Stiffness Mapping of Deformable Objects Through Supervised Embedding and Gaussian Process Regression

Evan Harber

CMU-RI-TR-22-14

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee:

Prof. Howie Choset (Advisor)
Prof. Wenzhen Yuan
Leonid Keselman

In Partial Fulfillment of the Requirements for the Degree of
Master of Science in Robotics

Acknowledgements

After getting both my Bachelor's and Master's at Carnegie Mellon University, I have many wonderful people to thank!

I would first like to thank my Advisor, Howie Choset; your support as my Research Advisor allowed me to pursue ideas and grow as an academic.

I would also like to thank Lu Li, who instilled in me a deep love of research and curiosity about how the world works, two traits that I hope to carry forward wherever I go next.

Next, I would like to thank Vickie Webster-Wood and John Galeotti and the members of the Biorobotics Lab like Nico, Evan, Xinyu, Abhi and the rest of the Tracir and MFI teams for supporting my research ideas, no matter how crazy they might have seemed at the time.

Finally, I would like to thank my Family: William, Christine, and Lydia, and my friends: Atom, Anja, Zoe, Sara, Andrew, Gordon, Sophie, and Noelle, for emotionally supporting me through my Graduate Degree. Without your proofreading of my work, acting as a sounding board for my ideas, and proving my food, I don't know if I would have finished.

Thank you all.

Abstract

The stiffness map of a deformable object stores information about that object's surface compliance. Thus, through a stiffness map, we gain insight into the physical properties of that object. Depending on the object, an understanding of stiffness has applications ranging from localizing tumors for surgery to grasping policies in manipulation. However, generating a stiffness map or stiffness mapping is challenging as even with information about the thickness of the underlying material, the object's physical geometry, and the composition of the material itself, it can be enormously computationally complex to model an object's surface compliance. Therefore it is often necessary to generate the stiffness map of an object by direct sampling through palpation or applying a force and measuring the subsequent displacement. Then by densely palpating over every point on an object's surface, a time-consuming process task for a very fine sampling, a stiffness map for an entire object can be generated. Alternatively, it is also possible to only palpate a subset of points from the surface. Then by applying regression, the sampled stiffness data can be used to estimate a function between points on a given surface (known as inputs or predictors) and the stiffness data. Finally, using this function, an object's entire stiffness map can be extrapolated.

Previous work on stiffness mapping [1, 2, 3] has specifically been interested in Gaussian Process Regression (GPR) due to its ability to estimate uncertainty about its predicted stiffness values. This uncertainty estimate helps direct the sampling of stiffness data, guiding where to palpate, and ideally leading to a more accurate estimate of an object's stiffness map with fewer data points. A key component of GPR is its kernel or covariance function which measures the similarity among our regression's predictors. For stiffness mapping, we assume that our predictor's covariance or points on the object's surface are correlated as a function of geodesic distance; points nearby on the surface of an object have similar stiffness values. However, GPR requires a smooth kernel function and thereby a smooth distance function between inputs. Consequently, stiffness mapping through GPR struggles with surfaces with a non-smooth geodesic distance function. Notably, this causes difficulty when the surface of our object is modeled using a discrete representation, for example a mesh, which is a common representation used in many fields, including robotics. Due to the mesh's discrete nature, the resulting geodesic distances between points are non-smooth, leading to an invalid kernel function and thus a poor fit of our data.

The main contribution of this thesis is a new method of GPR for fitting a function between predictors sampled from a surface and an associated surface-dependent scalar

distribution. The main focus of this work is using GPR in tandem with an embedding or mapping of our predictors into a higher dimensional Euclidean space. By mapping our predictors into a Euclidean space we can use the smooth, Euclidean distance metric to measure the similarities between data points, thereby ensuring a smooth kernel function. Specifically, this thesis presents a supervised learning approach for constructing embeddings where the embedded surface is constructed solely based on its observed stiffness data.

We evaluate our supervised method for embedding by testing its ability to build a map of an object's stiffness using a series of synthetic and real-world stiffness data sets. Then we compare these results against alternative techniques, such as unsupervised embedding methods, which have been the focus of research to this point. Unsupervised embedding methods construct an embedding as a function of some predetermined cost function instead of the observed data and is applied to the predictors as a preprocessing step before regression occurs. We find that our supervised embedding better predicts our underlying stiffness distributions over existing unsupervised embedding techniques. Although this work focuses on stiffness mapping, it can be applied to fit any data with an underlying function mapping between points sampled from a manifold to some scalar value. We hope to explore other distributions with a similar relationship in our future work, such as mapping an object's local thermal or friction information.

Contents

List of Tables

List of Algorithms

List of Figures

List of Symbols

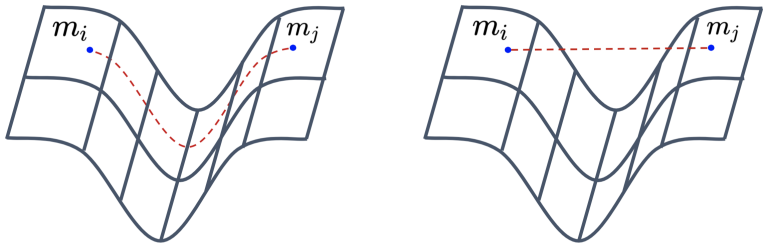
X	Set of independent variables
Y	Set of dependent variables
$\mathbb{D}$	Set of possible sampled data consisting of corresponding independent and dependent variables, $\mathbb{D} \equiv \{X, Y\}$
$\{x_i, y_i\}$	Data point from set of possible corresponding data points, $x_i, y_i \in \mathbb{D}$
$\mathbb{S}$	Observed or sampled data set consisting of a subset of corresponding independent and dependent variables, $\mathbb{S} \subset \{X, Y\}$
$\{\tilde{x}_i, \tilde{y}_i\}$	Corresponding data points from set of sampled data, $x_i, y_i \in \mathbb{S}$
GPR	Gaussian Process Regression
k	GPR kernel or covariance function relating two independent data points $k(X_i, X_j)$
f	function mapping between the independent variables to the dependent variables.
$\mathcal{M}$	A manifold, in the context of this work specifically a 2 dimensional manifold which is represented by a mesh and embedded into $\mathbb{R}^3$
m_i, m_j	Points on the surface of manifold $\mathcal{M}$ represented by vertices on a mesh, $m_i, m_j \in \mathcal{M}$.
$\mathcal{M}'$	Manifold $\mathcal{M}$ embedded in a manor not consistent with the original affine embedding of the original mesh. These embeddings are typically into $\mathbb{R}^N$.
m'_i, m'_j	Points on the new embedded manifold $\mathcal{M}'$, in a manner differs than the original affine embedding. $m'_i, m'_j \in \mathcal{M}'$
d_g	Geodesic distance between two points on the manifold i.e. $d_g(m_i, m_j)$
k_{ext}	Extrinsic GPR kernel or covariance function relating two independent data points sampled from an embedded manifold and is a function of the Euclidean distance $k_{ext}(m_i, m_j) \sim \|\|m_i - m_j\|\|$
k_{int}	Intrinsic GPR kernel or covariance function relating two independent data points sampled from an embedded manifold and is a function of the geodesic distance $k_{int}(m_i, m_j) \sim d_g(m_i, m_j)$
MDS	Multidimensional Scaling
Isomap	Isometric feature mapping
mGPR	manifold Gaussian Process regression
amGPR	appended manifold Gaussian Prcoess regression
K-nn	K nearest neighbors
NLML	Negative Log Marginal Likelihood

Glossary

Manifolds - a set of points ($\mathcal{M}$) that locally resembles Euclidean space. For example around every point on a two-dimensional manifolds the local neighborhood is indistinguishable from $\mathbb{R}^2$. Examples of two-dimensional manifolds (or surfaces) include the sphere, surface of a mustard bottles or the curved parabolic sheet seen in Fig. 1.

Geodesic Distance - the shortest distance between two points on a surface whose path is constrained to the surface. For example the shortest path between m_i and m_j in Fig. 1 has a geodesic distance of $d_g(m_i, m_j)$ and is best represented by the curved line in Fig. 1a.

Embedding - an injective continuous, structure preserving map between topological spaces. For example Fig. 1 shows a 2 dimensional manifold embedded within 3 dimensional space ($\mathbb{R}^3$).



(a) Geodesic Distance (b) Euclidean Distance

Figure 1: Difference between geodesic (a) and Euclidean (b) distances on a parabolic sheet, a 2 dimensional surface embedded into $\mathbb{R}^3$.

Isometric - having equal measurements i.e. isometric embedding refers to a distance-preserving map from one space to another.

Data set - A collection of related data points. For example a data set can consist of variables X and Y where the value $x_i \in X$ corresponds to $y_i \in Y$. Typically X represents the domain also known as predictors, features or independent variable and Y represents

the response variable data which are the dependent variables that vary as a function of X.

Regression - a statistical process for estimating the relationship between an independent variable and a dependent variable. An example of regression is linear regression, which fits a line between the independent and dependent variables.

Acquisition Functions - data collection technique that selects what features to sample in order to best improve the regression's model. For example, the most naive acquisition function is random sampling, which randomly acquired data points.

Supervised Learning - algorithms in machine learning for determining a functional relationship between a set of independent and dependent variables. Supervised learning methods primarily rely on a training data set to minimize a cost function that corresponds to the mapping between these two variables. Regression, such as linear regression, is an example supervised learning.

Unsupervised Learning - algorithms in machine learning used to interpret unlabeled data. These methods are often applied as a preprocessing step before supervised learning to uncover patterns in the unlabeled data resulting in a better fit of the underlying relationship between the independent and dependent variables. One typical example is Principal Component Analysis (PCA), which finds a lower-dimensional coordinate system which represents the data.

Chapter 1

Introduction

Stiffness maps, which relate points on the surface of an object to an associated value measuring that object's local compliance, give insight into the physical properties of deformable objects [5, 1, 2, 3]. For example, in medicine, the stiffness map of a soft organ can be used to localize tumors [1]. Furthermore, in the field of manipulation, stiffness maps have been used to inform grasping policies for picking up compliant objects [5]. However, constructing a stiffness map of a given object can be complicated as even with complete information about an object's material properties and geometry, it is often still computationally complex to generate a model describing how an object might deform [8].

One approach to generating a stiffness map or stiffness mapping of a real-world object is through sampling [5, 1, 2, 3]. Current methods for sampling stiffness focus on palpation or poking a discrete point on the surface of an object to measure the deflection vs. force, where stiffness is calculated as the proportionality constant between these two values [1]. Consequently, a stiffness map can be generated by densely palpating across every point on the surface of an object. However, densely palpating every point on an object can be quite time-consuming; therefore, a regression can use a subset of stiffness values to fit a function between the points on the object's surface (known as the inputs or the predictors) and the scalar stiffness values. Then, through extrapolation, this regression model can be used to estimate an understanding of an object's entire stiffness distribution, Fig. 1.1, without the need to palpate every point on an object's surface.

Recent work on generating stiffness maps (or stiffness mapping) has used Gaussian Process Regression (GPR) for two main reasons [9, 10, 11, 12, 8, 13]. (1) Its nonparametric approach to regression does not force a structure to the relationship between its predictors, points on the surface of each object, and their associated scalar stiffness values. Therefore we do not have to make prior assumptions about how stiffness varies over the surface of an object. (2) GPR estimates an uncertainty about its predicted stiffness values. This uncertainty estimate is helpful as a part of the palpation process as it can be used in acquisition functions: techniques that guide where on the surface of an object to palpate

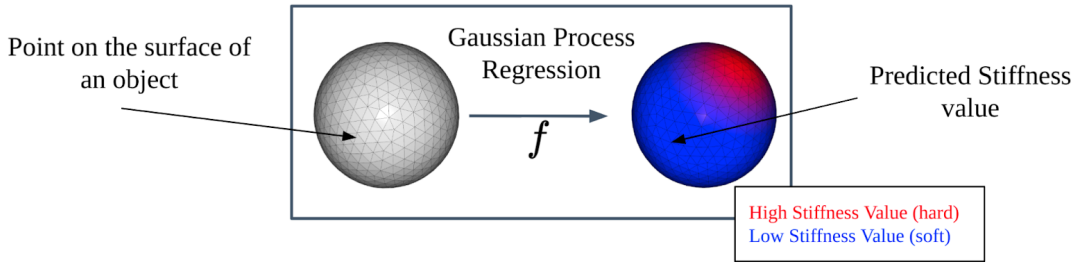


Figure 1.1: GPR used to fit a function f mapping from points sampled from a surface (predictors) to a scalar stiffness value.

and sample an object's stiffness. A carefully designed acquisition function can therefore reduce the number of samples or palpations necessary to get a reasonable estimate of an object's stiffness distribution [1].

One of the key components of GPR is its covariance or kernel function which estimates the similarity between its predictors, the points on the surface of an object [9, 12, 11]. For stiffness mapping, we assume that stiffness correlates with the geodesic distance function; points nearby on an object's surface have similar stiffness values. However, GPR inherently requires a smooth covariance or kernel function to estimate the similarity between its predictors. [12]. Consequently, it is difficult to map the stiffness of objects with a non-smooth geodesic distance [12, 11, 14]. For example, a mesh, which is a commonly used discrete representation of a surface, especially in computer graphics and robotics, Fig. 1.2b often results in a non-smooth geodesic distance functions. Therefore, if the surface of the object we are attempting to build the stiffness map for is modeled using a mesh, the kernel function will often be ill-defined and GPR will fit the data poorly [12].

To use GPR to map the stiffness of a three-dimensional object, modeled by a discrete representation, we must find a method of incorporating the predictors that results in a smooth distance function. This can be accomplished by first embedding or mapping the points sampled from a manifold into a higher dimensional Euclidean space [12] before fitting the GPR. Mapping these points into a Euclidean space allows the kernel function to leverage a smooth Euclidean metric to measure similarity instead of the non-smooth geodesic metric, creating a well-defined kernel. This thesis will explore different methods for constructing embeddings for discrete estimates of surfaces using a higher dimensional Euclidean space. Then it will test how well these embeddings can be used as a part of GPR by fitting a subset of data sampled from a known stiffness map and comparing how well the regression can extrapolate an object's stiffness distribution.

The most recent work on constructing an embedding of points sampled from manifolds is based on unsupervised learning [15, 16, 17, 18]. Unsupervised learning strategies create an embedding by finding a configuration of the predictors whose pairwise Eu-

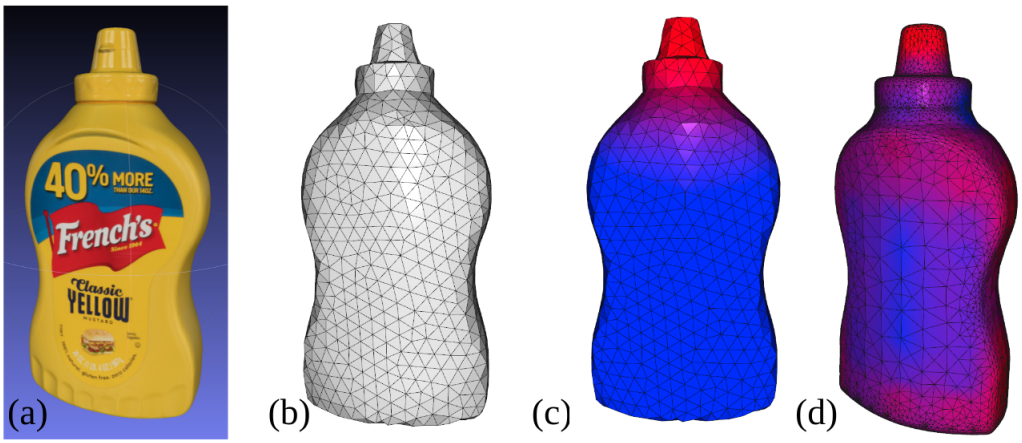


Figure 1.2: (a) the original continuous manifold of a mustard bottle, (b) the mesh representation of the object's surface sourced from the YCB data set [4], (c) a Gaussian projected onto the surface used as testing data and (d) real world stiffness data used for further validation [5].

clidean distance in the embedded space still respects the manifold's intrinsic structure. After this embedding is constructed, the smooth Euclidean distance metric can be used in the GPR kernel function to measure the similarities between the embedded predictors. In this thesis, we will evaluate two of the most popular unsupervised embedding strategies, Multidimensional Scaling (MDS) [16] and Isometric feature mapping (Isomap) [17].

An alternative method to embedding is through a supervised approach. A supervised embedding can be constructed using a parametric function to map the points from the manifold to the embedded space. Then by incorporating the parameters of this function into the regression, the parameters and therefore the embedding can be optimized to fit the measured stiffness values. Similar to the unsupervised embedding approach, this supervised approach to embedding means that a smooth Euclidean distance metric can be used in the GPR kernel function to measure the similarities between the embedded predictors. However, unlike the unsupervised approach to embedding, this supervised approach is a novel method. Therefore, **the main contribution of this thesis is the introduction of a supervised approach to embedding as a part of GPR: a new method to fitting data with predictors sampled from a discrete representation of a manifold**. To evaluate our new approach to embedding we will compare how well a supervised embedding in tandem with GPR is able to fit data against more traditional unsupervised approaches to embedding.

As this work is primarily motivated by stiffness mapping for applications in deformable object manipulation [5], we focus on testing our supervised embedding methods against unsupervised embedding methods using a manifold representation of particular interest, a mesh, like the one in Fig. 1.2b. Mesh representations have been constructed for many household objects, including the mustard bottle seen in Fig. 1.2 and the sphere

seen in Fig. 1.1, both sourced from the YCB data set [4]. The vertices of these meshes represent points on a manifold and will therefore act as our predictors. To test our supervised methods against alternative unsupervised methods for embedding, we generated a series of 42 synthetic data sets containing mesh vertices and a scalar, vertex-dependent attribute, stiffness, Fig. 1c. We also tested the models on a series of 5 real-world stiffness maps [5], Fig. 1d. These data sets contain mesh vertices and corresponding scalar stiffness values collected using a robotic gripper to palpate data from real deformable objects.

We evaluate our supervised methods for embedding against the unsupervised method for embedding through three simple tests. (1) For each of the 42 synthetic data sets, we randomly sample a subset of data and fit our GPR. Then we compare GPR's predicted stiffness values to the original data set. This test demonstrates the first reason GPR is useful for constructing stiffness maps, its nonparametric nature. (2) We further evaluate the model by comparing GPR in tandem with uncertainty sampling to our baseline random sampling methods. We find that using the uncertainty estimate to guide data sampling results in a more efficient stiffness mapping. This result demonstrates the second reason GPR is useful for constructing stiffness maps, a helpful estimate of uncertainty. (3) Finally, we demonstrate our models' real-world capabilities through testing on 5 data sets containing stiffness distributions sampled from real 3-dimensional objects [5], like the one seen in Fig. 1.2d. We find that methods for unsupervised embedding result in a distorted representation of the data, stretching the distance between neighboring points to reduce the global reconstruction error, thus occasionally resulting in a poor fit of the GPR. Therefore, across these three tests, we consistently find that our supervised methods for embedding result in a better fit of the underlying stiffness distributions over traditional unsupervised approaches to embedding.

Furthermore, the method presented in this thesis for supervised embedding as a part of GPR is not just constrained to stiffness mapping. Our models can be generalized to data with a similar relationship, predictors sampled from a manifold with an associated scalar attribute. In our future work, we hope to test our models on data with a similar relationship, such as mapping an object's local thermal or friction information, as these properties are helpful for more context-aware grasping [19]. Furthermore, beyond mapping the physical properties of 3D objects, this work can potentially generalize to predictors from higher dimensional manifolds, such as in medical image analysis where similar algorithms have aided in disease diagnosis [12] for potential future applications to surgical robotics [20].

Chapter 2

Background Information

The work presented in this thesis combines three significant concepts: Gaussian Process Regression (GPR), data with predictors sampled from a manifold, and stiffness maps. This amalgam provides the foundation for the following report. However, to best contextualize our findings, we will begin by outlining each idea independently.

2.1 Regression

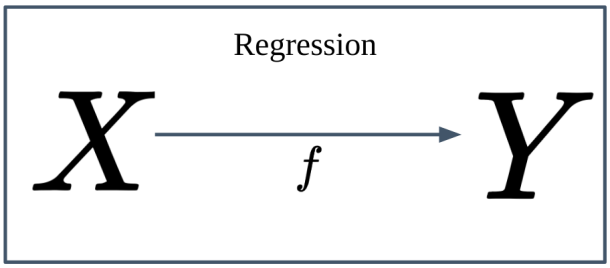


Figure 2.1: Model of classic regression, f is a map between the independent variables X and the dependent variable Y.

In classical statistical modeling, regression is a supervized learning method for deriving the functional relationship (f) between one or more independent variables (X) and a dependent variable (Y) [21], Fig. 2.1:

$$Y \approx f(X) \tag{2.1}$$

One of the simplest models for regression is a linear model:

$$y_i \approx \alpha x_i + \beta \tag{2.2}$$

where the value x_i is an independent data point ($x_i \in X$) and corresponds to the dependent data point $y_i \in Y$. Together, the set of all possible corresponding data points x_i and

y_i create $\mathbb{D} \equiv \{x_i, y_i\}$.

