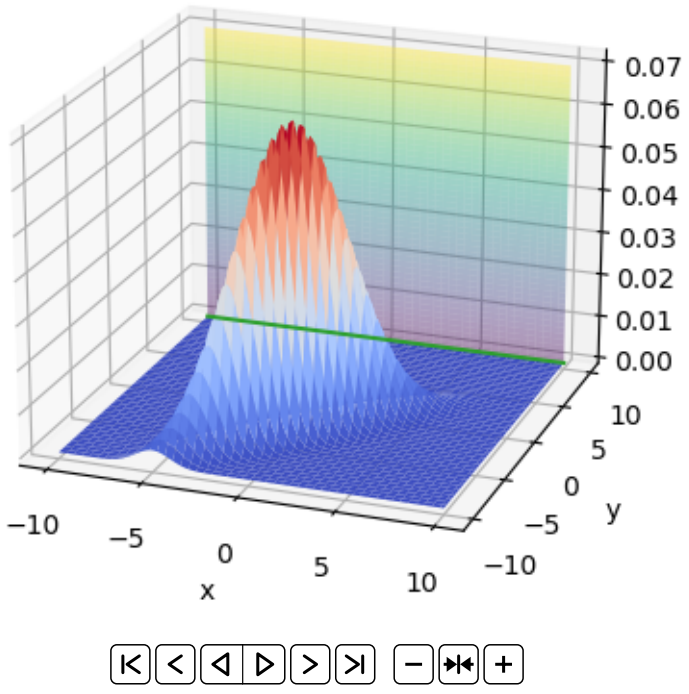


Figure 3.3: Simplified Hyperplane Example[3]

For the application of reconstruction, this transform is used to ”couple” two three-dimensional Gaussians (the prior coordinates of the points) by adding an additional

[3]Requires Adobe Acrobat or Okular PDF viewer to display animation.

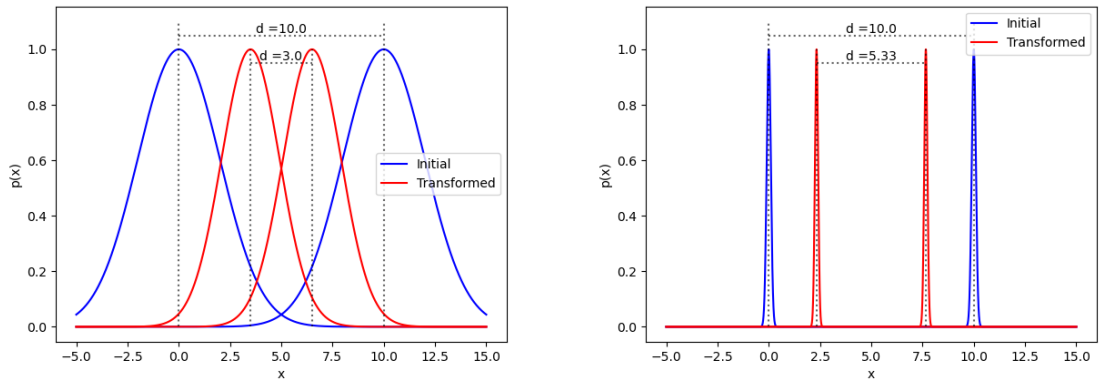
dimension that is a non-linear function (Euclidean distance between the points) of the other six positional dimensions:

$$\mathbf{Y} = f(\mathbf{X}) \tag{3.16}$$

$$\text{where} \quad \mathbf{X} = \begin{bmatrix} x_1 \\ \vdots \\ x_6 \end{bmatrix} \quad \text{and} \quad \mathbf{Y} = \begin{bmatrix} x_1 \\ \vdots \\ x_6 \\ d \end{bmatrix} \quad \text{and} \quad d = \sqrt{(x_1 - x_4)^2 + (x_2 - x_5)^2 + (x_3 - x_6)^2}$$

By observing the distance d value of $\mathbf{Y}$, a hyperplane through the seven-dimensional distribution is taken that relates the two constituent three-dimensional distributions according to their measured distance.

This can be observed in the simplified 2-dimensional case given by Fig. 3.4, where the two initial 1-dimensional Gaussians have a distance of 10 between their means. However, after applying the unscented transform as in Eq. 3.16 and observing the distance on the added axis to be 3, these distributions are shifted to conform to this observation.



(a) Standard Parameters **(b)** Lowered Initial Variance

Figure 3.4: Simplified Unscented Transform with Observed Distance Example

When looking at Fig. 3.4b, it is enlightening to note, however, that this process may struggle when the locations of the initial distributions are well defined (i.e. low variance). In a sense, the system is already certain of the distance between the points and is very resistant to change this belief. This process also reduces the variance of the points at each step, potentially causing numerical issues, such as underflow, after repeated applications and must be considered during implementation.

3.3. Positional Inference

At each iteration of the loop for the probabilistic reconstruction algorithm, after the prior positions have been readjusted using the observed distances alongside the UT, known positions have been observed and the new positional beliefs have been propagated across the network using the joint distribution, new positional distributions must be inferred to serve as priors for the next iteration.

By saving a copy of these inferred positions, inferred positions of the next loop of the probabilistic algorithm (implemented in Sec. 5.3), can be compared against until a certain tolerance of convergence is reached.

3.3.1. Marginal Inference

As seen in Eq. 3.3, a Markov Network can be represented as a joint probability distribution by taking the normalized product of its factors.

Marginal inference is a simple method of inference that computes the distribution of a subset of variables from the distribution that is possibly conditioned on a set of evidence [18, p. 77].

As an example, consider the joint probability distribution of the discrete random variables $P(x_1, x_2, x_3, x_4)$ and the observed evidence $x_4 = 2$. For the distribution of $p(x_1)$:

$$P(x_1 \,|\, x_4 = 2) = \sum_{x_2, x_3} P(x_1, x_2, x_3, x_4 = 2) \tag{3.17}$$

Applying this to the case of reconstruction, assuming that every point in a set of n points is described by three random continuous variables ($\mathcal{P}_i = \{p_{i_x}, p_{i_y}, p_{i_z}\}\ \forall 1 \leq i \leq n$ from the set of all positional random variables $\mathcal{P}$), the marginal inference can repeatedly by applied to the joint distribution of all of the positional random variables conditioned on the subset of known positions (e.g. the set $\mathcal{O} = \{p_{j_x} = x_{j_{\text{obs}}}, ..., p_{k_z} = z_{k_{\text{obs}}}\}$ where $j, k \in [1, 2, ..., n]$):

$$P(p_{i_x} \,|\, \mathcal{O}) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} P(p_{1_x}, p_{1_y}, p_{1_z}, ..., p_{n_x}, p_{n_y}, p_{n_z}, \mathcal{O}) dp_{1_x} dp_{1_y} ... dp_{n_z} \tag{3.18}$$

From these distributions for the positional random variables, the maximum a posteriori (MAP) estimate can be taken which, since these are Gaussian distributions, is the mean.