Linear regression fits the parameters α and β by minimizing a cost function. For example, one of the most common approaches, Ordinary Least Squares [21], minimizes the mean squared error (MSE) of a set of n training data points $\mathbb{S} \subset \mathbb{D}$.

$$\alpha, \beta = argmin_{\alpha,\beta} \sum_{i=1}^{n} (\tilde{y}_i - f_{\alpha,\beta}(\tilde{x}_i))^2/n = argmin_{\alpha,\beta} \sum_{i=1}^{n} (\tilde{y}_i - (\alpha\tilde{x}_i + \beta))^2/n \tag{2.3}$$

Where $\tilde{x}_i$ and $\tilde{y}_i$ are corresponding independent and dependent points in the set of training data points $\{\tilde{x}_i, \tilde{y}_i\} \in \mathbb{S}$. Due to the parameters α and β, linear regression is a form of parametric regression. This model sustains one major disadvantage, it assuming a linear relationship between X and Y.

On the other hand, nonparametric regression does not attempt to force a structure on the relationship between X and Y. One of the most basic nonparametric regression models is k-nearest neighbors (k-nn) [21]. K-nn uses the set of training data points ($\mathbb{S}$) to define the structure of its model. To estimate the value of y_j given a value for x_j, k-nn first searches through the set of training data points and finds the closest k dependent values to x_j, then averages their corresponding dependent values. For $k = 1$, k-nn is modeled as follows:

$$y_j \approx \tilde{y}_{\hat{i}} \tag{2.4}$$

$$\hat{i} = argmin_i(||\tilde{x}_i - x_j||) \tag{2.5}$$

Where $\hat{i}$ is the index of whichever value $\tilde{x}_i$ from the set of training data points, is closest to x_j. Therefore the value for y_i is estimated as, $\tilde{y}_{\hat{i}}$, which is the dependent term which corresponds to $\tilde{x}_{\hat{i}}$. Therefore, the relationship between the independent and dependent values is purely a function of our training data points.

This work will specifically focus on regression through the lens of GPR [9, 12, 10], a form of nonparametric regression.

2.1.1 Gaussian Process Regression

GPR is a generic supervised learning method with two main advantages [9]:

1. **The model is nonparametric**: it can fit a function without making assumptions about the structure of the data. In the context of stiffness mapping, GPR does not make assumptions about the shape of our stiffness distributions and how they relate to the object's surface from which they are sampled.

2. **Its prediction is probabilistic**: For every independent data point (x_i) GPR not only estimates a value (y_i), but also provides an uncertainty of its estimate (σ_i).

This uncertainty estimate (σ_i) allows for the development of sampling schemes that sample data which the model is most uncertain in its prediction.

A GPR model is fully defined by its mean (m) and covariance or kernel function (k) [9, 10]:

$$f \sim GP(m, k) \tag{2.6}$$

Where the values predicted by f is modeled by a Gaussian distribution ($\mathcal{N}$) with an expected value of $\mu(x_i)$ and a standard deviation or uncertainty of $\sigma(x_i)$:

$$p(f(x_i)|\mathbb{S}, x_i) = \mathcal{N}(\mu(x_i), \sigma^2(x_i)) \tag{2.7}$$

$\mu(x_i)$ therefore represents the most likely value for $f(x_i)$ with an uncertainty in its prediction bounded by $\sigma(x_i)$.

For this work we will use normalized data, therefore the mean of the data, and thereby the mean function m is 0 ($m \equiv 0$). This gives the following definitions for GPR's expected value $\mu(x_i)$ [9]:

$$\mu(x_i) = K_i^T K^{-1} \tilde{Y} \tag{2.8}$$

and its uncertainty in its prediction, $\sigma(x_i)$

$$\sigma(x_i) = K_{i,i} - K_i^T (K)^{-1} K_i \tag{2.9}$$

Where K is the kernel matrix such that $K_{i,j} = k(x_i, \tilde{x}_j)$ with K_i as the i^{th} column of K and $\tilde{Y}$ is a vector of the training dependent variables such that the i^{th} row of $\tilde{Y}$ is $\tilde{y}_i$ [9, 10].

The form of the kernel function k encodes high-level assumptions about the data being fit. For instance, a periodic kernel assumes GPR is attempting to fit data with a sinusoidal trend. The only constraint on the kernel function is that it must be semi-positive definite. This work will consider one of the most common kernel functions, the squared exponential function, as this choice of kernel assumes data whose dependent values are close in space must have similar independent values. The squared exponential kernel can be described in a general form as:

$$k(x_i, x_j) = \alpha exp(-\frac{d(x_i, x_j)^2}{2\beta^2}) \tag{2.10}$$

where d is a distance function between two independent data points x_i and x_j and α and β are hyperparameters that can be adjusted to effect the accuracy of the GPR. However the squared exponential kernel commonly uses the Euclidean distance metric as it is infinitely differentiable and thus smooth. This feature is extremely important, as a non-differentiable distance function can lead to a kernel which is not semi-positive definite

and thus an ill-defined GPR [11, 12]. This simplifies Eq. 2.10 as follows:

$$k(x_i, x_j) = \alpha exp(-\frac{d(x_i, x_j)^2}{2\beta^2}) = \alpha exp(-\frac{||x_i - x_j||^2}{2\beta^2}) \tag{2.11}$$

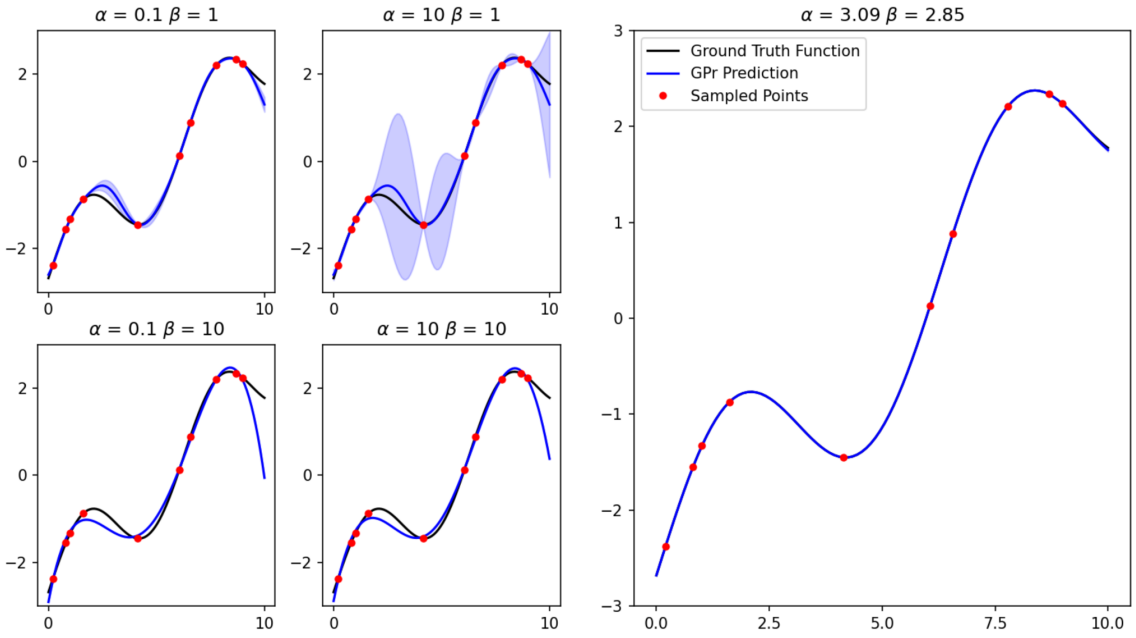


Figure 2.2: Four examples (left) of non-optimized hyperparameters α and β. By adjusting these parameters, we can see how α roughly corresponds to scale along the y axis and β corresponds to the length scale of the function. Using NLML, we can optimize the hyperparameters α and β (right) to create a reasonable fit for the underlying distribution.

To evaluate how GPR fits a distribution we will start by constructing a ground truth data. For this experiment our independent variable X consists of real numbers from 0 to 10 ($X \in (0, 10)$) and Y is an arbitrary, single variable function:

$$y_i = sin(x_i) + 0.5x_i \tag{2.12}$$

Together, these corresponding values for x_i and y_i create our ground truth data $\mathbb{D}$. This ground truth data set can be seen in Fig. 2.2 in black. To evaluate how changing the hyperparameters α and β will affect GPR we will randomly sample 10 values from $\mathbb{D}$. These 10 corresponding $\{\tilde{x}_i, \tilde{y}_i\}$ pairs are what we consider our training data set $\mathbb{S}$. By plugging these values into Eq. 2.8 and 2.9, along with various values for α and β, we can construct the plots in blue in Fig. 2.2, where the red points represent $\mathbb{S}$, the blue line represents GPR's prediction (Eq. 2.8) and the shading is one standard deviation from the mean (Eq. 2.9). Fig. 2.2 therefore demonstrates how changing these hyperparameters affects GPR's fit. In this context, α is a constant scaling value that tunes the GPR model's vertical scale, while β similarly tunes its horizontal scale.

A popular approach to setting the hyperparameters of the GPR kernel, α and β, is

through minimizing their Negative Log Marginal Likelihood (NLML).

$$NLML_k(\alpha, \beta) = -log(p(\tilde{Y}|\tilde{X}, \alpha, \beta)) = \frac{1}{2}\tilde{Y}^T K^{-1}\tilde{Y} + \frac{1}{2}log|K| \tag{2.13}$$

$$\alpha, \beta = argmin_{\alpha,\beta} NLML_k(\alpha, \beta) \tag{2.14}$$

This approach is often preferred over alternatives, such as a Monte Carlo method or separating the set of training data points into training and test data as it has a lower computational requirement [8]. Because NLML is a non-convex function it takes an optimization tool such as L-BFGS [22] with multiple random restarts of its initial parameters to yield an optimal value. Through this technique we can derive the right-most plot in Fig. 2.2, a minimum solution given 10 randomly sampled points. In conjunction with a simply defined kernel function (Eq. 2.10), GPR estimates the shape of an underlying ground truth function without assuming the underlying function's shape. This demonstrates the notable benefits of GPR's nonparametric nature.

2.1.2 Acquisition Functions

An acquisition function is a technique [1, 9] that guides data sampling. Random sampling, an example of one of the most naive acquisition functions, randomly selects a data point ($x_{\hat{i}}$) from a set of independent data X:

$$\hat{i} = random[x_i] \tag{2.15}$$

This method selected the data points in Fig. 2.2. Another, more informed acquisition function is uncertainty sampling, which samples the independent data point with the highest predicted uncertainty [9]. Because GPR outputs an uncertainty of its prediction (Eq. 2.9), we can use it in conjunction with uncertainty sampling for increased average efficiency.

$$\hat{i} = argmax_i[\sigma(x_i)] \tag{2.16}$$

To explore how these acquisition functions affect sampling, we utilize a simple experimental method with the known ground truth distribution ($\mathbb{D}$) introduced in Section 2.1.1. using the code outlined in Algorithm 1:

Algorithm 1: Steps to evaluate GPR

Input: D # D.X, D.Y are lists of all independent and dependent data points

Input: samplingScheme # randomSampling or uncertaintySampling

Input: maxIter # maximum number of sampled data points (10)

Input: K # kernel function

Result: MSE # list of MSE per number of sampled data points

begin

S.X, S.Y are lists of training independent and dependent data points ;

$S.X \longleftarrow \emptyset$;

$S.Y \longleftarrow \emptyset$;

$MSE \longleftarrow \emptyset$;

while $size(S.X) < maxIter$ **do**

Step 1: sample corresponding data points ;

if $samplingScheme == uncertaintySampling$ *and* $size(S.X) \geq 1$ **then**

$\hat{i} = argmax_i(\sigma(D.X, K, S, \alpha, \beta))$;

else

$\hat{i} = random(D.X)$;

Step 2: add points to observed data set ;

S.X.append($D.X[\hat{i}]$) ;

S.Y.append($D.Y[\hat{i}]$) ;

Step 3: minimize NLML (fit the GPR) ;

α, $\beta = argmin_{\alpha,\beta}(NLML_k(\alpha, \beta))$;

Step 4: calculate MSE ;

MSPE.append($\sum_i^{size(D.X)}((\mu(D.X[i], S, K, \alpha, \beta) - D.Y[i])^2)/size(D.X)$) ;

return MSE ;

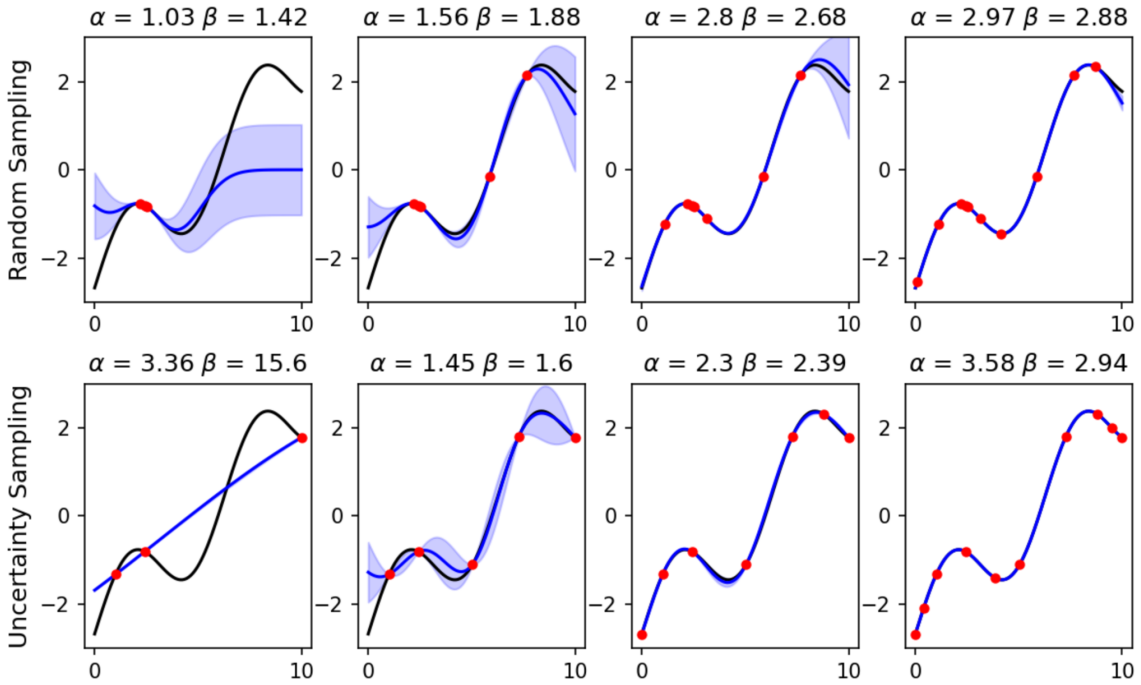


Figure 2.3: A comparison of two acquisition functions, uncertainty sampling and random sampling, and their regression estimates as the number of sampled data points increases. In this example, random sampling and uncertainty sampling have a similarly bad estimate for under 5 data points. However, as the number of samples increases, uncertainty sampling has an astoundingly close estimate of the underlying ground truth distribution

By sampling up to 10 data points and running this experiment once with random sampling and once with uncertainty sampling, we can create the plots in Fig. 2.3. As the number of sampled points increases, both random and uncertainty sampling ultimately yield an increasingly close model of the ground truth distribution. However, uncertainty sampling converges to the ground truth distribution significantly quicker. We can show these trends by plotting the mean squared prediction error (MSE) between GPR's predicted values ($\mu(x)$) and the testing data as a function of the number of points sampled (Fig. 2.4):

$$MSE = \sum_{i=1}^{n} (\mu(x) - y)^2 / n \tag{2.17}$$

Around three points, both random and uncertainty sampling yield comparable results. After five data points, however, uncertainty sampling has almost half the MSE. By the time we reach ten data points the effectiveness of these two methods are no longer remotely close, with uncertainty sampling out performing random sampling by several orders of magnitude. Therefore, we can conclude, the squared exponential kernel (Eq. 2.10) does an excellent job at not only estimating the underlying distribution but also understanding the uncertainty in its prediction. This simple experiment demonstrates the two essential components of GPR: its nonparametric nature and its probabilistic prediction.

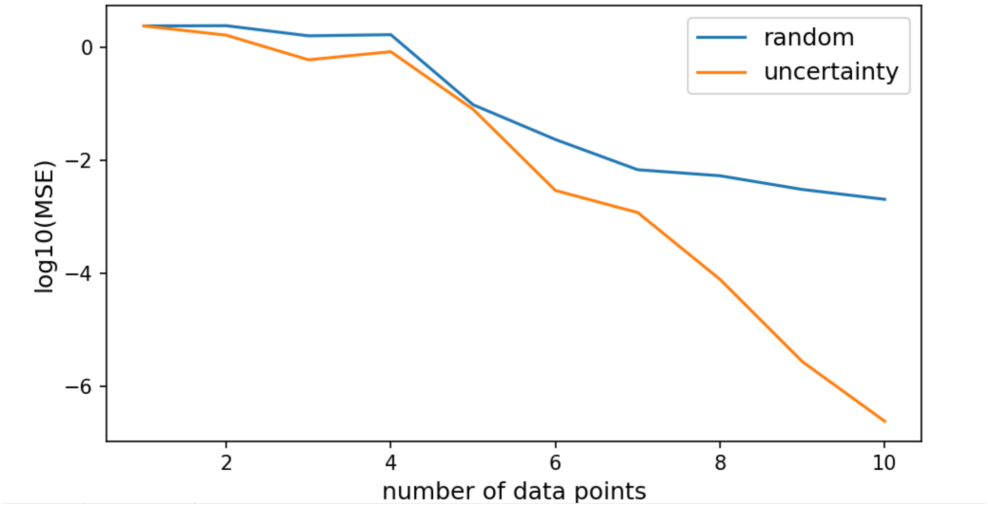


Figure 2.4: The MSE for the experiment seen in Fig. 2.3 plotted as a function of the number of data points sampled.

2.2 Meshes

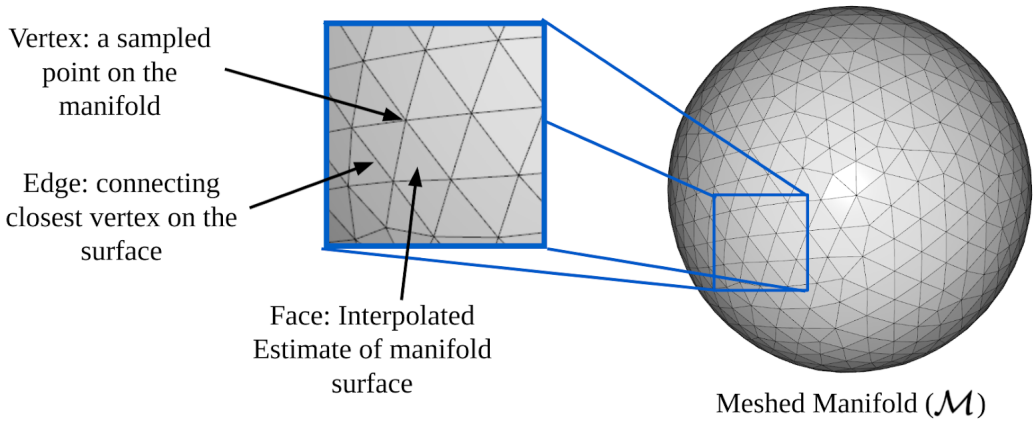


Figure 2.5: A meshed discrete manifold ($\mathcal{M}$) consists of 3 different elements. Vertices (m_i) represent directly sampled points from the underlying continuous manifold. Edges connect the closest vertices on the surface, and faces represent an interpolated manifold surface. The sphere mesh on the right is an example of discrete representation of a two-dimensional continuous manifold embedding in $\mathbb{R}^3$.

Manifolds are surfaces that contain a set of points that locally resemble Euclidean space. Some examples of manifolds are a sphere (the surface of a ball) as well as a more complex example, the surface of a mustard bottle (Fig. 1.2a). These are both examples of continuous two-dimensional manifolds with a natural embedding in three-dimensional space.

A discrete approximation of these surfaces can be generated by sampling a set of points from the continuous underlying manifold. The discrete representation of these

surfaces helps construct computer models of real objects whose surfaces cant be modeled by a simple equation, like the mustard bottle. For example, the meshes in the YCB [4] data set were constructed by sampling the surface of real 3D objects using depth sensors [23]. Combining many of these data points allows for a three-dimensional reconstruction of the object's surface using algorithms such as Poisson Reconstruction [23]. These types of algorithms output a variety of data structures, the most common of which is a mesh which is a discrete representation of a surface consisting of a series of vertices, edges, and faces, Fig. 2.5. Therefore, the generated mesh is discrete, representing a continuous two-dimensional surface with a natural embedding within three-dimensional Euclidean space.

This work focuses on the use of vertices from a meshes as independent data ($m_i \in \mathcal{M}$) to predict scalar-valued dependent data ($y_i \in \mathbb{R}$). This updates the regression model shown in Fig. 2.1 to Fig. 2.6 where the function f maps between $\mathcal{M}$ and Y:

$$y_i \approx f(m_i) \tag{2.18}$$

In Fig. 2.6 the blue and red shading visually depicts regions associated with low and high scalar y_i values, respectively. In stiffness mapping, this represents the soft (blue) and hard (red) regions.

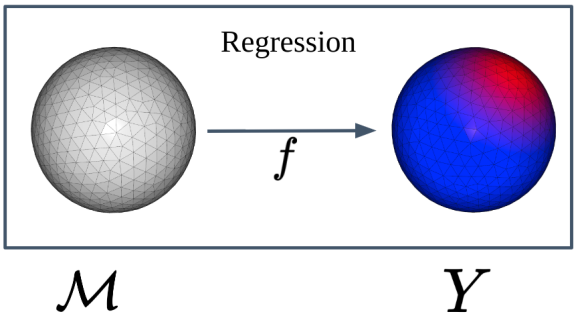


Figure 2.6: Regression for manifold-valued data, where the independent data is derived from vertices on a mesh ($m_i \in \mathcal{M}$) and scalar dependent data ($y_i \in \mathbb{R}$). The dependent data can be visualized through a red and blue color map.