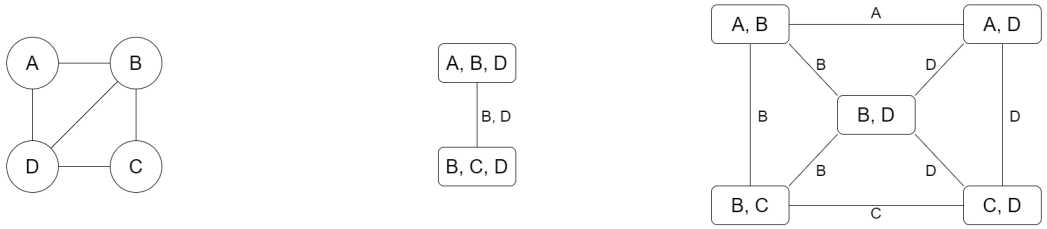
3.3.2. Junction Tree Algorithm

As shown in Sec. 3.1.1 and 3.3.1, propagating belief and inferring the values of the variables in a Markov network requires marginalising the full joint probability of all of the random variables in the network (obtained by taking the product of all of the factors as seen in

Eq. 3.3).

Calculating this joint probability, however, is computationally expensive and may be intractable for large systems.

A more efficient algorithm for the purpose of belief propagation is that of the junction tree algorithm. This algorithm constructs a new graph of connected subsets of the set of random variables ("clusters") that satisfies the running intersection property[4] (RIP) and where every cluster is fully connected ("cliques"). Belief about variables shared between subsets ("sepset") is propagated using the message passing algorithm. A junction tree of Fig. 3.5a is given by Fig. 3.5b.



(a) Simple Markov Network **(b)** Junction Tree of Fig. 3.5a **(c)** Cluster Graph of Fig. 3.5a

Figure 3.5: Junction Tree and Cluster Graph of Simple Markov Network

Fortunately, the size of the reconstruction problem in this report is small enough to simply use the joint distribution of all of the positional random variables[5].

3.3.3. Loopy Belief Propagation

In the case of large networks or networks with high interconnectivity, it may not be possible to construct a junction tree of Sec. 3.3.2 or the gains from doing so would be slim as the cliques would be very large. In these cases, alternative, approximate inference algorithms may be used such as loopy belief propagation.

This algorithm uses techniques similar to those in the junction tree algorithm (Sec. 3.3.2), notably message passing. The important distinction, however, is that while the junction tree (Fig. 3.5b) is guaranteed to be free of loops due to being constructed from cliques, in the loopy belief propagation algorithm constructs a cluster graph (Fig. 3.5c) using clusters that contain the random variables of single measurement pairs. This differing construction creates the possibility of loops in the cluster tree, meaning that the the inferred positions obtained using message passing is now iterative and approximate.

[4]If a variable is in two different clusters, then the variable is also in every cluster and sepset on the path between these two clusters

[5]In some sense, the full joint is simply the simplest case of the junction tree algorithm, with the singular cluster being the full set of variables

Chapter 4

Finding Erroneous Distance Measurements

> *Then, as his planet killed him, it occurred to Kynes that his father and all the other scientists were wrong, that the most persistent principles of the universe were accident and error.*
>
> — Liet Kynes, *Dune* [16]

Beside the expected measurement noise, in practice there may be distance records that are completely wrong, either measured between the wrong pair of points or the distance could be wildly incorrect. This may be due to human or machine error and these errors must be identified and removed to improve the reconstruction performance.

This chapter introduces a heuristic method in Sec. 4.1 for identifying these measurements based on the relative change between the original distance measurements and the distance measurements of the reconstructed points. A flowchart describing this method is given below in Fig. 4.1 below.

A possible improvement of this method is proposed in Sec. 4.2, which eschews the heuristic method for a probabilistic one. However, since the development of this method is outside the scope of this report, only the feasibility of this method is explored.

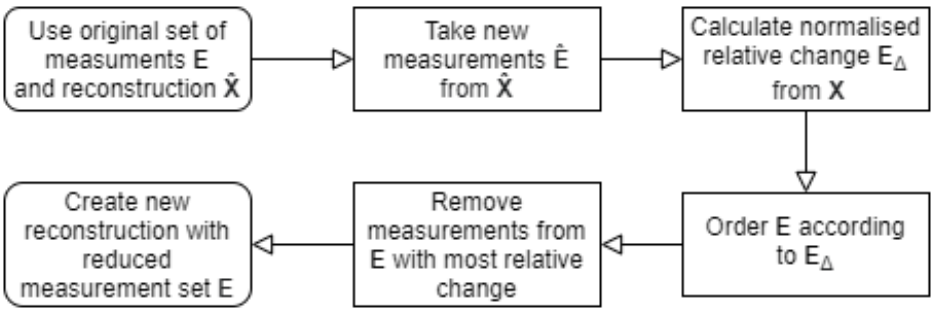


Figure 4.1: Heuristic Error Identification Method Flowchart

4.1. Heuristic Identification

The naive intuition used by this method is that the distances that change the most in the reconstruction versus the original distance measurement should be the ones that are anomalous. This feels logical, since if a point is described by ten distances from other points and one of these measurements are wrong, the point is "pulled" to the correct position against the influence of the incorrect measurement.

Therefore, ordering the distance measurements in the set $\mathcal{E}$ by their normalized, relative change when compared to the corresponding distances in the reconstructed set $\hat{\mathcal{E}}$:

$$\mathcal{E}_\Delta = \frac{|\,\mathcal{E} - \hat{\mathcal{E}}\,|}{\mathcal{E}} \tag{4.1}$$

$$\mathcal{E}_{\text{ord}} = \text{sort}_{\text{asc}}(\mathcal{E}\,|\,\mathcal{E}_\Delta) \tag{4.2}$$

Now $\mathcal{E}_{\text{ord}}$ represents the distance measurements used in the reconstruction that are in ascending order based on their relative change after reconstruction. From this, the last $x\%$ of measurements can be removed based on the error prevalence ($\text{err}_\%$) in the set. This reduced set of measurements are then used to reconstruct a new set of points with reduced reconstruction error.

This method can be used by both the deterministic- and probabilistic methods of Ch. 2 and 3, since it is only concerned with the original and reconstructed measurement sets ($\mathcal{E}$ and $\hat{\mathcal{E}}$), not the method of reconstruction.

A limitation of this method is that the error prevalence percentage is something that must be known beforehand or determined experimentally during implementation.