2.2.1 Calculating Geodesic Distances

To find correlations between independent data points m_i and m_j sampled from $\mathcal{M}$, it is crucial to know how to calculate distances that respect the manifold's structure. The geodesic distance ($d_g(m_i, m_j)$) measures the shortest distance between two points on a manifold, whose path is constrained to the surface. For a sphere, we can calculate a closed-form geodesic distance function, where $\mathcal{M}$ is a unit sphere embedded into $\mathbb{R}^3$.

$$d_g(m_i, m_j) = 2arcsin(\frac{||m_i - m_j||}{2}) \tag{2.19}$$

It is important to note that this geodesic distance function is non-differentiable around $||m_i - m_j|| = 1$ or points that are on the exact opposite side of the sphere, such as the north and south pole. Intuitively two points on the opposite side of the sphere contain an infinite number of shortest paths, causing this geodesic function not to be smooth.

Unfortunately, it is not always possible to derive a closed-form solution to the geodesic distance function, d_g, especially for our discrete representation. One naive approach to calculating geodesic distances on a mesh is to use Dijkstra's algorithm, Fig. 2.7a. Classifying a mesh as purely vertices and edges, Dijkstra's algorithm starts at m_i, and uses a simple graph search to explore the closest "unvisited" vertices until it reaches its goal, m_j. On a mesh, this allows Dijkstra's algorithm to derive a naive geodesic distance between vertices. However, this algorithm ignores a vital element of the mesh, its faces. To reconcile this naive assumption, one can use continuous Dijkstra's algorithm [24], a more exact method for measuring geodesic distances on a mesh.

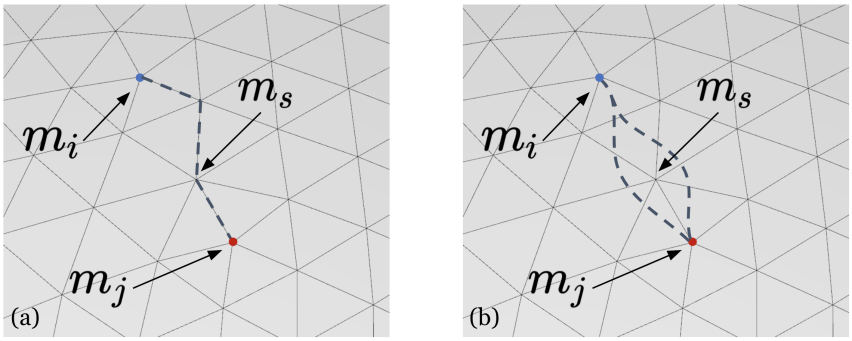


Figure 2.7: Geodesic paths between m_i and m_j calculated using discrete Dijkstra's (a) and continuous Dijkstra's algorithm (b). Due to the saddle point m_s, continuous Dijkstra's algorithm finds two equivalent paths.

Continuous Dijkstra's algorithm, Fig. 2.7b, calculates a geodesic distance by tracking groups of shortest paths [25] that expand out across the edges and faces from a source vertex. However when Continuous Dijkstra's encounters saddle points or vertices whose adjoining faces have a total angle greater than 2π (m_s), a wave is split into two waves representing paths of equivalent distances. These saddle points, while impossible for continuous 2-dimensional manifolds, are caused by the mesh's discrete nature approximating a continuous surface. Thus due to the discrete nature of a mesh, Continuous Dijkstra's often results in multiple solutions to the shortest geodesic path between two vertices. Furthermore, similar to how a sphere's geodesic distance function, Eq. 2.19, the geodesic distance function is not smooth when multiple paths exist between vertices so is this discrete geodesic distance function when it encounter's a saddle point. Therefore, the continuous Dijkstra's algorithm can calculate an exact geodesic distance measurement between all vertices on a mesh, however its result is not smooth. This is a basic explanation of the difficulties with taking geodesic distances between points sampled from a manifold, however for more information refer to [14, 26]

2.2.2 Constructing Testing Data

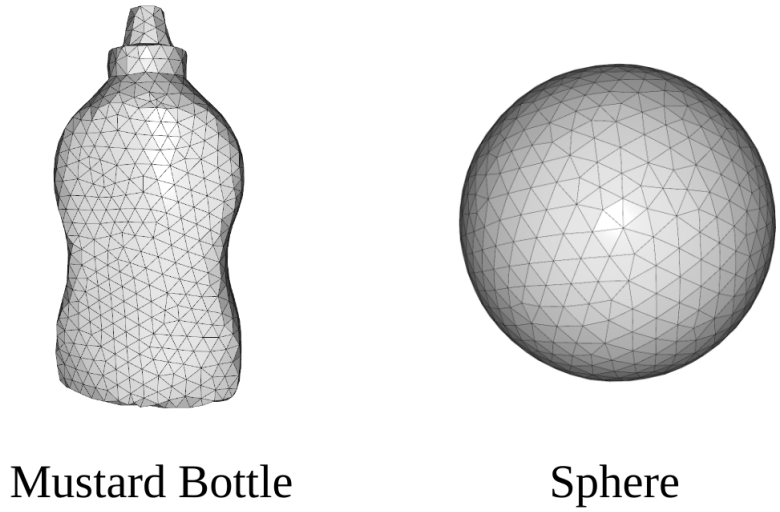


Figure 2.8: Two Meshes whose vertices represent independent data points.

To test different methods for handling regression with vertices from a mesh as independent data, we will initially focus on two different meshes: a mustard bottle and a sphere. These independent test data are shown in Fig. 2.8.

In order to construct these two sets of independent data, we relied on the YCB data set [4] to source our sphere and mustard bottle meshes. Then, to simplify our data, Meshlab [27] was used to reduce the number of data points to contain around 1,000 vertices (2,000 faces) and scaled each object to unit size.

To create our synthetic testing distributions for the dependent data (Y), we rely on the discrete geodesic distance between vertices calculated using the continuous Dijastra's algorithm [24, 25]. Following the steps in Fig. 2.9, we choose a vertex (m_{max}), project a three-dimensional Gaussian distribution onto the mesh centered around the selected vertex then sample the Gaussian distribution at each vertex.

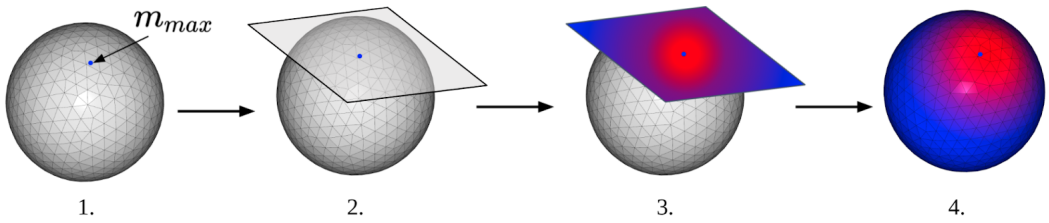


Figure 2.9: 1. Choose a point on the surface of the mesh (m_{max}) 2,3 Draw a Gaussian distribution above this point 4. Project Gaussian distribution onto the original surface

This is functionally equivalent to using the geodesic distances calculated using the continuous Dijkstra's algorithm [25, 24] as the distance metric in the Gaussian distribu-

tion:

$$y_i = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{1}{2}\frac{d_g(m_i, m_{max})^2}{\sigma^2}), \sigma = 0.2 \tag{2.20}$$

where m_{max} is the selected vertex and therefore also the maximum value of the Gaussian distribution.

By selecting two vertices for each sample mesh, we can create the four sample evaluation data sets seen in Fig. 2.10. These four distributions will be used as so-called 'ground truth' data sets ($\mathbb{D}$) when initially comparing the converging rates of different embedding models.

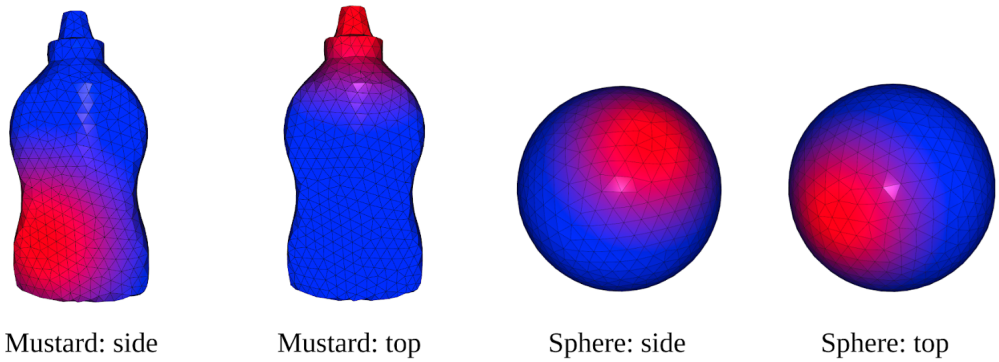


Figure 2.10: Four synthetic evaluation data set, showing a Gaussian distributions projected on to the surface of the mesh around different vertices.

Chapter 3

Related Work

Stiffness, a measurement of how easily an object resists deformation, can be measured through a process known as palpation, summarized in Fig. 3.1. First, given a deformable object, a probe can apply a force to the surface. Then measuring both the depth in which the probe deforms the surface and the amount of force applied, stiffness is calculated as the proportionality constant relating depth to force. Finally, by repeating this process over the deformable object's surface, a stiffness map is constructed. In the literature, there are two main methods for building these stiffness maps: simulation-based [28, 29, 6, 30, 31] and through robotic scanning, [5, 1, 7, 32, 2, 3, 33].

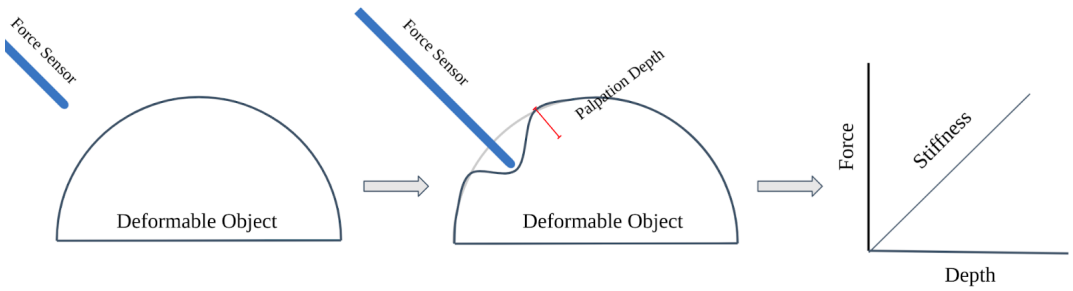


Figure 3.1: One example palpation: using a force sensor attached to a probe the stiffness is measured by a change in force vs palpation depth.

Simulation-based approaches to understanding deformable objects rely on a computational model describing the object and how it might respond to outside forces. For example, the work in Sin et. al. [6] outlines a series of popular finite element methods (FEM) for simulating deformable object interactions. The most basic model is the three dimensional Mass-Spring model, which characterizes an object as nodes connected by several springs. Then a simulated force can be applied, and the motion computed numerically for each node. Fig. 3.2 depicts a rectangular deformable object (a beam) and the resulting deformation as predicted by four comparable computational models.

In the case of stiffness mapping, FEMs could be employed to model palpation, simulating the probe's interaction with the object's surface. Therefore, constructing a

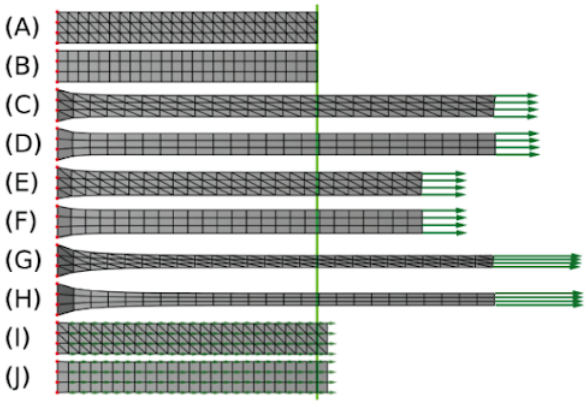


Figure 3.2: Simulated material behavior under stretching [6]. (A) (B) describe two cantilever beam at rest. (C) - (J) show the effects of pulling end of the beams as described by different simulators.

dense stiffness map is as simple as iteratively simulating forces over a set of points on the object's surface. However, in general, FEM and computational-based approaches require detailed knowledge of an object's material properties, such as material thickness, composition, geometry, etc [28, 29]. Therefore these methods struggle to make predictions about real-world objects. Furthermore, even with the same parameters different models, such as in Fig. 3.2, can also have varied results. Thus due to the complexities of model selection and parameter specification, using a computational approach to make predictions about specific real-world objects can be rather tricky.

An alternative approach to generating stiffness maps is through scanning or physically measuring the stiffness properties of deformable objects. For example, the work presented in Pai et al. [7, 32] describes a system for collecting data about the surface properties of objects, such as texture, reverberation, friction, and local stiffness. Fig. 3.3 shows how this data is collected: using a probe on the tip of a robotic arm, the probe can scan across the surface of an object collecting, in this image, information about an object's texture. However, their setup is designed to be general and is, therefore, identical to their setup for collecting information about stiffness.

However, this work creates a global map of stiffness, friction, etc., through a brute force approach. Pai et al. [7, 32] uses a robotic arm to scan across the entire surface of the object. This methodology can be time-consuming. Furthermore, this work does not consider how to interpolate between data points; instead, each region around the collected data is assumed to have the same stiffness value.

More recently, Zevallos et. al. [1] has also developed an approach to stiffness mapping. This work, which focuses on stiffness mapping for medical robotics, also generates a map through palpation, Fig. 3.4. However, instead of relying on a scanning or brute force-based approach, where every point is directly measured, this work employs GPR to fit the data and estimate a global stiffness distribution. Then using acquisition functions,

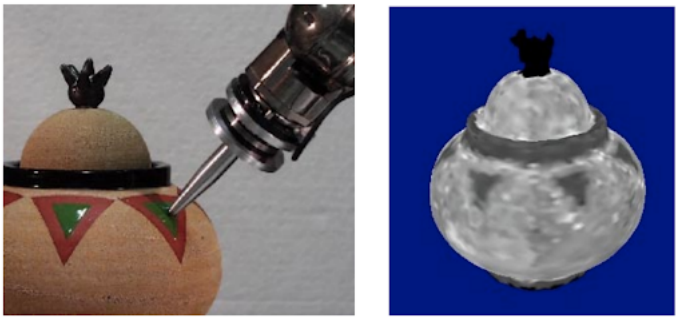


Figure 3.3: Robotic System described in Sin et. al. [7] designed to measure physical properties of objects for a better haptic models. In this example the robotic system (left) is rubbing the pot to determine local texture/friction which is described (right) as high (white) and low (grey) coefficients of friction.

such as uncertainty sampling, their methodology determines which location to palpate next. Zevallos et. al. [1] demonstrates that this approach to stiffness mapping reduces the number of sampled palpations necessary to locate tumors within organs. A similar methodology can be applied in general to map the stiffness of any three dimensional object, which is the primary goal of this thesis. However, Zevallos et. al. [1] assumes that the organ for which this work attempts to create a stiffness map is approximately flat or can easily be isometrically mapped to a two dimensional plane. However, this is impossible for many objects, ie a sphere [34].

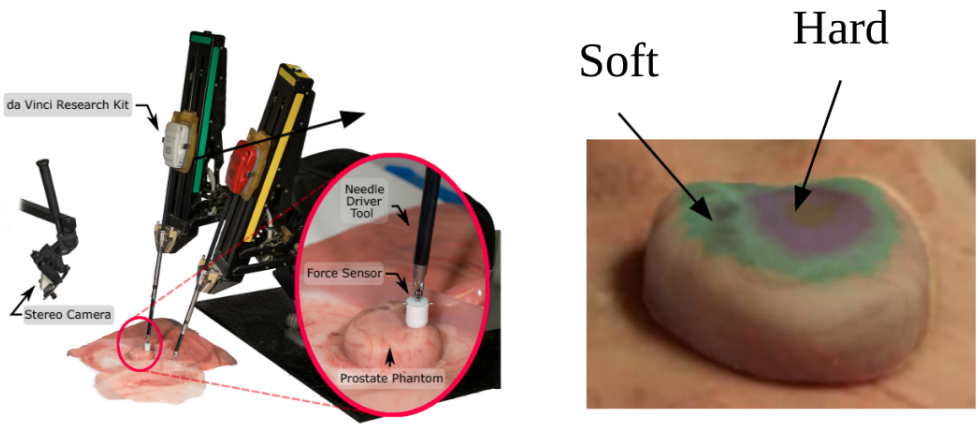


Figure 3.4: Robotic System described in Zevallos et. al. [1] designed to measure the stiffness of tumors of objects for tumor localization.

In this thesis, we will expand on the work presented in Zevallos et. al. [1] similarly using GPR as our regression model due to its connection to data acquisition methods, such as uncertainty sampling. In addition, however, we will improve their work through the use of supervised embedding. Supervised embedding algorithms, the primary contribution of this paper, allow us to map the stiffness of three dimensional objects more efficiently than both the work presented in Zevallos et. al. [1] and Pai et al. [7, 32].

Chapter 4

Intrinsic and Extrinsic Regression

This section will introduce two approaches for handling regression with independent data sampled from a manifold: intrinsic and extrinsic regression. The main difference between these two methods is the distance metric used to measure similarity between predictors. Extrinsic regression relies on a Euclidean distance between predictors embedded into a Euclidean space, while intrinsic regression relies on the geodesic distance. To compare these two methods, we first examine how an intrinsic K-nn and extrinsic K-nn algorithm might function in fitting the test distributions presented in Sec. 2.2.2. We use these results to create a baseline for general trends we should expect. We find that a K-nn algorithm, which incorporates the intrinsic, geodesic distance metric, fits our test distributions better. Next, we repeat the same experiments using an intrinsic and extrinsic distance metric as a part of GPR. However, the intrinsic GPR model performs far worse than the extrinsic GPR model due to discontinuities in the intrinsic distance metric. This experiment demonstrates the failure of GPR to generalize to non-smooth distance metrics. In future chapters, we will use the results from extrinsic GPR as a baseline comparison for our future work with supervised and unsupervised embedding methods.

4.1 Intrinsic and Extrinsic K-Nearest Neighbor Regression

K-nn, similar to GPR, is a supervised, non-parametric regression technique that relies on a distance function to measure similarities or covariance between data points. However, unlike GPR, K-nn does not require the distance function to be smooth; therefore, by experimenting with Intrinsic and Extrinsic K-nn we can uncover the generic trends we should expect from these two variations of the same method. To develop a comparable K-nn to GPR, we will start by expanding on the basic algorithm introduced in Sec. 2.1.: Instead of the teach and repeat style of learning presented in this algorithm K-nn regression, when predicting a given independent value, we can return a weighted average

of that using the K closest neighbors from a set of observed data points [21]. This more general version of K-nn regression can be described as follows:

Algorithm 2: K Nearest Neighbor Regression

Input: S # S.X, S.Y are lists of observed data points
Input: X # value for which K-NN is trying to make a prediction for
Input: dist # distance metric ie extrinsic of intrinsic
Input: K # K number of neighbors to consider
begin
$ClosestValues.X \longleftarrow \emptyset$;
$ClosestValues.Y \longleftarrow \emptyset$
# Step 1: Find the closest K number of neighbors from the observed data set ;
while $ClosestValues.size < K$ **do**
$i = argmin_i(dist(S.X[i], X))$;
$ClosestValues.X.append(S.X[i])$;
$ClosestValues.Y.append(S.Y[i])$;
$S.X = S.X.remove(i)$ $S.Y = S.Y.remove(i)$
Step 2: Return an average weighted by the distance to the original value X;
return $\sum_i^k ClosestValues.Y[i] *$
$dist(ClosestValues.X[i], X) / \sum_i^k dist(ClosestValues.X[i], X)$;

One crucial function to consider when applying K-nn to our data is the distance function or d. As described in Algorithm 2 this function affects both which points would be considered 'closest' and how they should be weighted. We can use two possible distance metrics to measure similarity between predictors. First, intrinsic K-nn uses the exact geodesic distance between data points as calculated using the Continuous Dijka's Algorithm outlined in Section 2.2.1. This distance metric considers the inherent geometry of the manifold from which the data points were sampled. Second, extrinsic K-nn uses the Euclidean distance between the data points. Because our independent data is already embedded in $\mathbb{R}^3$, this is a natural choice for measuring Euclidean distance.