It is also possible that measurements may be corrupted to values that are still somewhat valid, leading to situations where valid measurements that have been "pulled" by the presence of another anomalous measurement may score higher than these somewhat valid distances. In the worst case, these valid measurements may even be discarded by this heuristic, a situation that should obviously be avoided. This problem can be mitigated somewhat by only discarding a portion of the known error prevalence percentage (e.g. discard last $0.5 \times \text{err}_\%$ elements of set $\mathcal{E}_{\text{ord}}$).

In a similar vein, this method may also struggle in the presence of noise, since *all* of the measurements will be distorted by the noise, making it harder to rank the measurements based on the relative change metric.

4.2. Probabilistic Identification

A possible improvement of the heuristic method of Sec. 4.1 is to solve the problem probabilistically. This has the advantage of identifying erroneous measurements based on

neighbouring measurements, avoiding the problems of falsely identifying distorted valid measurements and needing to know the global error prevalence percentage beforehand as encountered during the heuristic method.

Since this will most likely be a Gaussian-based solution, it should also be more robust to noise.

This method of identification is outside the scope of this report, however, and remains a tantalizing avenue of potential research.

Chapter 5

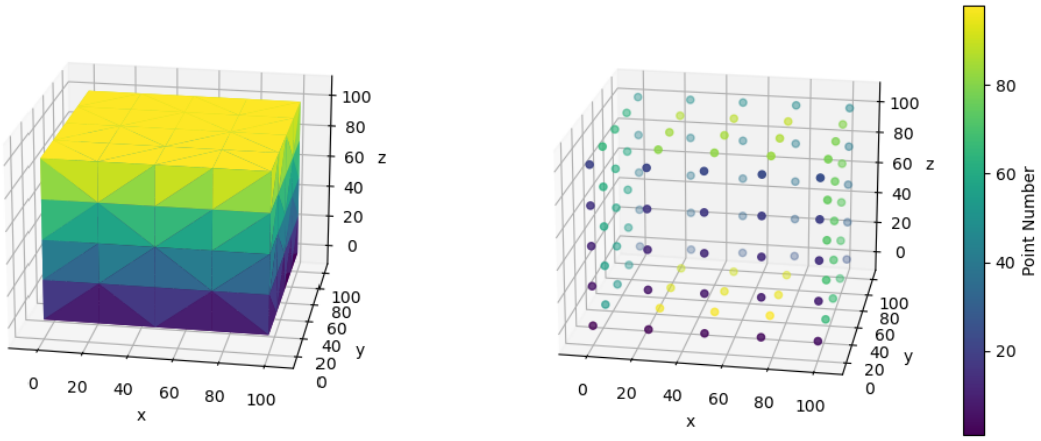
Experimental Setup

> *A beginning is the time for taking the most delicate care that the balances are correct. This every sister of the Bene Gesserit knows.*
>
> — from *"Manual of Muad'Dib"* by the Princess Irulan, *Dune* [16]

This chapter contains the implementation of the algorithms from Ch. 2 through 4 in Sec. 5.2 to 5.4, the generation of the datasets used by the algorithms in Sec. 5.1, the visualisation of reconstruction results in Sec. 5.5 and the error quantification of these results in Sec. 5.6.

5.1. Dataset Generation

For the testing purposes of this report, a virtual reference cube is constructed with 98 numbered points on its surface, such that every face has a grid of 5×5 points. This cube is visualised using the alpha shape (defined in Sec. 5.5) and the scatter plot of these points in Fig. 5.1a and 5.1b.



(a) Alpha Shape of Reference Object **(b)** Numbered Surface Points on Reference Object

Figure 5.1: Reference Object and Surface Points for Testing

From this set of true points a list of measurements can be taken according to a set of parameters to simulate the act of observing a physical object. These parameters include the

amount of connections per point, the noise added to the measurements and the percentage of records that are corrupted.

Using the Python NumPy [21] and pandas [22] libraries, the algorithm to generate this list of measurements is given below in Al. 4.2.

The format of the distance list is given in the Tab. 5.1 below. Note that the **Tolerance** column gives the standard deviation of the Gaussian noise added to the measurement and the **Changed** column contains a Boolean value that flags which record have been corrupted.

Index	Source	Target	Distance	Tolerance	Changed
0	1	8	55.89	0.56	False
1	1	14	90.38	0.90	False
...	...	...	...	...	...
246	86	98	102.54	1.03	False

Table 5.1: Example Distance Measurements List

5.2. Deterministic Reconstruction Algorithm Implementation

The implementation of this algorithm is based on the flowchart in Fig. 2.1 and its pseudocode is given in App. E. The software has been written in Python using the NumPy [21], pandas [22], SciPy [23] and scikit-learn [24] libraries and makes calls to Matlab to execute the script in App. C.

Recall from Sec. 2.3 that the semidefinite relaxation script proposed by Dokmanić *et al.* [9] contains the data fidelity term λ proposed by Biswas *et al.*. The value of this term is determined experimentally. The relative EDM error (as defined in Sec. 5.6) versus interconnection for various values of λ is plotted in Fig. 5.2.

It is clear from this figure that the network can perform reconstructions with less data (lower interconnection value) for higher values of λ, up to a certain point. This point seems to be the value $\lambda = 10$ and is the value used in this report. It is a point of possible further research to investigate whether this value holds for more general reconstruction problems or how the problem parameters affects this value.

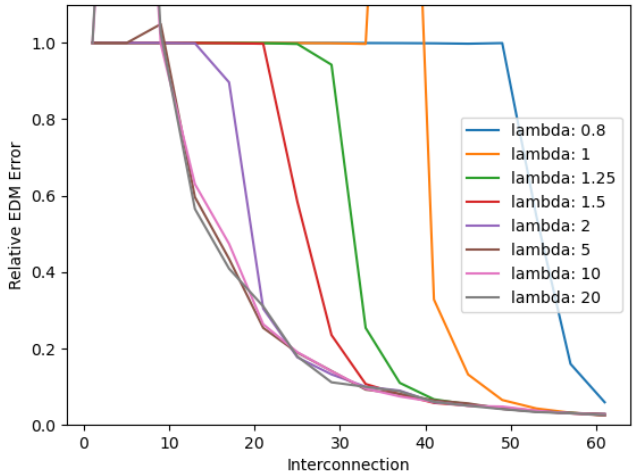
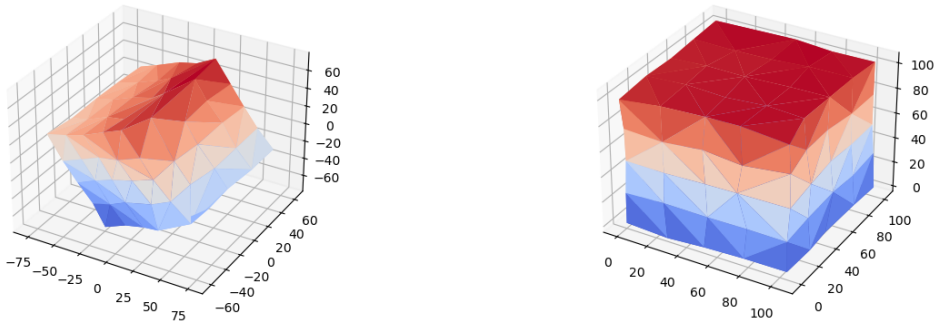


Figure 5.2: Deterministic Algorithm Error for Various λ Values

It is an interesting point to note that this algorithm can work with no prior knowledge of the positions of the true points (as opposed to the probabilistic algorithm that requires a prior distribution of the point positions), however this only produces a reconstruction up to translation, rotation and reflection as seen in Fig. 5.3. In some cases, where only EDM reconstruction is required, this may be an advantage when compared against the probabilistic algorithm which requires at least 3 known points to function.