To evaluate the two different K-nn models, intrinsic K-nn and extrinsic K-nn, we can repeat a process similar to Algorithm 1 outlined in Sec. 2.1. However, for these experiments, we will use data sampled from the surface of manifolds $\mathcal{M}$ to predict the ground truth distributions constructed in Fig. 2.10. To simplify our initial experiments, we will evaluate how these two models fit the ground truth distributions using solely random sampling averaging the resulting MSE (Eq. 2.17) over 20 trials. Furthermore, we will separate 60% of the data as training data set and 40% of the data to use as testing data. This will ensure that the K-nn regression does not overfit the observed data points. This experimental procedure can be outlined as:

Algorithm 3: Steps to evaluate intrinsic and extrinsic K-NN

Input: D # D.M, D.Y are lists of all independent and dependent data points

Result: MSE.train, MSE.test # list of MSE per number of sampled data points for the training and testing data sets

begin

Step 1: Divide the data set into testing and training data using a 60/40 split ;

$randomIndicies = randInts(0, size(D.M))$

Testing.M, Testing.Y are lists of segmented set of independent and dependent testing data points from which the observed data points are NOT sampled ;

$Test.M \longleftarrow D.M[randomIndicies[: 600]]$;

$Test.Y \longleftarrow D.Y[randomIndicies[: 600]]$;

Training.M, Training.Y are lists of segmented set of independent and dependent training data points from which the observed data points are sampled ;

$Train.M \longleftarrow D.M[randomIndicies[600 :]]$;

$Train.Y \longleftarrow D.Y[randomIndicies[600 :]]$;

S.M, S.Y are lists of observed independent and dependent data points ;

$S.M \longleftarrow \emptyset$;

$S.Y \longleftarrow \emptyset$;

MSE.train and MSE.test are the Errors for the training and testing data sets ;

$MSE.test \longleftarrow \emptyset$;

$MSE.train \longleftarrow \emptyset$;

while $size(S.M) < maxIter$ **do**

Step 2: sample corresponding data points ;

$\hat{i} = random(Train.M)$;

Step 3: add points to observed data set ;

S.M.append($Train.M[\hat{i}]$) ;

S.Y.append($Train.Y[\hat{i}]$) ;

Step 4: calculate MSEs use the KNN regression;

MSE.train.append($\sum_i^{size(Train.M)}((KNN(S, Train.M[i]) - Train.Y[i])^2)/size(Train.M)$) ;

MSE.test.append($\sum_i^{size(Test.M)}((KNN(S, Test.M[i]) - Test.Y[i])^2)/size(Test.M)$) ;

return MSE.train, MSE.test ;

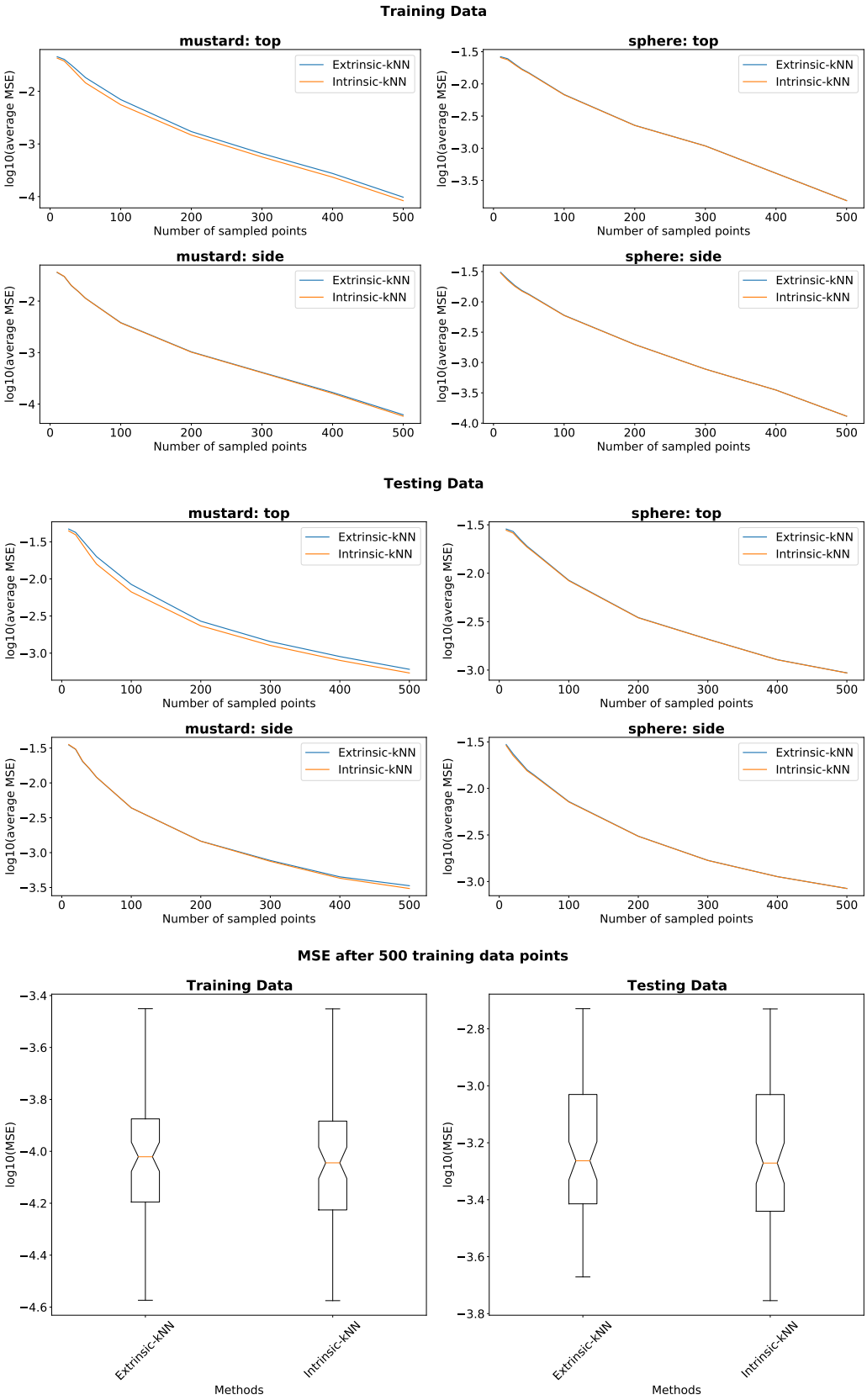


Figure 4.1: Averaged MSE of 4 sample distributions comparing extrinsic (with the original vertex embeddings) vs instrinic GPR.

Running this experimental procedure using the four test distributions (and a K=10) 10 times each, we get the following results, seen in Fig. 4.1. These plots show the general trends that we would expect from comparing our extrinsic and intrinsic methods using our constructed data sets. Since our original distributions were constructed using the intrinsic geodesic distance, the regression model which incorporates the same distance metric performs best. After 500 sampled data points, the intrinsic K-nn regression has a lower MSE for both the testing and training data sets across all four distributions. On average, the intrinsic method decreases our training error by 3.7 % and our testing error by 4.3 % as compared to our extrinsic method, with the most significant change coming from the distribution titled mustard:top.

4.2 Intrinsic and Extrinsic Gaussian Process Regression

Similar to the previous section, the intrinsic and extrinsic distance metric can be incorporated into GPR by altering the distance function it uses to measure similarity between predictors. In the case of intrinsic GPR this alters its kernel function from Eq. 2.11 to include the geodesic distance function [12, 11] :

$$k_{int}(m_i, m_j) = \alpha exp(-\frac{d_g(m_i, m_j)^2}{2\beta^2}) \tag{4.1}$$

With hyperpameters calculated using an identical method to Eq. 2.14 in Sec. 2.1.1:

$$\alpha, \beta = argmin_{\alpha,\beta} NLML_{k_{int}}(\alpha, \beta) \tag{4.2}$$

In the case of extrinsic GPR its kernel function is very similar to Eq. 2.10:

$$k_{ext}(m_i, m_j) = \alpha exp(-\frac{||m_i - m_j||^2}{2\beta^2}) \tag{4.3}$$

Again, the hyperpameters are calculated using an identical method to Eq. 2.14 in Sec. 2.1.1:

$$\alpha, \beta = argmin_{\alpha,\beta} NLML_{k_{ext}}(\alpha, \beta) \tag{4.4}$$

To evaluate the two different GPR models, intrinsic GPR and extrinsic GPR, we can repeat a process similar to Algorithm 2, the same method used to evaluate intrinsic and extrinsic K-nn. In this manner, we can compare these two approaches to regression with relative ease.

Algorithm 4: Steps to evaluate intrinsic and extrinsic GPR

Input: D # D.M, D.Y are lists of all independent and dependent data points

Input: samplingScheme # randomSampling or uncertaintySampling

Input: K # kernel function (extrinsic or intrinsic)

Result: MSE.train, MSE.test # list of MSE per for the training and testing data

begin

Step 1: Divide the data set into testing and training data using a 60/40 split ;

$randIndicies = randInts(0, size(D.M))$

Testing.M, Testing.Y are lists of segmented set of independent and dependent testing data points from which the observed data points are NOT sampled ;

$Test.M, Test.Y \longleftarrow D.M[randIndicies[:600]], D.Y[randIndicies[:600]]$;

Training.M, Training.Y are lists of segmented set of independent and dependent training data points from which the observed data points are sampled ;

$Train.M, Train.Y \longleftarrow D.M[randIndicies[600:]], D.Y[randIndicies[600:]]$;

S.M, S.Y are lists of observed independent and dependent data points ;

$S.M, S.Y \longleftarrow \emptyset, \emptyset$;

MSE.train and MSE.test are the Errors for the training and testing data;

$MSE.test, MSE.train \longleftarrow \emptyset, \emptyset$;

while $size(S.M) < maxIter(500)$ **do**

Step 2: sample corresponding data points ;

if $samplingScheme == uncertaintySampling$ *and* $size(S.M) \geq 1$ **then**

$\hat{i} = argmax_i(\sigma(Train.M, K, S, \alpha, \beta))$;

else

$\hat{i} = random(Train.M)$;

Step 3: add points to observed data set ;

S.M.append($Train.M[\hat{i}]$) ;

S.Y.append($Train.Y[\hat{i}]$) ;

Step 4: minimize NLML (fit the GPR) ;

$\alpha, \beta = argmin_{\alpha,\beta}(NLML_k(\alpha, \beta))$;

Step 5: calculate MSE ;

MSE.train.append($\sum_i^{size(Train.M)}((\mu(Train.M[i], S, K, \alpha, \beta) - Train.Y[i])^2)/size(Train.M)$) ;

MSE.test.append($\sum_i^{size(Test.M)}((\mu(Test.M[i], S, K, \alpha, \beta) - Test.Y[i])^2)/size(Test.M)$) ;

return MSE.train, MSE.test ;

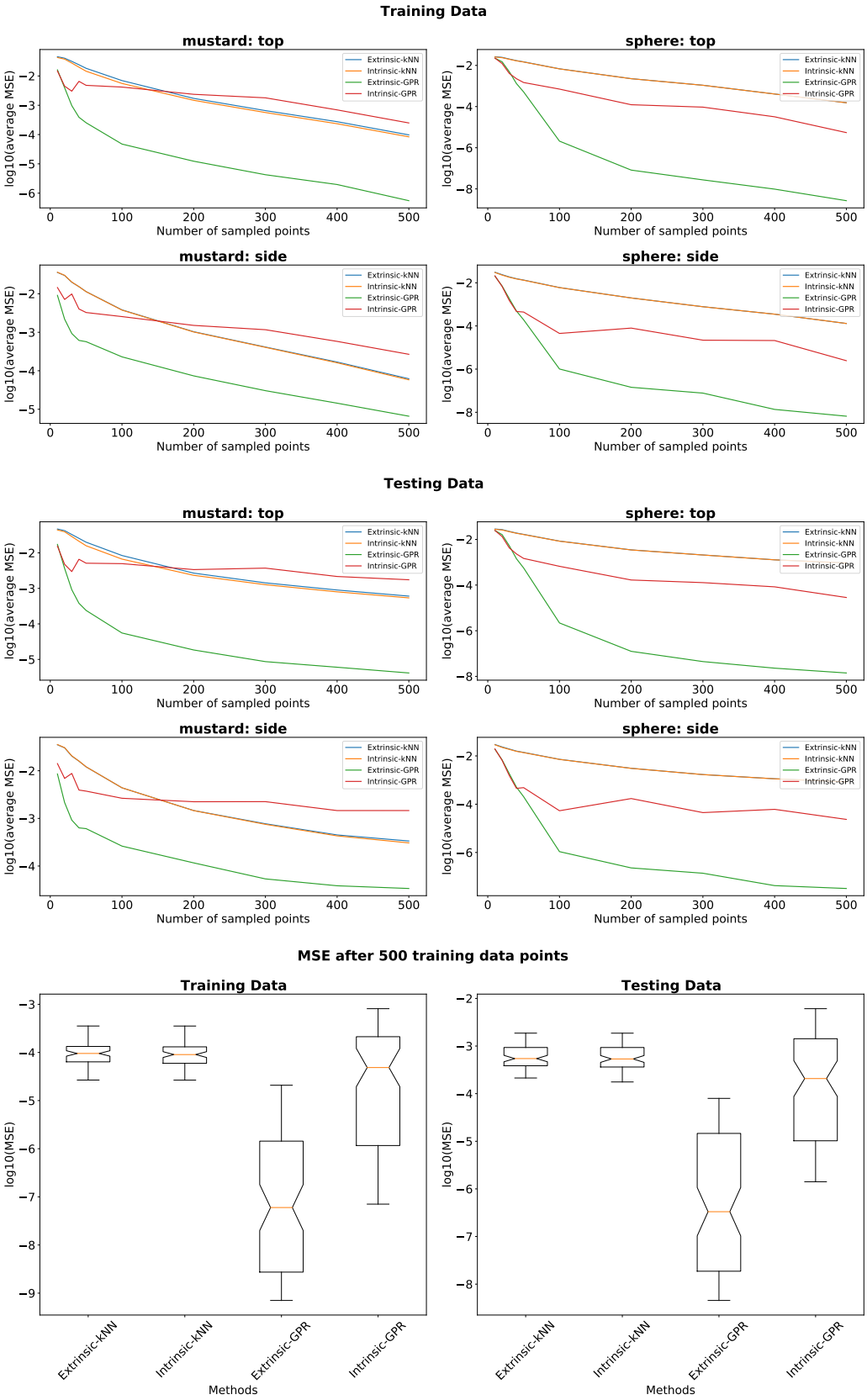


Figure 4.2: Averaged MSE of 4 sample distributions comparing extrinsic (with the original vertex embeddings) vs instrinic GPR.

As seen in Fig. 4.2, unlike the results for intrinsic and extrinsic K-nn, due to the ill-defined intrinsic kernel, intrinsic GPR performs significantly worse than extrinsic GPR. We are driven to this conclusion because intrinsic GPR increases the average training error by 84 % and the average testing error by 72 %. These results are in line with the results of Lin et. al. [12], which also demonstrates how fitting data with discontinuities in the intrinsic distance function leads to worse results. Therefore we can conclude that using an embedding of the data, even an embedding that ignores much of the manifold's structure, allows extrinsic GPR to outperform intrinsic GPR. This judgment leads us to this thesis's basic premise: is it possible to improve the embedding of the manifolds? We will explore this idea in the following chapters using two different methods, supervised and unsupervised learning. The unsupervised method attempts to improve the embedding of the manifold to reflect the intrinsic distances between data points. The supervised method attempts to improve the embedding as a function of the observed stiffness data. Both supervised and unsupervised methods for embedding then rely on extrinsic GPR to evaluate how well the embedded data can be used to predict an object's stiffness map, attempting to improve upon the results presented in this chapter.

Chapter 5

Unsupervised Embedding

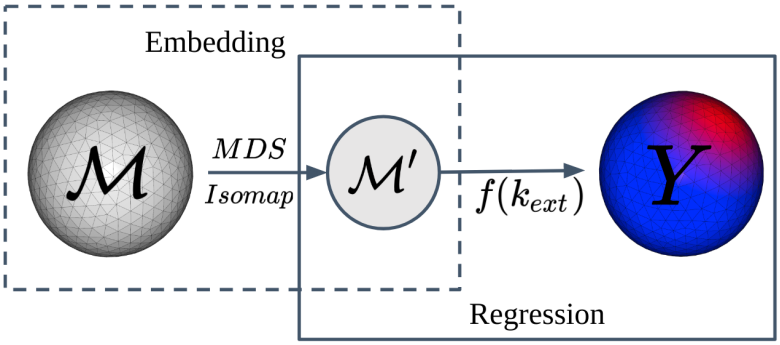


Figure 5.1: Embedding mesh regression, a form of extrinsic regression where the mesh is first embedded into some Euclidean space before applying the extrinsic GPR kernel.

In Sec. 4.2 we explored extrinsic GPR using the original mesh embedding $\mathcal{M}$ to represent the independent data. However, this embedding ignores much of the manifold's structure. This section will focus on constructing new embeddings which incorporate the intrinsic geometry of the manifolds by relying on unsupervised algorithms developed for manifold learning [15]. Two common unsupervised embedding methods are multidimensional scaling (MDS) [16] and isometric feature mapping (Isomap) [17]. Both of these methods attempt to find an embedding of the independent data that minimizes some reconstruction error; in our case, the difference between the Euclidean distance between any two embedded points and their original geodesic distance. This choice of reconstruction error attempts to find a representation of the data with a smooth distance metric while still representing the manifold's structure. After constructing the embedding, a new representation of the independent data ($\mathcal{M}'$) can then be incorporated into GPR using the extrinsic kernel outlined in the previous section. This creates a regression model seen in Fig. 5.1. To test the quality of these methods, we can repeat a similar experiment to the one outlined in Sec. 4.2. We find that both MDS and Isomap result in a worse fit of our test distributions. One possible explanation is that MDS and Isomap reduce the overall reconstruction error at the price or stretching near by points.

5.1 Multidimentional Scaling

Multidimensional Scaling (MDS) [16, 35] is an unsupervized embedding method that attempts to place points in a Euclidean space such that the distance between the data points reflect the manifold's structure [16, 35]. Modern, metric multidimensional scaling accomplishes this goal by minimizing what is called the stress or the reconstruction error:

$$\sum_{i,j=1}^{n} (d(m_i, m_j) - ||m'_i - m'_j||)^2 \tag{5.1}$$

Which is the sum squared error between the a user defined distance function and the euclidean distances of the embedding. Where $m_i, m_j \in \mathcal{M}$ are vertices on the original mesh and $m'_i, m'_j \in \mathcal{M}'$ are vertices from the newly embedded mesh. For this work we will focus on the preservation of the intrisic distance between points, therefore we will define our reconstruction error using the geodesic distance; the same values calculated using continuious Dijkstra's for the construction of the ground truth gaussian distributions in Eq. 2.20.

$$\sum_{i,j=1}^{n} (d_g(m_i, m_j) - ||m'_i - m'_j||)^2 \tag{5.2}$$

When applying MDS to the meshes from Fig. 2.8 we can embed the vertices into an arbitrary number of dimensions, $\mathcal{M}' \subset \mathbb{R}^N$. As MDS [16] is only able to quasi-isometrically embed the vertices we can plot the reconstruction error to initially quantify the results. In Tb. 5.1 we can see as the number of dimensions increase, there is generally a decrease in the reconstruction error, especially compared to the original embedding. However, for dimensions $N > 2$, there is no significant difference in the reconstruction error. Therefore, we will mainly focus on MDS for $N = 3$. This choice of the number of dimensions allows us to visualize the shape of this newly embedded mesh $\mathcal{M}'$, Fig. 5.2.

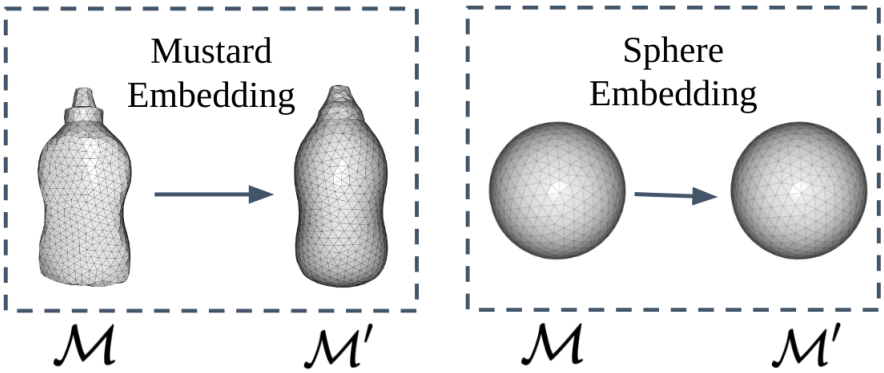
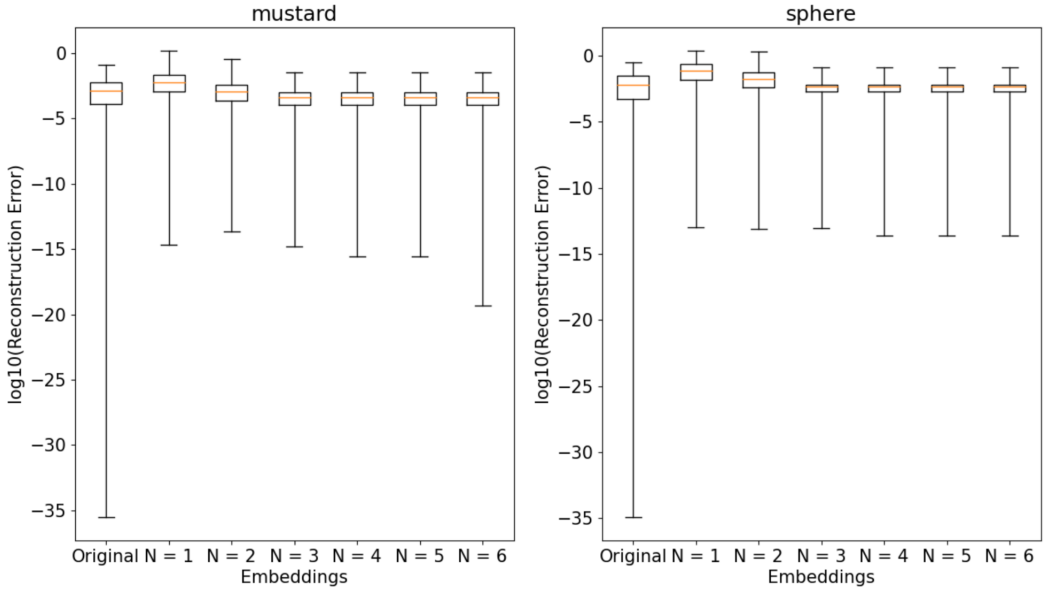


Figure 5.2: Embedded Meshes using MDS for N=3

An essential feature to note in Fig. 5.2 is how these new embedded manifolds lose a lot of the small detail as compared to the original mesh. For example, the mustard

Table 5.1: Reconstruction Error (Eq. 5.2) for MDS embedding into $\mathbb{R}^N$



Mesh	Original	N=1	N=2	N=3	N=4	N=5	N=6
mustard	4.62e-03	2.56e-02	4.56e-03	8.51e-04	8.51e-04	8.51e-04	8.51e-04
sphere	2.79e-02	1.83e-01	5.67e-02	7.07e-03	7.07e-03	7.07e-03	7.07e-03

bottle no longer displays small bumps on the side or top. One explanation is that MDS's cost function optimizes for points with a considerable geodesic distance at the detriment to relatively close points. Therefore MDS is primarily interested in reducing the global reconstruction error.

After the mesh is successfully embedded, incorporating this new representation of the data into the GPR is quite simple. By modify the extrinsic kernel (Eq. 4.3) our extrinsic kernel therefore becomes:

$$k_{ext}(m'_i, m'_j) = \alpha exp(-\frac{||m'_i - m'_j||^2}{2\beta^2}) \tag{5.3}$$

where $\mathcal{M}' = MDS(\mathcal{M}, N = 3)$. Because this method uses the same underlying extrinsic kernel function, optimizing α and β follows the same steps as Equation 3.3.

5.2 Isometric Feature Mapping

Isometric Feature Mapping (Isomap) is an method that attempts to construct a new embedding of points sampled from a manifold and place them in an N-dimensional Euclidean space with the goal of preserving the intrinsic geometry of the manifold [17, 35]. Isomap is therefore very similar to MDS, with one distinct difference: unlike MDS, Isomap

constructs its own rough estimate of geodesic distance. Isomap calculates its estimate of the geodesic distances by first constructing a discrete, graphical representation of the manifold and uses a graph search, such as Disktra's algorithm, to find the shortest distance between each vertex. Although this method of calculating the geodesic distances is inexact (as discussed in Sec. 2.2.1) it does allow room for an additional parameter that defines the number of nearest neighbors connecting each vertex in Isomap's graphical representation. This additional parameter determines how sensitive Isomap is to local or global distances. For example, after some experimentation, a value of nearest neighbors thought to be optimal is 800 vertices for our testing data sets. Thus, Isomap optimizes for a reconstruction error that attempts to preserve a local region around each vertices' 800 nearest neighbors. Through this additional parameter, Isomap might strike a balance between global and local optimization.

When applying Isomap to the meshes from Sec. 2.8 we embed the vertices into an arbitrary number of dimensions, $\mathcal{M}' \subset \mathbb{R}^N$. Like MDS, we can plot the reconstruction error to initially quantify the success of these embedding algorithms. As seen in Tb. 5.1 there are slightly different results for our two meshes. The sphere mesh has a minimum reconstruction error around $N = 3$, while the mustard bottle mesh has a continual improvement in its reconstruction error in higher dimensions $N > 3$. However as these improvements are rather negligible we will focus this work on Isomap in 3 dimensions as it allows us to visualize our results. It is also important to note that these reconstruction errors are slightly higher than the previously seen for MDS in Tb. 5.1. Considering Isomap is not optimizing for a global reconstruction error using an exact metric for measuring geodesic distances, this is not a surprise.

For N=3, the embedded meshes result in Fig. 5.3. An important feature to notice is how these new, embedded vertices look like a slightly scaled and distorted version of their original meshes ($\mathcal{M}$). Due to Isomap's cost function optimizing for points within a region of 800 vertices local regions of the mesh are preserved.

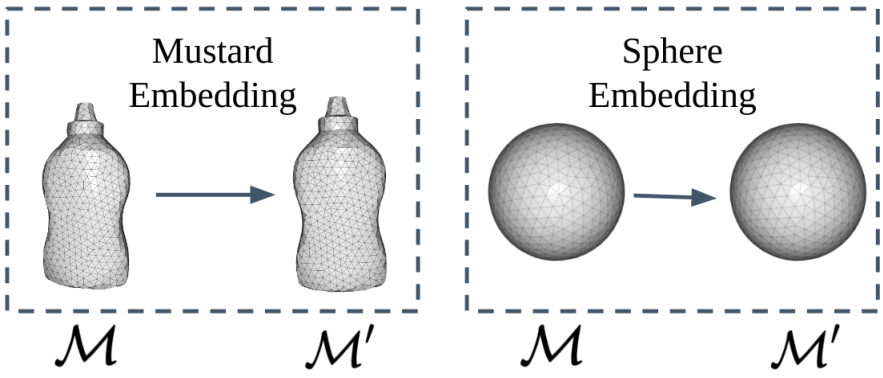
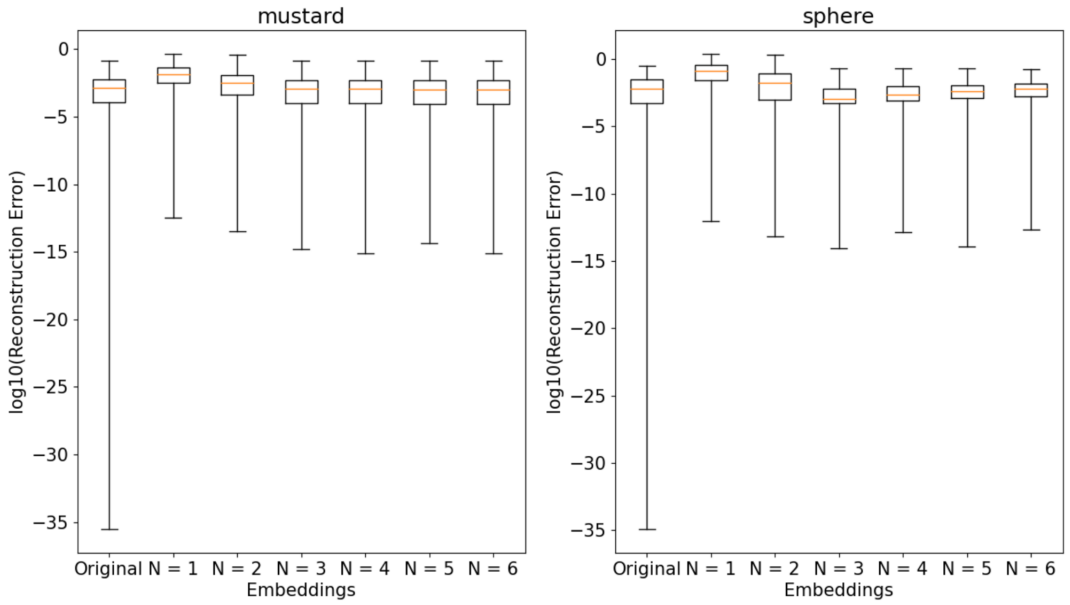


Figure 5.3: Embedded Meshes for Isomap with N=3

After the mesh is successfully embedded, incorporating this new representation of

Table 5.2: Reconstruction Error (Eq. 5.2) for Isomap embedding into $\mathbb{R}^N$



	Original	N=1	N=2	N=3	N=4	N=5	N=6
mustard	4.62e-03	3.41e-02	9.46e-03	4.34e-03	4.24e-03	4.14e-03	4.08e-03
sphere	2.79e-02	2.64e-01	9.02e-02	1.09e-02	1.10e-02	1.15e-02	1.24e-02

the data into the GPR follows the exact same procedure as for MDS. By modifying the extrinsic kernel, Eq. 4.3), to Eq. 5.3 where $\mathcal{M}' = Isomap(M, N = 3)$. Furthermore optimizing α and β follows the same steps as Equation 3.3.

5.3 Analysis

Using the embeddings produced by MDS and Isomap, we can repeat a process similar to the one in Alg. 2, as described in Alg. 5

Generally, we find that the new embeddings result in a worse MSE than our naive embedding. This conclusion is demonstrated by an increase in MSE between MDS and the naive embedding method of 16 % for the training data and 18 % for the testing data. Similarly, Isomap also sees an increase in MSE of 68 % for the training data and 71 % for the testing data. One potential explanation is that MDS and Isomap result in a worse representation of the data. Suppose we plot the reconstruction error for each vertex nearest neighbor, as seen in Tb. 5.3, the original embedding has a significantly lower local reconstructing error, especially for the sphere. Even though these methods decrease the global reconstruction error, this disparity between global and regional reconstruction could be one potential explanation why unsupervised embedding does not consistently

yield the best results. Therefore an alternative solution to embedding our data might come from a supervised approach instead.

	Original	MDS	Isomap
Mustard	3.24e-11	2.83e-05	3.30e-07
Sphere	1.62e-34	9.08e-05	2.80e-05

Table 5.3: Local reconstruction error for each vertices' nearest neighbors.

Algorithm 5: Steps to evaluate MDS and Isomap as apart of extrinsic GPR

Input: D # D.M, D.Y are lists of all independent and dependent data points
Input: samplingScheme # randomSampling or uncertaintySampling
Input: K # kernel function (extrinsic or intrinsic)
Result: MSE.train, MSE.test # list of MSE per for the training and testing data
begin
# Step 0: Preprcoess independent data using MDS or Isomap ;
D.M' = MDS(D.M)/Isomap(D.M)
Step 1: Divide the data set into testing and training data using a 60/40 split ;
$randIndicies = randInts(0, size(D.M))$
Testing.M, Testing.Y are lists of segmented set of independent and dependent testing data points from which the observed data points are NOT sampled ;
$Test.M, Test.Y \longleftarrow D.M'[randIndicies[: 600]], D.Y[randIndicies[: 600]]$;
Training.M, Training.Y are lists of segmented set of independent and dependent training data points from which the observed data points are sampled ;
$Train.M, Train.Y \longleftarrow D.M'[randIndicies[600 :]], D.Y[randIndicies[600 :]]$;
S.M, S.Y are lists of observed independent and dependent data points ;
$S.M, S.Y \longleftarrow \emptyset, \emptyset$;
MSE.train and MSE.test are the Errors for the training and testing data;
$MSE.test, MSE.train \longleftarrow \emptyset, \emptyset$;
while $size(S.M) < maxIter(500)$ **do**
# Step 2: sample corresponding data points ;
if $samplingScheme == uncertaintySampling$ *and* $size(S.M) \geq 1$ **then**
$\hat{i} = argmax_i(\sigma(Train.M, K, S, \alpha, \beta))$;
else
$\hat{i} = random(Train.M)$;
Step 3: add points to observed data set ;
S.M.append($Train.M[\hat{i}]$) ;
S.Y.append($Train.Y[\hat{i}]$) ;
Step 4: minimize NLML (fit the GPR) ;
$\alpha, \beta = argmin_{\alpha,\beta}(NLML_k(\alpha, \beta))$;
Step 5: calculate MSE ;
MSE.train.append($\sum_i^{size(Train.M)}((\mu(Train.M[i], S, K, \alpha, \beta) - Train.Y[i])^2)/size(Train.M)$) ;
MSE.test.append($\sum_i^{size(Test.M)}((\mu(Test.M[i], S, K, \alpha, \beta) - Test.Y[i])^2)/size(Test.M)$) ;
return MSE.train, MSE.test ;

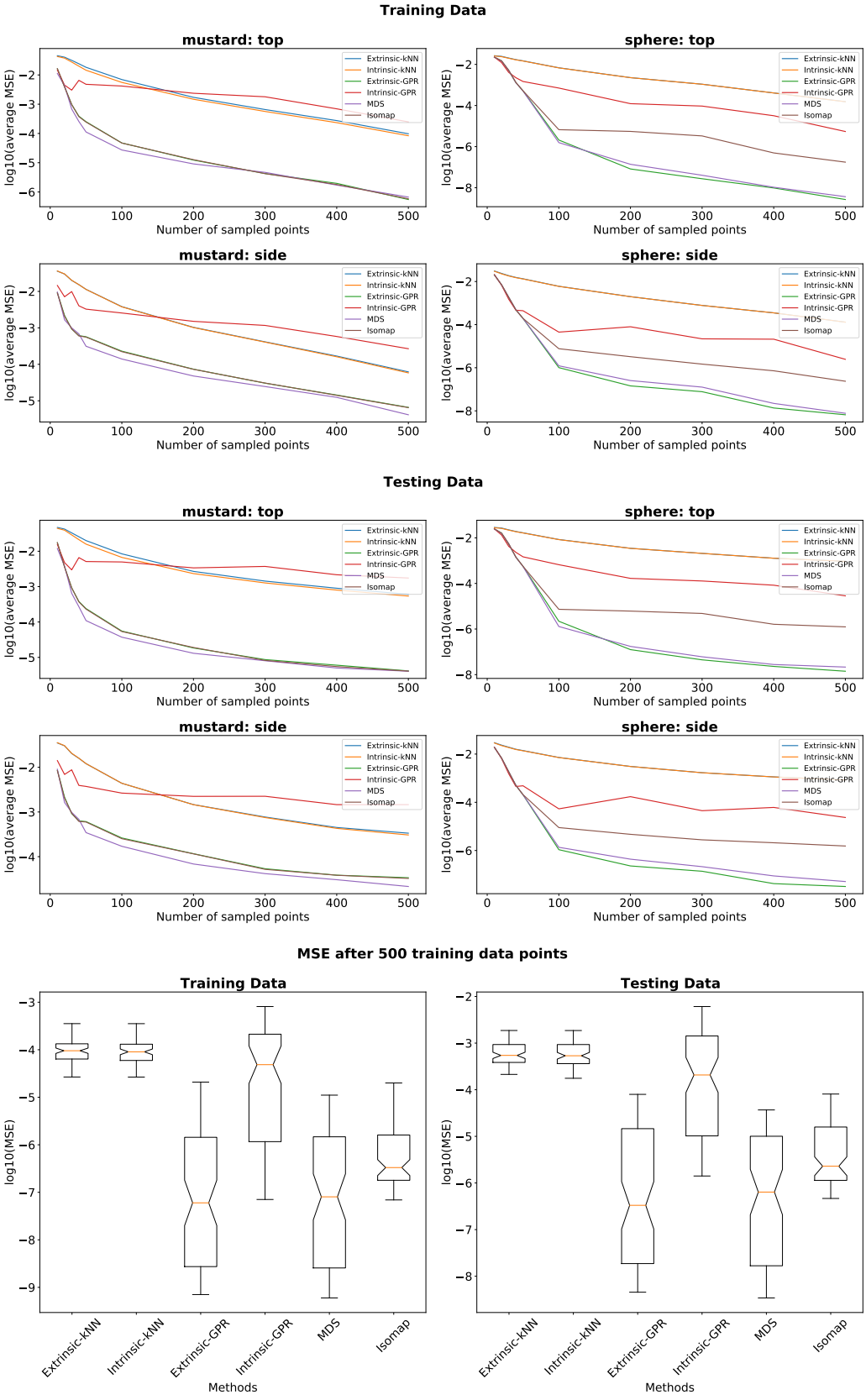


Figure 5.4: Averaged MSE of 4 sample distributions comparing extrinsic GPR using MDS and Isomap to prepossess the mesh vertices's embedding.

Chapter 6

Supervised Embedding

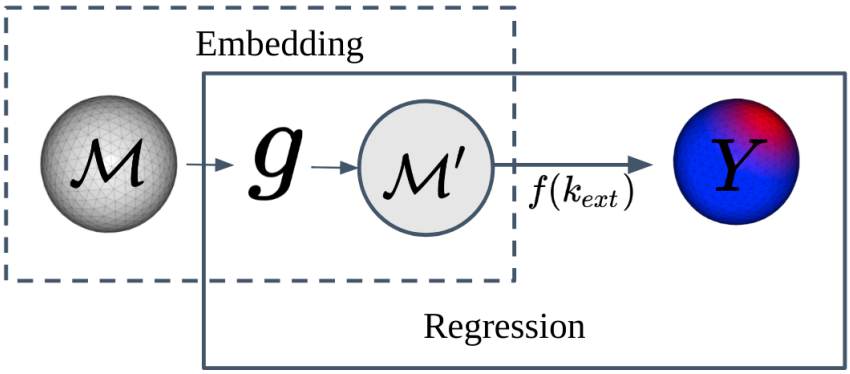


Figure 6.1: Supervised embedding relies on a parametric embedding function g to map from the points on the manifold ($\mathcal{M}$) to the new embedding ($\mathcal{M}'$). Then applying GPR, the parameters of g are optimized simultaneously with the extrinsic kernel's hyperparameters as a function of the observed data.

In this chapter, we will introduce the concept of supervised embedding. In contrast to MDS or Isomap, which learns a new representation of the inputs as a function of the dependent data, a supervised approach incorporates observed data into the embedding process. A key component to supervised embedding is a parametric embedding function g, which maps between $\mathcal{M}$ and $\mathcal{M}'$. This parametric embedding function can be incorporated into GPR by optimizing its parameters simultaneously with the hyperparameters α and β from our extrinsic kernel (Eq. 2.11), Fig. 6.1. Therefore the embedding of our mesh ($\mathcal{M}'$) is constructed as a function of the observed stiffness distribution. This supervised approach can take a few different shapes: manifold Gaussian Process regression (mGPR) [10] which entirely alters the independent data ($\mathcal{M}$) and appended manifold Gaussian Process regression (amGPR), which instead appends on a higher dimensional embedding to fine-tune the embedding. However, both of these methods have one thing in common- their goal is to alter the independent data's embedding ($\mathcal{M}'$) based on the dependent data (Y). Similar to the previous sections, we will test our methods by repeating similar experiments to Alg. 2.

6.1 Manifold Gaussian Process Regression

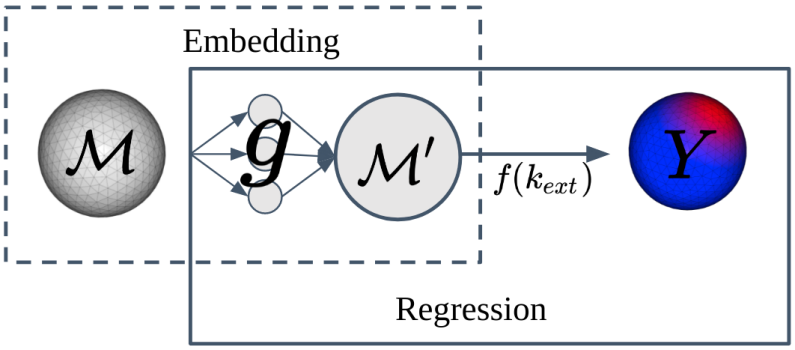


Figure 6.2: mGPR, a form of supervized embedding.

Manifold Gaussian Process Regression (mGPR) [10], is a supervised approach to embedding that relies on a parametric function to embed the data. Unlike MDS and Isomap, which construct this embedding by minimizing some pre-determined function based on just the predictors, mGPR optimizes its embedding as a function of the observed dependent data. By forming its embedding, $\mathcal{M}'$, as a function of the dependent data mGPR ideally results in a better fit of the underlying ground truth distributions.

In order to construct our supervised embedding, first we start by defining a function g, our embedding function, which maps between $\mathcal{M}$ and $\mathcal{M}'$:

$$m_i' = g(m_i) \tag{6.1}$$

The embedding function g must be a deterministic function that maps between $\mathbb{R}^3$ to an embedded space of $\mathbb{R}^N$. Therefore there are many different functions that can be considered, for example g may be considered the identity:

$$m_i' = g(m_i) = Im_i = m_i \tag{6.2}$$

Where I the identity matrix of $\mathbb{R}^{3\times3}$. However, this would result in an extrinsic GPR identical to Sec. 4.2.

Inspired by the work presented in Caranda et. al. [10] we will consider g instead as a single layer neural network with a hyper-tangent activation function:

$$g(m_i) = tanh(Wm_i + W_0) \tag{6.3}$$

Where the parameters of g are $W \in \mathbb{R}^{3x3}$ and $W_0 \in \mathbb{R}^{3x1}$ therefore, similar to MDS and Isomap, the resulting embedding is in 3-dimensions, $\mathcal{M}' \subset \mathbb{R}^3$. This choice of embedding function is also inspired by previous work in Antonova et. al. [36] where intuitively g can be thought of as a nonlinear function that warps the shape of the manifold. For example,

Fig. 6.3 shows what happens when applying the function g to our testing surfaces when $W = I$ and $W_0 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^T$.