(a) Unrotated Reconstruction **(b)** Rotated Reconstruction

Figure 5.3: Unrotated- and Rotated Deterministic Reconstructions

5.3. Probabilistic Reconstruction Algorithm Implementation

The implementation of this algorithm is based on the flowchart in Fig. 3.1 and its pseudocode is given in App. F. The software has been written in C/C++ using routines and structures from the proprietary PGM library EMDW.

A visualisation of the three steps at each iteration of this algorithm is given below in Fig. 5.4 (note that the observed distances are given by the dashed lines). These steps, in summary, are as follows:

1. Initialise factors from prior positions and connections defined in the distance list.
2. Update factor positions using the UT and the observed distances.
3. Infer new positions and update the prior for the next iteration.

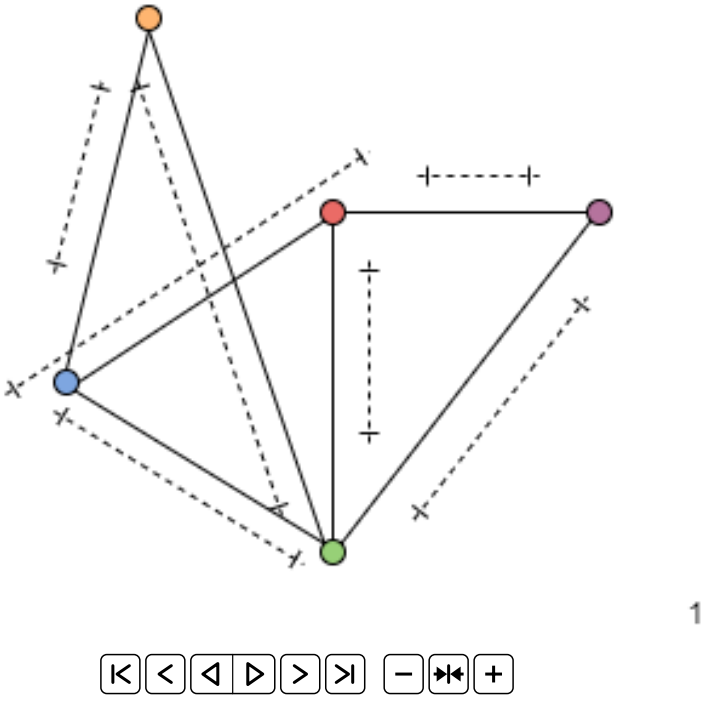


Figure 5.4: Simplified Probabilistic Algorithm Iteration Example[1]

This software creates a PGM using the dataset generated according to Sec. 5.1 and iteratively applies this to a list of prior positions (formatted as seen in Tab. 5.2) until convergence.

Index	Point_Num	X_Pos	Y_Pos	Z_Pos	X_Tol	Y_Tol	Z_Tol
0	1	0	0	0	0	0	0
1	2	31.74	7.38	11.58	10	10	10
...	...	...	...	...	...	...	...
97	98	88.65	25.69	10.27	10	10	10

Table 5.2: Example Prior/Result Positions List

[1]Requires Adobe Acrobat or Okular PDF viewer to display animation.

This list of positions represents the prior knowledge the algorithm receives regarding the positions of the points. For testing purposes, Gaussian noise is added to the true positions (with the standard deviation of this noise given in the **Tol** columns in Tab. 5.2) to simulate different levels of accuracy for this prior knowledge. If prior positions are not given, the positions are initialised randomly in predefined cube.

Dimensions with positions that are observed are represented in the list of positions by setting their values to the true values and the corresponding tolerance/standard deviation to zero. This is seen in the first record of Tab. 5.2, where this point has been chosen as the origin with all of its coordinates observed to be zero. It is important to note that this algorithm requires the positions of 3 observed points (excluding the origin) to function[2]. However, this nullifies the need for the rotation correction of the deterministic algorithm of Sec. 5.2.

In order to avoid the numerical issues mentioned in Sec. 3.2, an additional damping factor λ is added[3]. At each step where the UT is applied, the new variance is limited to the maximum of the new variance and the old variance multiplied by λ. This ensures that the system does not converge to sub-optimal positions too quickly and that the variances of the points do not underflow.

Convergence for this algorithm is calculated by taking the absolute difference of the positional beliefs at the start and end of each iteration. Once this difference is below a certain threshold, the system has converged. For this report, convergence is declared once the positions have reached 1% of their true positions (absolute difference of 294).

5.4. Error Identification Heuristic Implementation

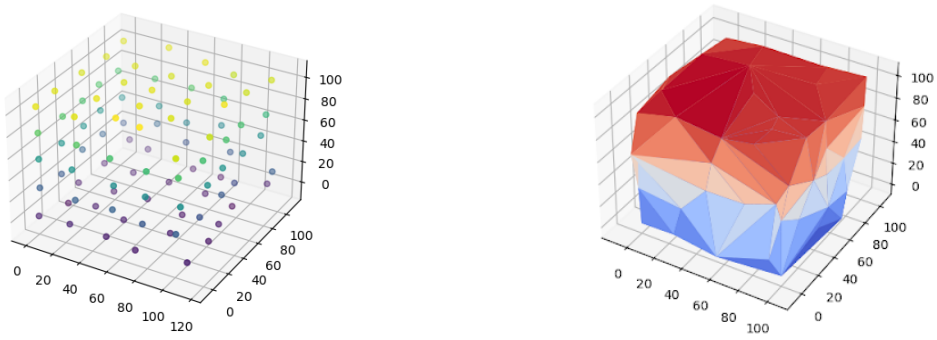
The implementation of this algorithm is based on the flowchart in Fig. 4.1 and its pseudocode is given in App. E. The software has been written in Python using the pandas [22] library.

5.5. Result Visualisation

Both of the algorithms from Sec. 5.2 and 5.3 return a set of 98 3-dimensional reconstructed points $\hat{\mathbf{X}}$. These points can be visualised using a simple scatter plot (Fig. 5.5a), but it can be hard to get a sense of the shape of the reconstructed object given the lack of depth in the image. To help in this regard, an alpha shape is constructed using these points and teh Alpha Shape Toolbox Python library [25] as seen in Fig. 5.5b.

[2]Since a 4 points are needed to orient a 3-dimensional volume with no ambiguity.

[3]Through trial and error, the value of $\lambda = 0.8$ seems to work in the most circumstances. This may be due to its proximity to the optimal damping ratio ($\frac{1}{\sqrt{2}}$) but no testing has been done to support this intuition.

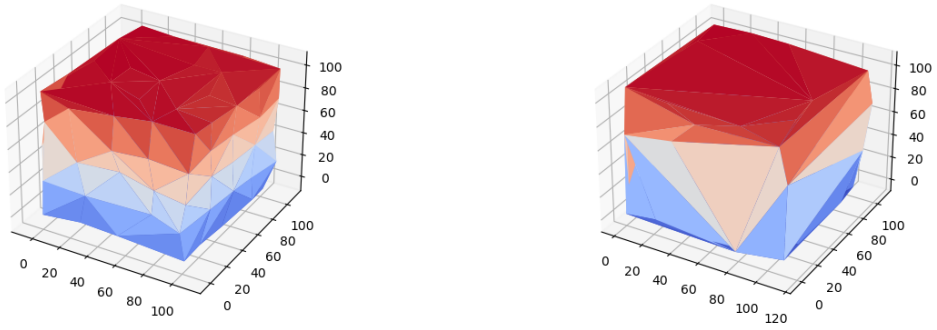


(a) Scatter Plot of Reconstructed Points Example **(b)** Alpha Shape of Reconstruction Example

Figure 5.5: Scatter and Alpha Shape Plot Examples

An alpha shape is defined as a minimum bounding polytope of a set of points [26], in a sense the ”shape” that would be obtained by stretching a rubber band around all of the points.

This method of visualisation requires the tuning of an α-parameter, from which the name ”alpha shape” derives, which sets how aggressively the shape is minimised by limiting the maximum length of the sides of the polytope. If this parameter is set too high, the process will fail and if it is set too high, surface detail may be lost (Fig. 5.6b as opposed to Fig. 5.6a). A simple heuristic to determine this value would be to take the reciprocal of the average distance between the reconstructed points, although this has not been extensively tested[4].



(a) Suitable $\alpha = 0.01$ **(b)** Excessive $\alpha = 0.001$

Figure 5.6: Alpha Shapes with Suitable and Excessive Values for α

[4]The value of $\alpha = 0.01$ used in the visualisations of this report was determined though trial and error, but roughly follows this heuristic.

5.6. Error Quantification

To provide an objective and quantised measure of the error of the reconstruction, a measure is required that is decoupled from the subjective measure of visual fidelity. The measure of relative Euclidean distance matrix error, as used by Dokmanić *et al.* [9], is used in this report.

This measure uses the EDM of the true points ($\mathbf{D}$) and the EDM of the reconstructed points ($\hat{\mathbf{D}}$) and compares the two. It is defined by the following equation:

$$\text{err}_{\text{rel}} = \frac{\left\| \hat{\mathbf{D}} - \mathbf{D} \right\|_F}{\left\| \mathbf{D} \right\|_F} \tag{5.1}$$

This provides a way to measure the reconstruction performance without needing to first correct the orientation of the reconstruction, since the EDM is the same for all orientations of the reconstruction.

Broadly, the behaviour of the err_{rel} with respect to the connections per point (interconnections) follows the pattern in Fig. 5.7, with an initial plateau, a large spike and a descent to a converged value. As a good rule of thumb, reconstructions with an error of less than $o.1$ can be regarded as good reconstructions.

The corresponding alpha shapes of reconstructions in these regions will be similar to those also given in Fig. 5.7.

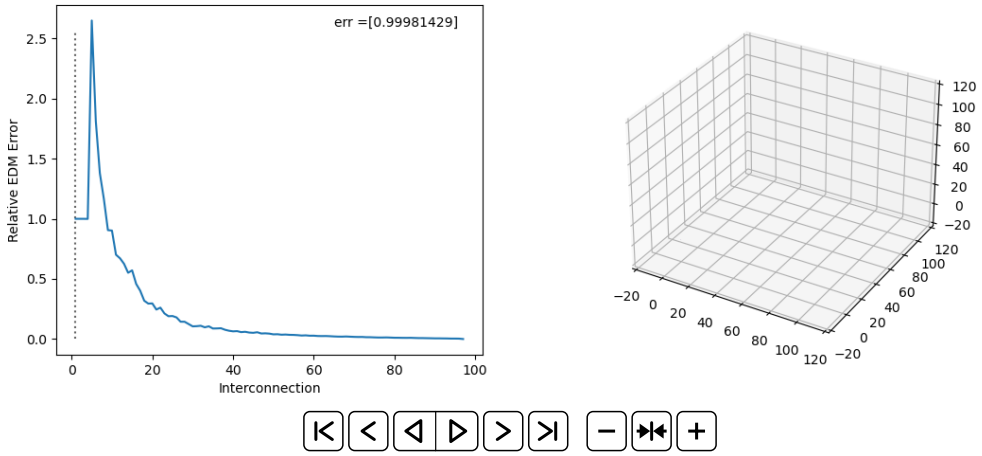


Figure 5.7: Reconstruction Behaviour Example[5].

[5]Requires Adobe Acrobat or Okular PDF viewer to display animation.

Chapter 6

Experimental Results

> *KULL WAHAD!: "I am profoundly stirred!" A sincere exclamation of surprise common in the Imperium. Strict interpretation depends on context. (It is said of Muad'Dib that once he watched a desert hawk chick emerge from its shell and whispered: "Kull wahad!")*
>
> — Terminology of the Imperium, *Dune* [16]

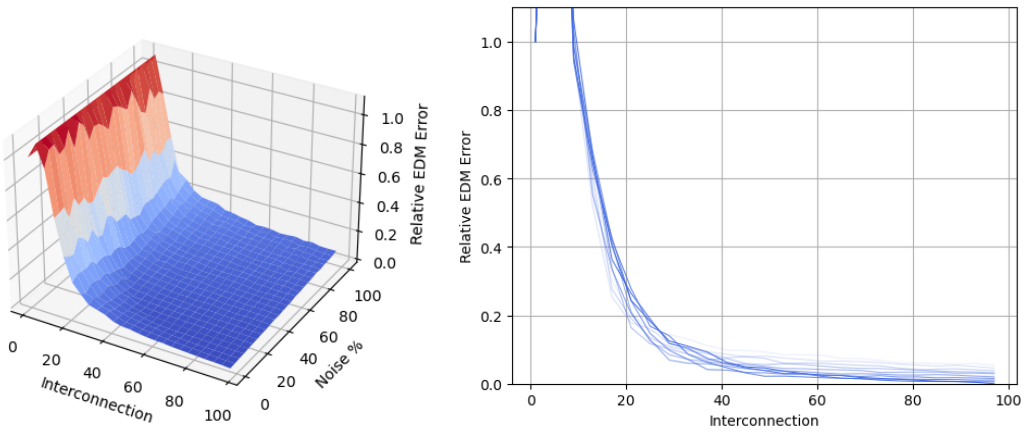
This chapter compiles the results of the reconstructions obtained from the algorithms from Ch. 2 and 3 using the testing framework of Ch. 5.