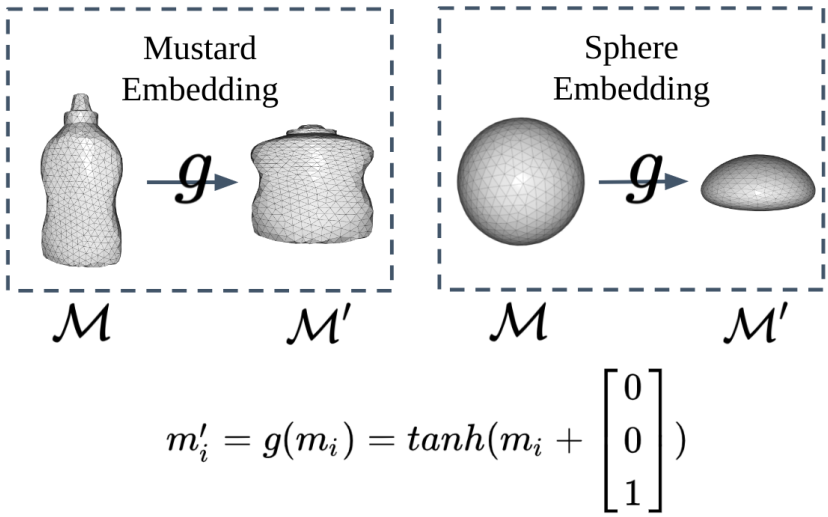


Figure 6.3: Example embeddings of mGPR using a simple equation for the embedding function g.

After constructing the parametric embedding function g it can then be incorporate in GPR by constructing a new kernel, k_{mGPR}, very similar to the original extrinsic kernel from Eq. 2.11:

$$\begin{aligned} k_{mGPR}(m_i, m_j) = k_{ext}(g(m_i), g(m_j)) = k_{ext}(m_i', m_j') \\ = \alpha \exp(-\frac{1}{2}\beta^{-2}||tanh(Wm_i + W_0) - tanh(Wm_j + W_0)||) \end{aligned} \tag{6.4}$$

By incorporating the embedding function g into the squared-exponential kernel, the parameters W and W_0 can now be consider as additional hyperparameters, similar to α and β. Therefore we can simultaneously optimized these values:

$$\alpha, \beta, W, W_0 = argmin_{\alpha,\beta,W,W_0}(NLML_{k_{mGPR}}(\alpha, \beta, W, W_0)) \tag{6.5}$$

Updating the optimization of the kernel's hyperparameters to incorporate the parameters of g will mean our new embedding $\mathcal{M}'$ will be a function of the observed data, $\mathbb{S}$. Thus, we have constructed the basis for supervised embedding as a part of GPR.

6.2 Appended Manifold Gaussian Process Regression

Appended manifold Gaussian Process regression (amGPR), Fig. 6.4, is a supervised approach to embedding very similar to mGPR; its embedding is a function of observed dependent data. However, amGPR varies in how it incorporates the embedding function g. While mGPR uses its embedding function to redefine its data completely, amGPR,

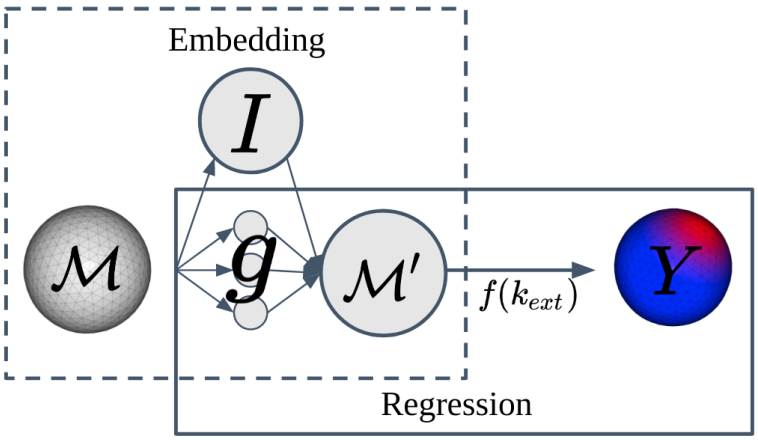


Figure 6.4: amGPR, a modification to the original mGPR.

instead appends the resulting values from g to its original embedding. The underlying assumption of amGPR is that the original embedding of the data, as shown in the previous sections, creates a pretty good fit of the stiffness distribution. Therefore, by appending the results of our embedding function to the original data, we can use amGPR to improve upon our original embedding.

amGPR, similar to mGPR, constructs its embedding through a function g; however instead of using the parametric function g to completely redefine the embedding, it is appended to the original data:

$$m_i' = \begin{bmatrix} m_i \\ g(m_i) \end{bmatrix} \tag{6.6}$$

where $\mathcal{M} \subset \mathbb{R}^N$, $g(m_i) \in \mathbb{R}^M$ and the resulting embedding $\mathcal{M}' \subset \mathbb{R}^{N+M}$. Similar to mGPR we will consider a single layer neural network (Eq. 6.3) with a hyper-tangent activation function as our parametric function g. This will allow us to make comparisons between two models that contain the same underlying embedding function, with the same number of hyper parameters. Furthermore, this embedding can be be incorporate in GPR by constructing a new kernel, k_{amGPR} very similar to the original extrinsic kernel from Eq. 2.11:

$$\begin{aligned} k_{amGPR}(m_i, m_j) = k_{ext}(\begin{bmatrix} m_i \\ g(m_i) \end{bmatrix}, \begin{bmatrix} m_j \\ g(m_j) \end{bmatrix}) = k_{ext}(m_i', m_j') \\ = \alpha \exp(-\frac{1}{2}\beta^{-2}|| \begin{bmatrix} m_i \\ tanh(Wm_i + W_0) \end{bmatrix} - \begin{bmatrix} m_j \\ tanh(Wm_j + W_0) \end{bmatrix} ||) \end{aligned} \tag{6.7}$$

And whose hyperparamter's can be optimized as:

$$\alpha, \beta, W, W_0 = argmin_{\alpha,\beta,W,W_0}(NLML_{k_{mGPR}}(\alpha, \beta, W, W_0)) \tag{6.8}$$

Through amGPR, the resulting embedding $\mathcal{M}'$ has the following property when

setting the parameters of the neural network to 0:

$$||m'_i - m'_j||^2 = || \begin{bmatrix} m_i \\ g(m_i, W = 0, W_0 = 0) \end{bmatrix} - \begin{bmatrix} m_j \\ g(m_j, W = 0, W_0 = 0) \end{bmatrix} ||^2 = ||m_i - m_j||^2 \tag{6.9}$$

The pairwise Euclidean distance between points sampled from $\mathcal{M}$ and $\mathcal{M}'$ are identical. Intuitively this construction allows amGPR to tweak the embedding instead of altogether redefining it, as in the case of mGPR. Hopefully, this will enable amGPR to be comparable, if not better than, the original embedding $\mathcal{M}$.

6.3 Analysis

Similar to our previous experiments, we will compare these models by repeating a process identical to the one in Alg. 2:

Algorithm 6: Steps to evaluate mGPR and amGPR

Input: D # D.M, D.Y are lists of all independent and dependent data points

Input: samplingScheme # randomSampling or uncertaintySampling

Input: K # kernel function (extrinsic or intrinsic)

Result: MSE.train, MSE.test # list of MSE per for the training and testing data

begin

Step 1: Divide the data set into testing and training data using a 60/40 split ;

$randIndicies = randInts(0, size(D.M))$

Testing.M, Testing.Y are lists of segmented set of independent and dependent testing data points from which the observed data points are NOT sampled ;

$Test.M, Test.Y \longleftarrow D.M[randIndicies[:600]], D.Y[randIndicies[:600]]$;

Training.M, Training.Y are lists of segmented set of independent and dependent training data points from which the observed data points are sampled ;

$Train.M, Train.Y \longleftarrow D.M[randIndicies[600:]], D.Y[randIndicies[600:]]$;

S.M, S.Y are lists of observed independent and dependent data points ;

$S.M, S.Y \longleftarrow \emptyset, \emptyset$;

MSE.train and MSE.test are the Errors for the training and testing data;

$MSE.test, MSE.train \longleftarrow \emptyset, \emptyset$;

while $size(S.M) < maxIter(500)$ **do**

Step 2: sample corresponding data points ;

if $samplingScheme == uncertaintySampling$ *and* $size(S.M) \geq 1$ **then**

$\hat{i} = argmax_i(\sigma(Train.M, K, S, \alpha, \beta))$;

else

$\hat{i} = random(Train.M)$;

Step 3: add points to observed data set ;

S.M.append($Train.M[\hat{i}]$) ;

S.Y.append($Train.Y[\hat{i}]$) ;

Step 4: minimize NLML (fit the GPR) ;

α, β, W, $W_0 = argmin_{\alpha,\beta,W,W_0}(NLML_k(\alpha, \beta, W, W_0))$;

Step 5: calculate MSE ;

MSE.train.append($\sum_i^{size(Train.M)}((\mu(Train.M[i], S, K, \alpha, \beta) - Train.Y[i])^2)/size(Train.M)$) ;

MSE.test.append($\sum_i^{size(Test.M)}((\mu(Test.M[i], S, K, \alpha, \beta) - Test.Y[i])^2)/size(Test.M)$) ;

return MSE.train, MSE.test ;

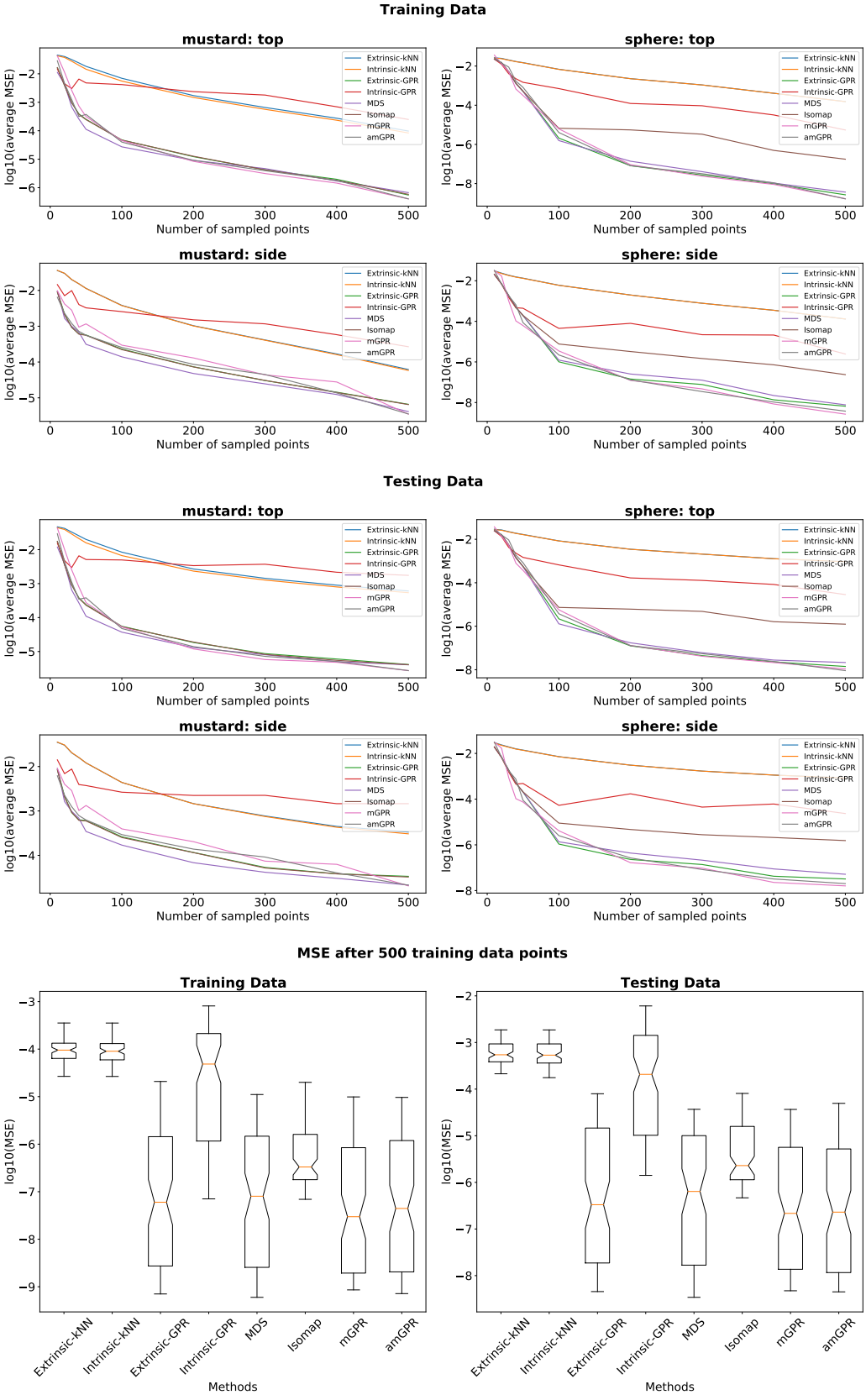


Figure 6.5: Averaged MSE of 4 sample distributions comparing mGPR and amGPR.

Fig. 6.5 summarizes these results. Generally, both supervised learning methods have significantly lower averaged MSE than the other methods, especially after sampling many data points. This conclusion is demonstrated by a decrease in MSE between mGPR and the naive embedding method of 17 % for the training data and 20 % for the testing data. Similarly, amGPR also sees an increase in MSE of 4 % for the training data and 7 % for the testing data. Between all methods introduced in this thesis, our supervised approaches to embedded results in the lowest MSE. Therefore, we can conclude supervised approaches to embedding are the best method for fitting our synthetic stiffness distributions. However, to generalize these claims, we will need to evaluate these algorithms further.

Chapter 7

Results

To fully compare supervised and unsupervised embedding methods, which are summarized in Tb. 7.1, we evaluate these models with three additional experiments. (1) To assess the generalizability of our results, we expanded the number of ground truth data sets used to test our models. Instead of the two objects used to initially gauge the different algorithms, we increased the number of meshes to 7, each with 6 different dependent scalar distributions. In total, this constructs 42 different test distributions, which can be seen in Appendix A. (2) We construct a basic active learning strategy that incorporates uncertainty sampling (Eq. 2.16). This further validates not only these models' mean (Eq. 2.8) is well defined, but also their uncertainty (Eq. 2.9). (3) We test these models on 5 stiffness maps sampled from real world object [5] found in Appendix B. Unlike our previous experiments, this represents testing on real-world data and demonstrates potential applications beyond our simple evaluation data sets.

7.1 More Test Data

To further test the generalizability of our initial findings, we first test our models on an additional set of synthetic distributions. Following a similar methodology to Alg. 2 for this experiment we sourced a total of ten meshes from the YCB data set [4] and simplified each object to around 1000 vertices, or 1000 independent data points. Then, to construct the dependent data, we continued to use a Gaussian distribution projected onto each one of these surfaces. Similar to Sec. 2.2.2, by selecting four vertices for each mesh, we can construct a total of 42 test data sets. Images of all testing distributions are located in Appendix A.

Similar to our previous experiments, we will evaluate the models for supervised and unsupervised embedding for each object's synthetic distribution by splitting the data randomly with 60 % into a training set and 40 % into a testing data set. Then by randomly sampling up to 500 data points from the training data set of each object we can compare how well each model extrapolates their original distribution. However, unlike

			Kernel Function: $k(m_i, m_j)$	No. params	Embedded Dimensions	Uncertainty Sampling
Intrinsic		K-nn	NA	1	3	✗
		GPR	$\alpha\ \exp(-\frac{1}{2}\beta^{-2}d_g(m_i, m_j))$ $\alpha, \beta = argmin_{\alpha,\beta}(NLML_{k_{ext}}(\alpha, \beta))$	2	3	✓
Extrinsic		K-nn	NA	1	3	✗
		GPR	$\alpha\ \exp(-\frac{1}{2}\beta^{-2}\|\|m_i - m_j\|\|)$ $\alpha, \beta = argmin_{\alpha,\beta}(NLML_{k_{ext}}(\alpha, \beta))$	2	3	✓
	Unsupervized	MDS	$\alpha\ \exp(-\frac{1}{2}\beta^{-2}\|\|m'_i - m'_j\|\|)\ , \mathcal{M}' = MDS(\mathcal{M}, N)$ $\alpha, \beta = argmin_{\alpha,\beta}(NLML_{k_{ext}}(\alpha, \beta))$	2	N=3	✓
		Isomap	$\alpha\ \exp(-\frac{1}{2}\beta^{-2}\|\|m'_i - m'_j\|\|)\ , \mathcal{M}' = Isomap(\mathcal{M}, N)$ $\alpha, \beta = argmin_{\alpha,\beta}(NLML_{k_{ext}}(\alpha, \beta))$	2	N=3	✓
	Supervized	mGPR	$\alpha\ \exp(-\frac{1}{2}\beta^{-2}\|\|tanh(Wm_i + W_0) - tanh(Wm_j + W_0)\|\|)$ $\alpha, \beta, W, W_0 = argmin_{\alpha,\beta,W,W_0}(NLML_{k_{mGPR}}(\alpha, \beta, W, W_0))$	14	N = 3	✓
		amGPR	$\alpha\ \exp(-\frac{1}{2}\beta^{-2}\|\|\begin{bmatrix} m_i \\ tanh(Wm_i + W_0) \end{bmatrix} - \begin{bmatrix} m_j \\ tanh(Wm_j + W_0) \end{bmatrix}\|\|)$ $\alpha, \beta, W, W_0 = argmin_{\alpha,\beta,W,W_0}(NLML_{k_{amGPR}}(\alpha, \beta, W, W_0))$	14	3 + N = 6	✓

Table 7.1: Simplified kernels with information about each regression method

our previous experiments we run only a single trial for each ground truth distribution. Then, we average our results across a total of 42 trials. Again, to ensure consistency, we will use the same randomly sampled data points for each trial.

As seen in Fig. 7.1, on average, we find very similar trends to the results in our previous sections. Our naive implementation of intrinsic GPR has a significantly higher MSE than the extrinsic approaches to GPR. This demonstrates the continual ill-defined nature of the intrinsic kernel (Eq. 4.1). Furthermore, after 500 sampled data points the unsupervised methods for embedding results in very similar MSEs, regardless of the embedding method, with MDS slightly outperforming Isomap. Finally both supervised methods, mGPR and amGPR, far outperform the unsupervised methods for embedding with mGPR resulting in a 25 % decrease in training data and a 48 % decrease in testing error as compared to the Original Extrinsic GPR while amGPR results in a 23 % decrease in training data and a 37 % decrease in testing error as compared to the Original Extrinsic GPR.

7.2 Uncertainty Sampling

Up to this point, our experiments have focused on the non-parametric nature of GPR by fitting a subset of our test distributions and using its predicted mean in order to extrapolate the entire distribution. However we can further test our models by incorporating uncertainty sampling into our experimentation. This is accomplished by altering the sampling scheme used, replacing the random sampling scheme with uncertainty sampling, Eq. 2.16. The goal of uncertainty sampling is to improve a model's fit of a distribution by only sampling points which best improve the model. In previous work [1, 2, 3] a well-conditioned estimate of uncertainty leads to a decrease in MSE, as the GPR results in a good estimate an uncertainty in its prediction.

Similar to our previous experiments, we will evaluate our models on the original four distributions data by dividing each distribution into training and testing data sets and sampling up to 500 data points. However, this time, instead of randomly sampling our independent data points, we will use a variation of uncertainty sampling from Eq. 2.16. Unlike traditional uncertainty sampling, we will instead only use uncertainty sampling every 5 sampled data points, otherwise we will continue to use random sampling. After these experiments, we will then average across these four trials. We will also run an identical experiment using purely random sampling as a base line for comparing how well incorporating uncertainty sampling does to improve the results.

Comparing these two sampling schemes, as seen in Fig. 7.2, we find all of our uncertainty estimates are well-conditioned as demonstrated by the drop in averaged MSE between the randomly sampling trial and the uncertainty sampling trials. Furthermore both supervised methods, mGPR and amGPR, far outperform the unsupervised methods

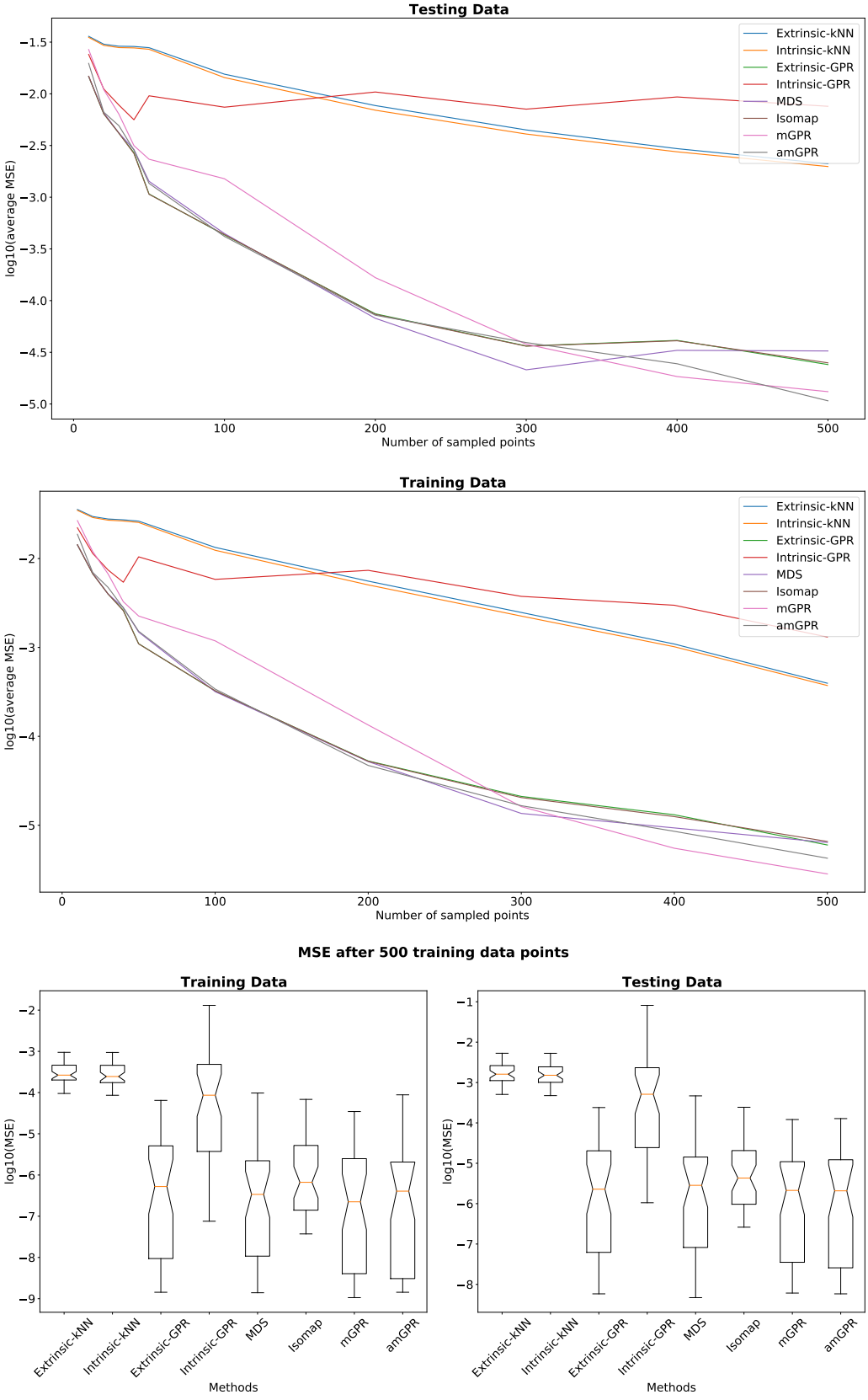


Figure 7.1: Comparison of the different supervised and unsupervised approaches to embedding in conjunction with GPR averaged across 60 different test distributions.