In the context of these results, the noise percentage added to the distance measurements during the dataset generation refers to the 3.5σ boundary of the added zero-mean Gaussian noise as a percentage of the measurement. Additionally, the error percentage refers to the percentage of records that have been corrupted by changing either the source point, target point or the measured distance.

Also note that the relative EDM error has been clipped to 1.1 in all of the graphs for better readability.

6.1. Deterministic Algorithm Performance

err surface, inter vs noise

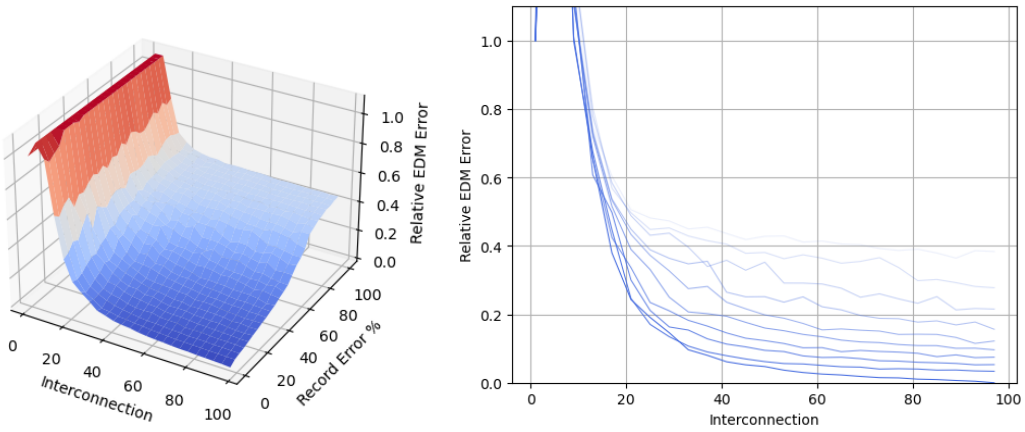


(a) Error Surface **(b)** Projected Line Graph

Figure 6.1: Error Surface and Line Graph for Network Interconnection and Measurement Noise Percentage with Deterministic Algorithm

It is clear that the deterministic algorithm is quite robust to Gaussian noise (as speculated in Sec. 2.2), returning good good reconstructions even in the presence of large amounts of noise.

err surface, inter vs err

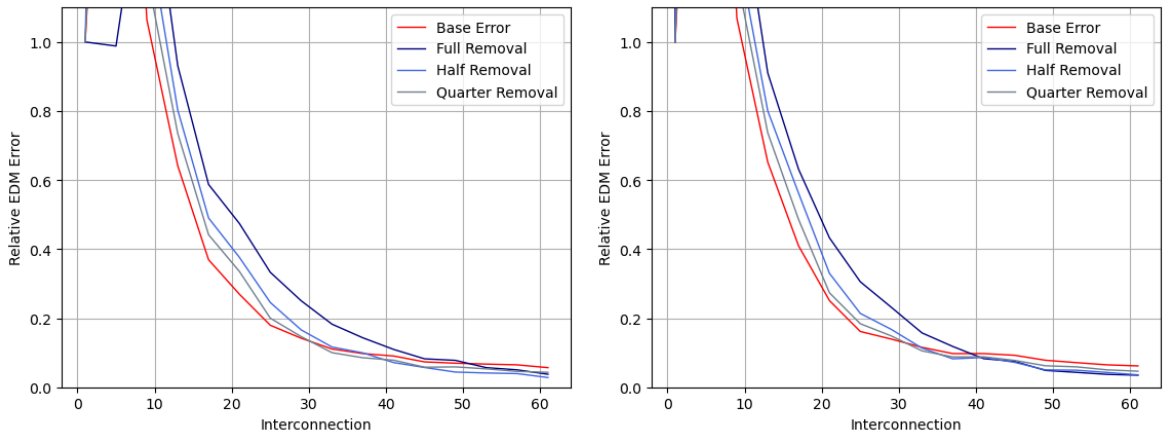


(a) Error Surface **(b)** Projected Line Graph

Figure 6.2: Error Surface and Line Graph for Network Interconnection and Record Error Percentage with Deterministic Algorithm

No longer returns good reconstructions at around the 40% record error mark.

Using the heuristic error identification algorithm of Ch. 4, the deterministic algorithm is applied to datasets with 10% record error and 1% (Fig. 6.3a) and 10% (Fig. 6.3b) Gaussian measurement noise. These graphs show the reconstruction performance of the deterministic algorithm after various portions of the identified records are removed.



(a) Error Surface **(b)** Line Graph

Figure 6.3: Relative Error of Deterministic Algorithm after Removing Corrupted Records Identified by Heuristic Algorithm

improves performance above ≈ 30 connections per point with half removal giving the best balance

6.2. Probabilistic Algorithm Performance

err inter vs init_std
err surface, inter vs noise with low and high init_std
err surface, inter vs err with noise levels and low and high init_std

ditch

6.3. Performance Comparison

comp inter vs noise

comp standard cicum (5% noise, 1% err)

comp ditch

show what prior knowledge is required

Chapter 7

Summary and Conclusion

Truth suffers from too much analysis.

— Ancient Fremen saying, *Dune Messiah* [1]

note that it can reconstruct EDMs, making it applicable to more general problems.

Bibliography

[1] F. Herbert, *Dune Messiah*. Ace, 1969.

[2] J. Dattorro, *Convex Optimization Euclidean Distance Geometry*. Meboo Publishing USA, 2005, ch. 5. [Online]. Available: https://books.google.co.za/books?id=VxuRF4NhjqkC

[3] W. Kaplan, *Maxima and Minima with Applications: Practical Optimization and Duality*, ser. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2011, pp. 61–62. [Online]. Available: https://books.google.co.za/books?id=bAo6KNZcUP0C

[4] J. C. Gower, "Euclidean distance geometry," *Mathematical Scientist*, vol. 7, pp. 1–14, 1982.

[5] W. Torgerson, "Multidimensional scaling: I. theory and method." *Psychometrika*, vol. 17, p. 401–419, 1952. [Online]. Available: https://doi.org/10.1007/BF02288916

[6] F. Wickelmaier, *An introduction to MDS*, ser. Aalborg Universitetscenter. Institut for Elektroniske Systemer. Afdeling for Kommunikationsteknologi. Rapport. Aalborg Universitetsforlag, 2003, no. R00-6003, p. 10.