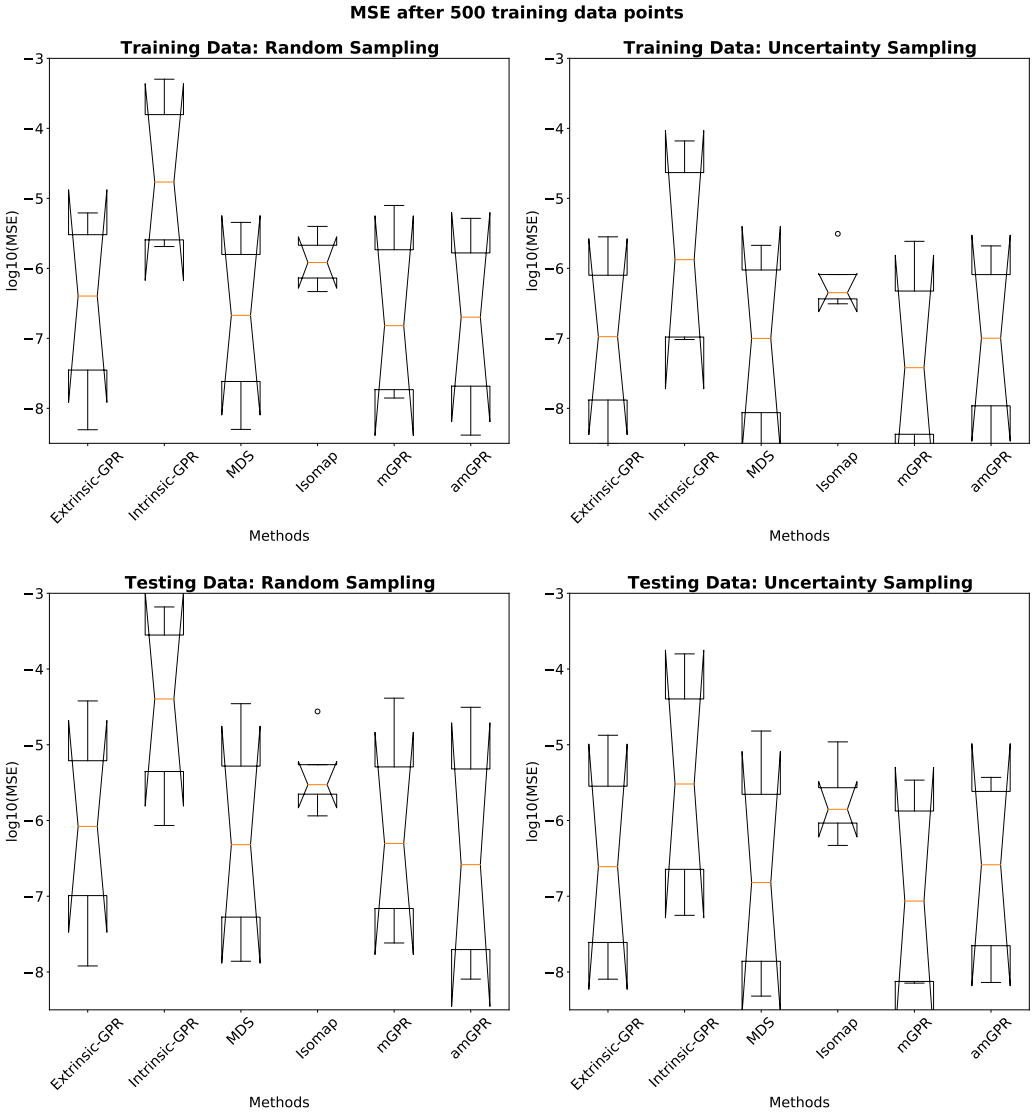


Figure 7.2: Random sampling vs Uncertainty Sampling schemes comparing the different supervised and unsupervised approaches to embedding in conjunction with GPR.

for embedding with mGPR resulting in a 19 % decrease in training data and a 70 % decrease in testing error as compared to the Original Extrinsic GPR while amGPR results in a 19 % decrease in training data and a 61 % decrease in testing error as compared to the Original Extrinsic GPR. This experiment thereby demonstrates that our supervised methods for embedding, mGPR and amGPR, contain both a well defined prediction and a good estimate of the uncertainty on this prediction. Similar to the work presented in Zevallos et. al. [1] these methods are therefore well suited for efficient stiffness mapping of an unknown object.

7.3 Real-World Data

Our final test incorporates the 'real-world' stiffness distributions collected in Jingyi Xu et. al. [5]. Similar to the work presented in Pai et al. [7, 32], these stiffness maps were collected using a robotic mechanism, which was used to palpate the surface of 5 example objects: a shampoo bottle, a w5 bottle, a box, a plastic cup, and a mustard bottle. These real-world distributions can be seen in Appendix B. We further test our supervised and unsupervised models by repeating the same experiment in Sec. 7.2 using these real-world data sets.

As seen in Fig. 7.3, these results follow a very similar trend to the results seen for the Gaussian distributions. Each uncertainty sampling performs better than their random sampling counter parts. Furthermore both supervised methods, mGPR and amGPR, far outperform the unsupervised methods for embedding with mGPR resulting in an 11 % decrease in training data and a 33 % decrease in testing error as compared to the Original Extrinsic GPR while amGPR results in a 17 % decrease in training data and a 30 % decrease in testing error as compared to the Original Extrinsic GPR. Therefore we can conclude that our supervised methods for embedding along with GPR result in the the most optimal method for constructing a stiffness map of a deformable object.

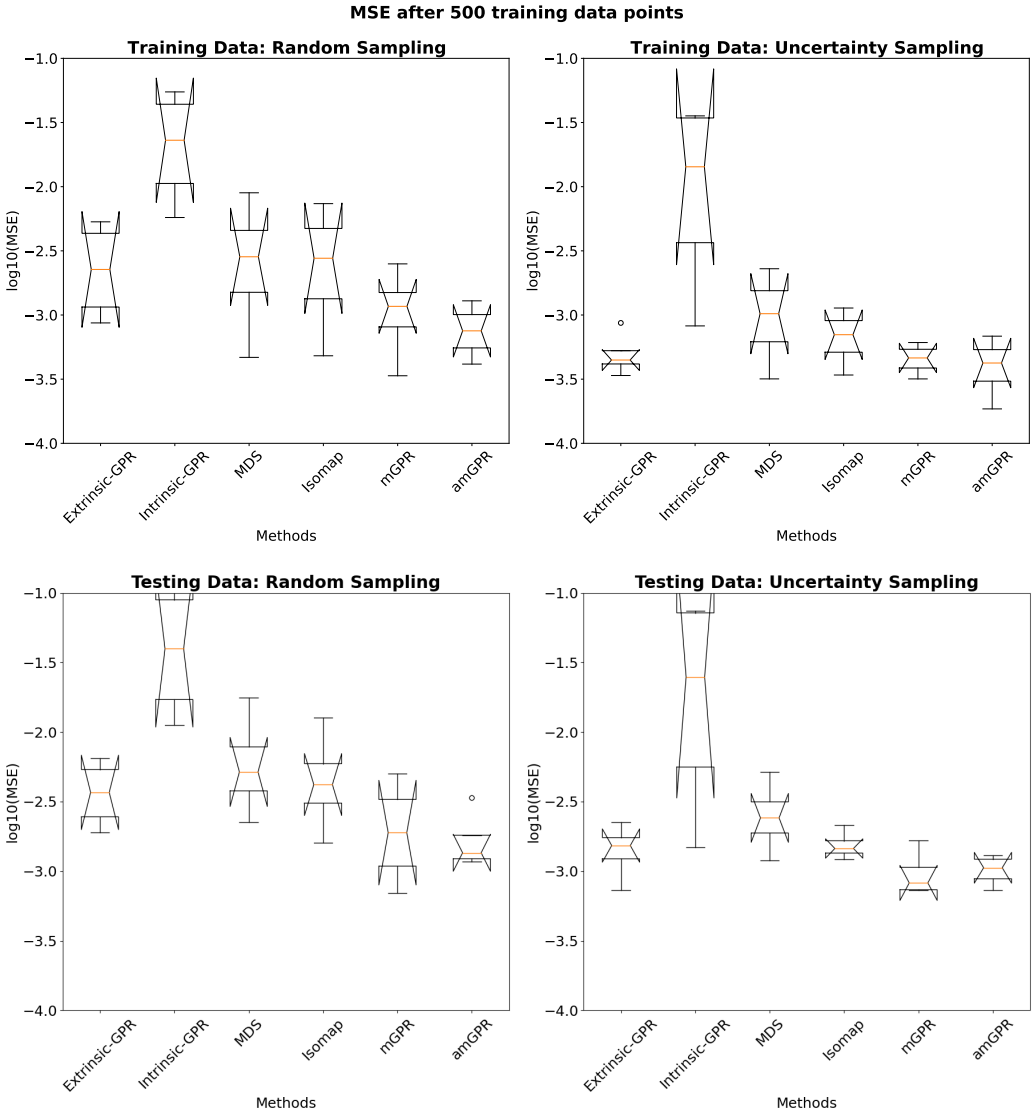


Figure 7.3: Comparison of the different supervised and unsupervised approaches to embedding in conjunction with GPR averaged across 5 real-world stiffness distributions.

Chapter 8

Conclusion and Future Work

This thesis introduces supervised embedding, a new approach for fitting GPR with predictors sampled from a compact manifold. By introducing a parametric function mapping the data to a new embedding, we can optimize these parameters simultaneously with the parameters in the GPR kernel. Therefore supervised learning strategies can find an embedding as a function of the data it is trying to fit. This thesis introduces two approaches to supervised embedding, mGPR and amGPR. These two methods differ slightly in how they incorporate their parametric function into the embedding. mGPR utilizes the functional mapping to redefine the manifold's embedding completely, while amGPR appends a higher dimensional embedding to the original data. The development of these approaches to supervised embedding as a part of GPR is the primary contribution of this thesis.

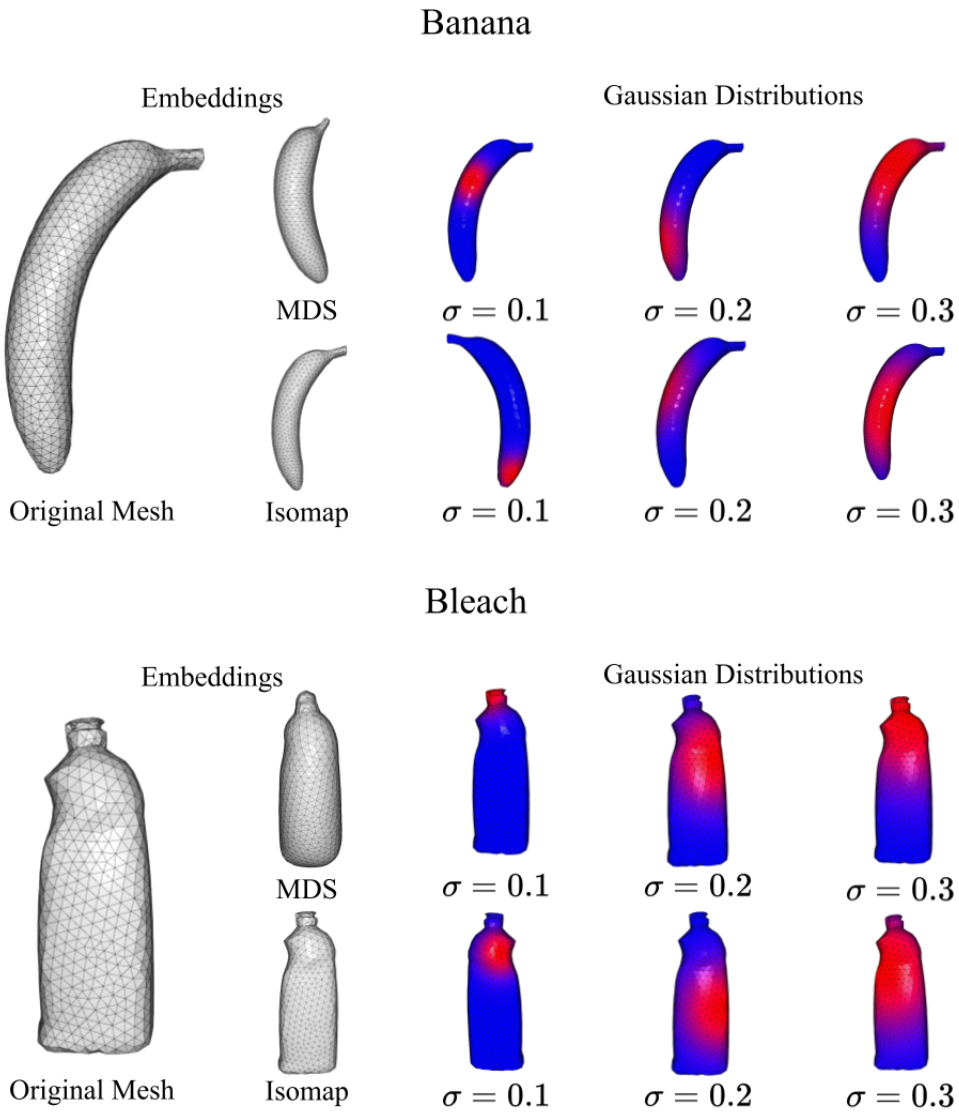
As this work is primarily motivated by stiffness mapping, we tested these models on a series of 42 evaluation distributions consisting of predictors sourced from the YCB data set. For the dependent data, we projected a Gaussian distribution onto the surface of these meshes and sampled at each mesh vertex. We demonstrated the generalizability of our models by testing on these 42 evaluation data sets. We further validated these models using real-world stiffness maps of three-dimensional objects. We demonstrated that these supervised methods, mGPR and amGPR, can fit data with predictors sampled from compact manifolds better than traditional, unsupervised approaches like MDS and Isomap or a naive embedding method.

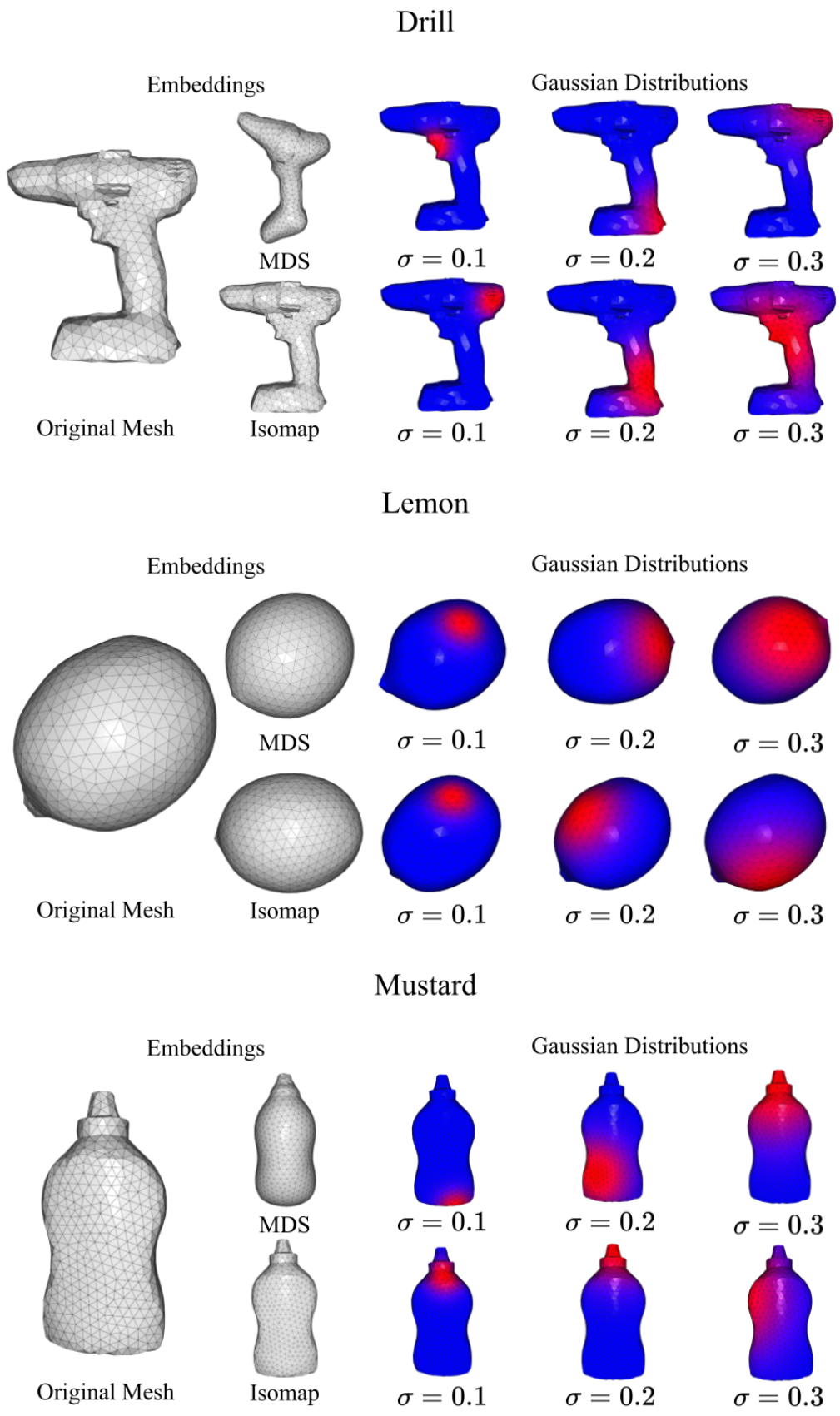
The current limitation of these supervised methods for embedding stems from the boundaries used as a part of their kernel's optimization. This value is a constraint on the hyperparameter. Although it did not require a significant amount of time to tune these boundaries, a single solution did not generalize to all test distributions. In the future, it will be essential to explore more robust embedding functions that require minor tuning of their boundary conditions. One possible solution may be to tweak the function (g) to use a different activation function instead of the hyperbolic tangent; perhaps, a less steep derivative will help supervised methods generalize.