[7] A. Buja, D. Swayne, M. Littman, N. Dean, H. Heike, and L. Chen, "Data visualization with multidimensional scaling," *Journal of Computational and Graphical Statistics*, vol. 17, pp. 444–472, Jun. 2008. [Online]. Available: https://doi.org/10.1198/106186008X318440

[8] E. Peterfreund and M. Gavish, "Multidimensional scaling of noisy high dimensional data," 2018. [Online]. Available: https://doi.org/10.1016/j.acha.2020.11.006

[9] I. Dokmanić, R. Parhizkar, J. Ranieri, and M. Vetterli, "Euclidean distance matrices: Essential theory, algorithms, and applications," *IEEE Signal Processing Magazine*, vol. 32, no. 6, pp. 12–30, 2015. [Online]. Available: https://doi.org/10.1109/MSP.2015.2398954

[10] N. Krislock and H. Wolkowicz, *Euclidean Distance Matrices and Applications*. Boston, MA: Springer US, Jan. 2012, pp. 879–914. [Online]. Available: https://doi.org/10.1007/978-1-4614-0769-0_30

[11] P. Biswas, T.-C. Liang, K.-C. Toh, Y. Ye, and T.-C. Wang, "Semidefinite programming approaches for sensor network localization with noisy distance measurements," *IEEE Transactions on Automation Science and Engineering*, vol. 3, no. 4, pp. 360–371, 2006. [Online]. Available: https://doi.org/10.1109/TASE.2006.877401

[12] MATLAB, *version 9.11 (R2021b)*. Natick, Massachusetts: The MathWorks Inc., 2021.

[13] M. Grant and S. Boyd, "Cvx: Matlab software for disciplined convex programming, version 2.1." Mar. 2014. [Online]. Available: http://cvxr.com/cvx

[14] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 32, no. 5, pp. 922–923, Sep. 1976. [Online]. Available: https://doi.org/10.1107/S0567739476001873

[15] A. S. Householder, "Unitary triangularization of a nonsymmetric matrix," *Journal of the ACM*, vol. 5, no. 4, p. 339–342, Oct. 1958. [Online]. Available: https://doi.org/10.1145/320941.320947

[16] F. Herbert, *Dune*. Chilton Books, 1965.

[17] R. Trudeau, *Introduction to Graph Theory*, ser. Dover Books on Mathematics. Dover Publications, 2013. [Online]. Available: https://books.google.gg/books?id=eRLEAgAAQBAJ

[18] D. Barber, *Bayesian Reasoning and Machine Learning*, ser. Bayesian Reasoning and Machine Learning. Cambridge University Press, 2012. [Online]. Available: https://books.google.co.za/books?id=yxZtddB_Ob0C

[19] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, ser. Adaptive computation and machine learning. MIT Press, 2009. [Online]. Available: https://books.google.co.za/books?id=7dzpHCHzNQ4C

[20] S. J. Julier and J. K. Uhlmann, "New extension of the Kalman filter to nonlinear systems," in *Signal Processing, Sensor Fusion, and Target Recognition VI*, I. Kadar, Ed., vol. 3068, International Society for Optics and Photonics. SPIE, 1997, pp. 182 – 193. [Online]. Available: https://doi.org/10.1117/12.280797

[21] NumPy, *version 1.21.1*. Community Project, 2021. [Online]. Available: https://numpy.org/

[22] pandas, *version 1.3.0*. Wes McKinney, 2021. [Online]. Available: https://pandas.pydata.org/

[23] SciPy, *version 1.7.1*. Community Project, 2021. [Online]. Available: https://scipy.org/

[24] scikit learn, *version 1.0.1*. Community Project, 2021. [Online]. Available: https://scikit-learn.org/

[25] A. S. Toolbox, *version 1.3.1*. Kenneth E. Bellock, 2021. [Online]. Available: https://pypi.org/project/alphashape/

[26] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel, "On the shape of a set of points in the plane," *IEEE Transactions on Information Theory*, vol. 29, no. 4, pp. 551–559, 1983. [Online]. Available: https://doi.org/10.1109/TIT.1983.1056714

Appendix A

Project Planning Schedule

Week	Date	Planned Progress
1	08/08/2021	Familiarise with problem
2	15/08/2021	Do literature review of deterministic methods (MDS, SDP)
3	22/08/2021	Do literature review of probabilistic methods (PGMs, UT)
4	29/08/2021	Implement deterministic algorithm
5	05/08/2021	Add testing framework for algorithms
6	12/09/2021	Add parallel execution functionality
7	19/09/2021	Obtain results from deterministic algorithm
8	26/09/2021	Implement probabilistic algorithm
9	03/10/2021	Obtain results from probabilistic algorithm
10	10/10/2021	Compile literature review in report
11	17/10/2021	Compile implementation in report
12	24/10/2021	Compile results in report
13	31/10/2021	Incorporate feedback
14	07/11/2021	Finalise Report

Table A.1: Project Schedule per Week

Appendix B

Outcomes Compliance

Outcomes	Chapters	Justification
ELO 1: Problem solving	1, 2, 3, 4	Identifying the reconstruction problem, solving the reconstruction problem from paired point distances using the theory from deterministic/probabilistic methods.
ELO 2: Application of scientific and engineering knowledge	2, 3, 4, 5	Application of abstract concepts from linear algebra, statistics, and probabilistic theory in designing algorithms to solve the reconstruction problem, including Euclidean distance matrices, multidimensional scaling, semidefinite programming, probabilistic graphical models, and the unscented transform.
ELO 3: Engineering Design	2, 3, 4, 5	In the implementation of the designed algorithms, various concepts are synthesized using non-procedural methods. The various abstract concepts are brought together to design novel procedures that use each of these concepts sequentially to return the desired results.
ELO 4: Investigations, experiments and data analysis	5, 6	This report contains the metrics and experimental framework to test the algorithms designed as well as the data analysis and presentation of these results obtained from this framework.
ELO 5: Engineering methods, skills and tools, including Information Technology	5, 6	Various engineering tools and skills have been used during this report, from software such as Python, C++ and MATLAB, mathematics during the algorithm design to presentation tools such as LATEX for this report.
ELO 6: Professional and technical communication	All chapters	This report in its entirety displays evidence of effective competence in professional and technical communication skills. This evidence is supplemented by the oral- and poster presentations that are to be presented.
ELO 8: Individual work	All chapters	All work contained in this report has been done alone by E. Visser, with advice from weekly consultations with supervisor Prof. J. du Preez. External sources required have been cited fully.
ELO 9: Independent learning ability	All chapters	The theory required for this report is not covered in the standard engineering curriculum and had to be self-taught and researched. Additional skills such as MATLAB and C++ also had to be acquired individually.

Table B.1: ECSA Outcomes Compliance

Appendix C

Matlab/CVX Script for Semidefinite Relaxation

This appendix contains the Matlab/CVX script [12] [13] used to solve the SDP-problem in Sec. 2.3. The script as witten by Dokmanić *et al.* [9] can be found below:

```
function [EDM, X] = semidefiniteRelaxation(D, W, lambda)

n = size(D, 1);
x = -1/(n + sqrt(n));
y = -1/sqrt(n);
V = [y*ones(1, n-1); x*ones(n-1) + eye(n-1)];
e = ones(n, 1);

cvx_begin sdp
        variable G(n-1, n-1) symmetric;
        B = V*G*V';
        E = diag(B)*e' + e*diag(B)' - 2*B;
        maximize trace(G) ...
                - lambda * norm(W .* (E - D), 'fro');
        subject to
                G >= 0;
cvx_end

[U, S, V] = svd(B);
EDM = diag(B)*e' + e*diag(B)' - 2*B;
X = sqrt(S)*V';
```

Appendix D

Dataset Generation Algorithm Pseudocode

Algorithm 4.2: Distance Measurements List Generation

Input: numPoints, reqCons, noisePercent, errorPercent
Output: distList

```
function getValidPoints(s, numPoints, reqCons)
    for t ← 1 to numPoints do
        if s ≠ t and cons(t) < reqCons then
            validSet ← validSet.append(t)
        end if
    end for
    return validSet
end function

for s ← 1 to numPoints do
    if cons(s) < reqCons then
        validSet ← getValidPoints(s, numPoints, reqCons)
        t ← pickRand(validSet)
        d ← getDist(s, t)
        d ← gaussianNoise(d, noisePercent)
        distList ← distList.append(s, t, d)
    end if
end for
for s ← 1 to numPoints*errorPercent do
    if cons(s) < reqCons then
        validSet ← getValidPoints(s, numPoints, reqCons)
        record ← pickRand(distList)
        record ← corrupt(record)
        distList ← distList.update(record)
    end if
end for
return distList
```

Appendix E

Deterministic Reconstruction Algorithm Pseudocode

Algorithm 5.3: Deterministic Reconstruction Implementation

Input: distList
Output: recPoints

```
numPoints ← MAX(MAX(distList["Source"]), MAX(distList["Target"]))
maskMat, edmMatRaw ← NUMPY.ZEROS((numPoints, numPoints))
truePoints ← GETREFERENCEPOINTS(numPoints)

for i ← 1 to distList.len do
    s ← disList[i]["Source"]
    t ← disList[i]["Target"]
    maskMat[s, t], maskMat[t, s] ← 1
    edmMatRaw[s, t], edmMatRaw[t, s] ← disList[i]["Distance"]
end for

edmMatRec ← CALLMATLABFUNCTION(SEMIDEFRELAXATION(edmMatRaw, λ = 10))
resPoints ← SKLEARN.MANIFOLDS.MDS(edmMatRec, components= 3)

truePoints ← truePoints - truePoints[0, :]
resPoints ← resPoints - resPoints[0, :]
trueSubset ← TAKESUBSET(truePoints, 3)
resSubset ← TAKESUBSET(resPoints, 3)
trueCross ← NUMPY.CROSS(trueSubset[1, :], trueSubset[2, :])
trueCrossNormed ← trueCross/NUMPY.NORM(trueCross)
resCross ← NUMPY.CROSS(resSubset[1, :], resSubset[2, :])
resCrossNormed ← resCross/NUMPY.NORM(resCross)
trueDot ← NUMPY.CROSS(trueCrossNormed, trueSubset[3, :])
trueDotNormed ← trueDot/NUMPY.NORM(trueDot)
resDot ← NUMPY.CROSS(resCrossNormed, resSubset[3, :])
resDotNormed ← resDot/NUMPY.NORM(resDot)
```

```
if NUMPY.SIGN(trueDotNormed) ≠ NUMPY.SIGN(resDotNormed) then
    householder ← IDENTITY(3) - 2*NUMPY.DOT(resCrossNormed.T, resCrossNormed)
    resPoints
    resPoints ← NUMPY.DOT(resPoints, householder)
end if

rotMat ← SCIPY.SPATIAL.TRANSFORM.ROTATION.ALIGN_VECTORS(trueSubset, resSubset)
resPoints ← NUMPY.DOT(resPoints, rotMat)
return resPoints
```

Appendix F

Probabilistic Reconstruction Algorithm Pseudocode

Algorithm 6.4: Probabilistic Reconstruction Implementation

Input: distList, tolerance, observations, prior_positions (optional)

Output: recPoints

```
function initClusters(distList, positions)
    for dist in distList do
        s ← dist["Source"]
        t ← dist["Target"]
        cluster ← add postions[s] to cluster
        cluster ← add positions[t] to cluster
        Add cluster to clusterList
    end for
    return clusterList
end function

function applyUT(distList, clusterList)
    for (dist, cluster) in (distList, clusterList) do
        sigmaPoints ← sample sigma points from cluster
        sigmaPoints ← add dimesion using Euclidean distance
        cluster ← reconstruct cluster from sigma points
        cluster ← observe cluster using measurement from dist
    end for
    return clusterList
end function
```

```
function inferNewPos(clusterList)
    joint ← multiply clusters to obtain joint
    for point in Points do
        positions ← marginalize joint and take MAP estimate for new point coords
        positions ← limit tol to max of new tol and (λ = 0.8) * old tol
    end for
    return positions
end function

numPoints ← max(max(distList["Source"]), max(distList["Target"]))
if Prior positions given then
    priorfactors ← prior_positions
else
    priorfactors ← init prior positions randomly
end if
postions ← set observed values and set their tol to zero
while change > (tolerance = 294) and iter < (iter_lim = 25) do
    old_positions ← store current positions
    clusterList ← initClusters(distList, positions)
    clusterList ← applyUT(distList, clusterList)
    positions ← inferNewPos(clusterList)
    change ← old_positions - positions
    iter ← iter + 1
    prior_positions ← positions (prior for next iteration)
end while
recPoints ← positions
return recPoints
```

Appendix G

Heuristic Error Identification Algorithm Pseudocode

Algorithm 7.5: Heuristic Error Identification

Input: distList, recPoints, errPerc
Output: distListReduced

```
for dist in distList do
    s ← dist["Source"]
    t ← dist["Target"]
    newDist ← DIST(recPoints[s], recPoints[t])
    dist["Change"] ← (dist["Dist"] - newDist) / dist["Dist"]
end for
distList ← Sort by "Change" column ascending
distListReduced ← Remove first errPerc * length(distList) records from distList
return distListReduced
```