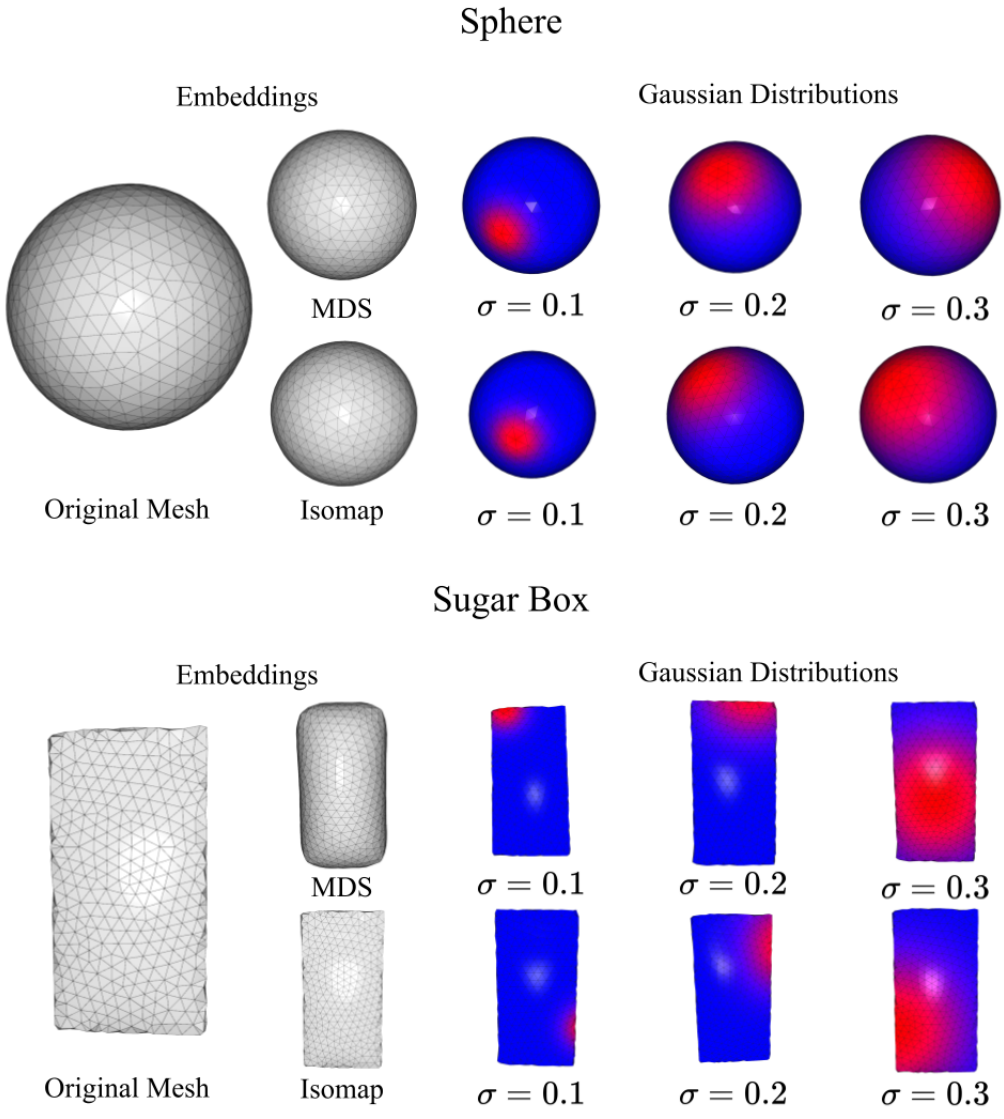
Stiffness maps, like the ones used to validate our models, are helpful in the field of deformable manipulation [5]. In the future, we would like to see these algorithms applied to scenarios where sampling schemes are crucial, such as the efficient generation of stiffness data sets. Understanding the physical properties of three-dimensional objects is an essential step to making more object-aware grasping policies. Through the development of algorithms like amGPR, we will hopefully promote these kinds of data sets. In our work, we have begun to develop the hardware capable of this future goal [37, 38].

Appendix A

Gaussian Distributions

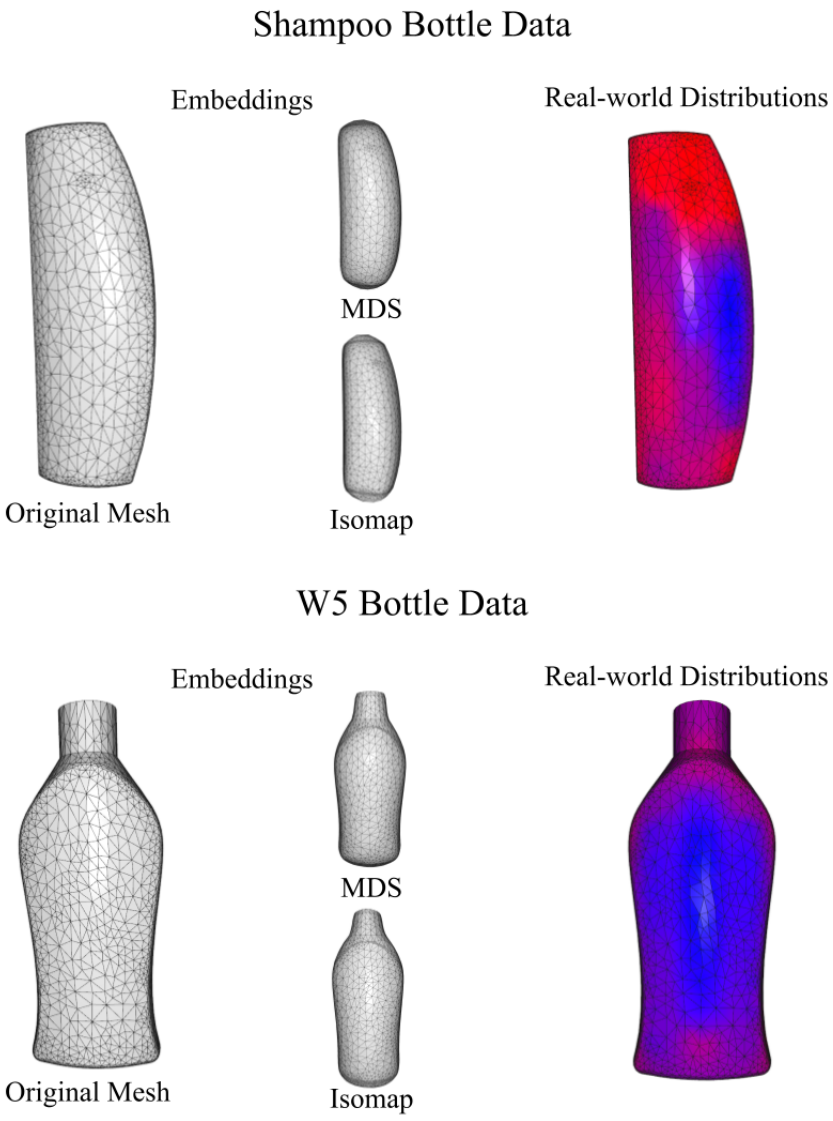


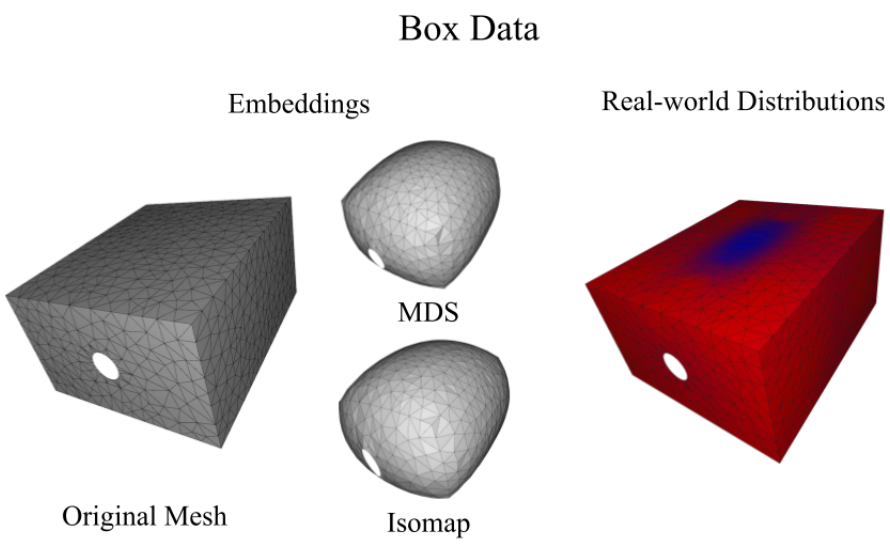

Drill
Embeddings
Gaussian Distributions
MDS
$\sigma = 0.1$
$\sigma = 0.2$
$\sigma = 0.3$
Original Mesh
Isomap
$\sigma = 0.1$
$\sigma = 0.2$
$\sigma = 0.3$
Lemon
Embeddings
Gaussian Distributions
MDS
$\sigma = 0.1$
$\sigma = 0.2$
$\sigma = 0.3$
Original Mesh
Isomap
$\sigma = 0.1$
$\sigma = 0.2$
$\sigma = 0.3$
Mustard
Embeddings
Gaussian Distributions
MDS
$\sigma = 0.1$
$\sigma = 0.2$
$\sigma = 0.3$
Original Mesh
Isomap
$\sigma = 0.1$
$\sigma = 0.2$
$\sigma = 0.3$


Sphere
Embeddings
Gaussian Distributions
MDS
$\sigma = 0.1$
$\sigma = 0.2$
$\sigma = 0.3$
Original Mesh
Isomap
$\sigma = 0.1$
$\sigma = 0.2$
$\sigma = 0.3$
Sugar Box
Embeddings
Gaussian Distributions
MDS
$\sigma = 0.1$
$\sigma = 0.2$
$\sigma = 0.3$
Original Mesh
Isomap
$\sigma = 0.1$
$\sigma = 0.2$
$\sigma = 0.3$

Appendix B

Real world Distributions




Box Data
Embeddings
Real-world Distributions
MDS
Original Mesh
Isomap
Plastic Cup Data
Embeddings
Real-world Distributions
MDS
Original Mesh
Isomap
Mustard Bottle 2 Data
Embeddings
Real-world Distributions
MDS
Original Mesh
Isomap

Appendix C

MDS Embeddings

	Original	N=1	N=2	N=3	N=4	N=5	N=6
mustard	6.26e-03	4.15e-02	7.87e-03	1.02e-03	1.02e-03	1.02e-03	1.02e-03
mustard-external	3.39e-03	1.13e-01	3.33e-03	6.03e-04	6.03e-04	6.03e-04	6.03e-04
drill	1.14e-02	4.41e-02	4.44e-03	7.81e-04	5.37e-04	5.06e-04	4.94e-04
bleach	3.55e-03	1.55e-02	4.48e-03	5.45e-04	5.45e-04	5.45e-04	5.45e-04
box-external	4.27e-02	1.56e-01	4.23e-02	5.48e-03	5.48e-03	5.48e-03	5.48e-03
banana	9.23e-04	1.38e-02	1.51e-03	1.76e-04	1.76e-04	1.76e-04	1.76e-04
shampoo-external	1.58e-03	9.59e-02	2.05e-03	3.90e-04	3.90e-04	3.90e-04	3.90e-04
plastic-cup-external	1.16e-02	1.19e-01	2.32e-02	2.25e-03	2.25e-03	2.25e-03	2.25e-03
sphere	2.77e-02	1.81e-01	5.36e-02	7.01e-03	7.01e-03	7.01e-03	7.01e-03
w5-external	2.12e-03	5.90e-02	2.47e-03	4.94e-04	4.94e-04	4.94e-04	4.94e-04
sugar-box	8.62e-03	4.24e-02	9.96e-03	1.40e-03	1.40e-03	1.40e-03	1.40e-03
hammer	3.68e-03	5.55e-02	5.35e-04	7.66e-05	6.01e-05	5.90e-05	5.87e-05
lemon	2.43e-02	1.50e-01	4.60e-02	5.97e-03	5.97e-03	5.97e-03	5.97e-03

Table C.1: Reconstruction Error for MDS embedding into $\mathbb{R}^N$

	Original	N=1	N=2	N=3	N=4	N=5	N=6
mustard	5.35e-35	3.96e-02	1.69e-04	4.98e-05	4.98e-05	4.98e-05	4.98e-05
mustard-external	3.30e-12	2.40e-01	2.03e-05	8.66e-06	8.66e-06	8.66e-06	8.66e-06
drill	7.50e-35	3.76e-02	1.07e-04	6.33e-05	6.71e-05	7.67e-05	8.16e-05
bleach	4.25e-35	2.00e-04	1.24e-04	3.61e-05	3.61e-05	3.61e-05	3.61e-05
box-external	1.43e-34	1.29e-03	4.45e-04	1.20e-04	1.20e-04	1.20e-04	1.20e-04
banana	2.35e-35	2.17e-02	9.63e-05	2.92e-05	2.92e-05	2.92e-05	2.93e-05
shampoo-external	4.00e-10	1.88e-01	8.38e-05	1.23e-05	1.23e-05	1.23e-05	1.23e-05
plastic-cup-external	1.99e-35	1.43e-01	4.46e-05	1.09e-05	1.09e-05	1.09e-05	1.09e-05
sphere	9.40e-35	6.68e-03	7.61e-04	1.18e-04	1.18e-04	1.18e-04	1.18e-04
w5-external	1.30e-12	1.08e-01	3.34e-05	1.45e-05	1.45e-05	1.45e-05	1.45e-05
sugar-box	6.58e-35	3.17e-04	1.63e-04	6.49e-05	6.49e-05	6.49e-05	6.49e-05
hammer	1.74e-35	9.37e-02	1.99e-03	1.59e-05	1.14e-05	1.16e-05	1.19e-05
lemon	9.54e-35	1.23e-03	5.13e-04	1.11e-04	1.11e-04	1.11e-04	1.11e-04

Table C.2: Local Reconstruction Error for MDS embedding into $\mathbb{R}^N$

Appendix D

Isomap Embeddings

	Original	N=1	N=2	N=3	N=4	N=5	N=6
mustard	6.26e-03	3.86e-02	1.38e-02	5.91e-03	5.76e-03	5.68e-03	5.63e-03
mustard-external	3.39e-03	2.53e-02	5.08e-03	1.26e-03	1.36e-03	1.46e-03	1.59e-03
drill	1.14e-02	6.80e-02	1.62e-02	1.14e-02	1.13e-02	1.13e-02	1.13e-02
bleach	3.55e-03	2.06e-02	8.07e-03	3.41e-03	3.33e-03	3.28e-03	3.26e-03
box-external	4.27e-02	2.36e-01	8.39e-02	3.68e-02	3.59e-02	3.52e-02	3.47e-02
banana	9.23e-04	8.00e-03	2.76e-03	9.15e-04	8.96e-04	8.87e-04	8.81e-04
shampoo-external	1.58e-03	1.49e-02	2.91e-03	7.67e-04	8.34e-04	9.16e-04	9.76e-04
plastic-cup-external	1.16e-02	1.01e-01	2.99e-02	2.94e-03	4.07e-03	5.39e-03	5.98e-03
sphere	2.77e-02	2.56e-01	8.19e-02	9.29e-03	9.83e-03	1.08e-02	1.21e-02
w5-external	2.12e-03	2.12e-02	3.54e-03	1.06e-03	1.06e-03	1.09e-03	1.10e-03
sugar-box	8.62e-03	5.54e-02	1.62e-02	8.45e-03	8.35e-03	8.30e-03	8.26e-03
lemon	2.43e-02	2.13e-01	7.64e-02	1.19e-02	1.17e-02	1.16e-02	1.17e-02

Table D.1: Reconstruction Error for Isomap embedding into $\mathbb{R}^N$

	Original	N=1	N=2	N=3	N=4	N=5	N=6
mustard	5.35e-35	3.38e-04	6.39e-05	4.97e-07	1.21e-05	2.80e-05	3.74e-05
mustard-external	3.30e-12	4.88e-05	1.23e-05	8.79e-06	2.10e-05	3.36e-05	4.11e-05
drill	7.50e-35	4.64e-04	9.70e-05	2.99e-08	9.61e-06	1.27e-05	1.97e-05
bleach	4.25e-35	2.50e-04	5.75e-05	1.81e-07	7.24e-06	1.92e-05	2.45e-05
box-external	1.43e-34	1.05e-03	3.32e-04	2.18e-06	2.74e-05	6.77e-05	1.11e-04
banana	2.35e-35	1.59e-04	4.80e-05	8.71e-08	1.92e-06	6.43e-06	1.33e-05
shampoo-external	4.00e-10	8.27e-05	1.73e-05	6.47e-06	1.60e-05	2.52e-05	3.98e-05
plastic-cup-external	1.99e-35	9.16e-05	2.60e-05	1.87e-05	5.10e-05	8.96e-05	1.38e-04
sphere	9.40e-35	9.25e-04	3.01e-04	5.35e-05	1.84e-04	3.34e-04	5.41e-04
w5-external	1.30e-12	9.48e-05	1.98e-05	6.77e-06	1.70e-05	2.20e-05	3.60e-05
sugar-box	6.58e-35	3.92e-04	1.06e-04	2.31e-07	1.90e-05	3.00e-05	4.63e-05
lemon	9.54e-35	7.19e-04	1.75e-04	2.22e-05	1.17e-04	2.27e-04	3.46e-04

Table D.2: Local Reconstruction Error for Isomap embedding into $\mathbb{R}^N$

Bibliography

[1] N. Zevallos, R. A. Srivatsan, H. Salman, L. Li, J. Qian, S. Saxena, M. Xu, K. Patath, and H. Choset, "A surgical system for automatic registration, stiffness mapping and dynamic image overlay," 2017.

[2] N. Zevallos, A. S. Rangaprasad, H. Salman, L. Li, J. Qian, S. Saxena, M. Xu, K. Patath, and H. Choset, "A real-time augmented reality surgical system for overlaying stiffness information," 06 2018.

[3] E. Ayvali, A. S. Rangaprasad, L. Wang, R. Roy, N. Simaan, and H. Choset, "Using bayesian optimization to guide probing of a flexible environment for simultaneous registration and stiffness mapping," 05 2016, pp. 931–936.

[4] B. Calli, A. Walsman, A. Singh, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Benchmarking in manipulation research: Using the yale-cmu-berkeley object and model set," *IEEE Robotics Automation Magazine*, vol. 22, no. 3, pp. 36–52, 2015.

[5] J. Xu, M. Danielczuk, J. Ichnowski, J. Mahler, E. G. Steinbach, and K. Goldberg, "Minimal work: A grasp quality metric for deformable hollow objects," *CoRR*, vol. abs/1909.11226, 2019. [Online]. Available: http://arxiv.org/abs/1909.11226

[6] F. S. Sin, D. Schroeder, and J. Barbic, "Vega: Non-Linear FEM Deformable Object Simulator," *Computer Graphics Forum*, 2013.

[7] D. Pai, K. van den Doel, D. James, J. Lang, J. Lloyd, J. Richmond, and S. Yau, "Scanning physical interaction behavior of 3d objects," *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 06 2001.

[8] Z. Chen and B. Wang, "How priors of initial hyperparameters affect gaussian process regression models," *Neurocomputing*, vol. 275, p. 1702–1710, Jan 2018. [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2017.10.028

[9] R. Gramacy, *Surrogates: Gaussian Process Modeling, Design, and Optimization for the Applied Sciences*. Chapman and Hall, 03 2020.

[10] R. Calandra, J. Peters, C. E. Rasmussen, and M. P. Deisenroth, "Manifold gaussian processes for regression," 2016.

[11] V. Borovitskiy, A. Terenin, P. Mostowsky, and M. P. Deisenroth, "Matérn gaussian processes on riemannian manifolds," 2020.

[12] L. Lin, M. Niu, P. Cheung, and D. Dunson, "Extrinsic gaussian processes for regression and classification on manifolds," *Bayesian Analysis*, vol. 14, 06 2017.

[13] E. Snelson, Z. Ghahramani, and C. Rasmussen, "Warped gaussian processes," in *Advances in Neural Information Processing Systems*, S. Thrun, L. Saul, and B. Schölkopf, Eds., vol. 16. MIT Press, 2004. [Online]. Available: https://proceedings.neurips.cc/paper/2003/file/6b5754d737784b51ec5075c0dc437bf0-Paper.pdf

[14] K. Crane, C. Weischedel, and M. Wardetzky, "The heat method for distance computation," *Commun. ACM*, vol. 60, no. 11, pp. 90–99, Oct. 2017. [Online]. Available: http://doi.acm.org/10.1145/3131280

[15] L. Cayton, "Algorithms for manifold learning," 2005.

[16] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications*. Springer, 2005.

[17] J. B. Tenenbaum, V. d. Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, 2000. [Online]. Available: https://science.sciencemag.org/content/290/5500/2319

[18] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.290.5500.2323

[19] W. Liu, A. Daruna, and S. Chernova, "Cage: Context-aware grasping engine," 2020.

[20] N. Zevallos, E. Harber, Abhimanyu, K. Patel, Y. Gu, K. Sladick, F. Guyette, L. Weiss, M. R. Pinsky, H. Gomez, J. Galeotti, and H. Choset, "Toward robotically automated femoral vascular access," in *2021 International Symposium on Medical Robotics (ISMR)*, 2021, pp. 1–7.

[21] S. J. Russell and P. Norvig, *Artificial Intelligence: a modern approach*, 3rd ed. Pearson, 2009.

[22] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical Programming*, vol. 45, no. 1-3, pp. 503–528, Aug. 1989. [Online]. Available: https://doi.org/10.1007/bf01589116

[23] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 509–516.

[24] J. S. B. Mitchell, D. M. Mount, and C. H. Papadimitriou, "The discrete geodesic problem," *SIAM Journal on Computing*, vol. 16, no. 4, pp. 647–668, 1987. [Online]. Available: https://doi.org/10.1137/0216045

[25] V. Surazhsky, T. Surazhsky, D. Kirsanov, S. J. Gortler, and H. Hoppe, "Fast exact and approximate geodesics on meshes," *ACM Trans. Graph.*, vol. 24, no. 3, p. 553–560, Jul. 2005. [Online]. Available: https://doi.org/10.1145/1073204.1073228

[26] N. Sharp, Y. Soliman, and K. Crane, "The vector heat method," *CoRR*, vol. abs/1805.09170, 2018. [Online]. Available: http://arxiv.org/abs/1805.09170

[27] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia, "Meshlab: an open-source mesh processing tool." *Computing*, vol. 1, pp. 129–136, 01 2008.

[28] E. S. Nicolas Alt, Jingyi Xu, "A dataset of thin-walled deformable objects for manipulation planning," 2016.

[29] ——, "Grasp planning for thin-walled deformable objects," 2015.

[30] "Model-based strategy for grasping 3d deformable objects using a multi-fingered robotic hand," *Robotics and Autonomous Systems*, vol. 95, pp. 196–206, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889016308089

[31] D. James and D. Pai, "A unified treatment of elastostatic contact simulation for real time haptics," *Haptics-e, the Electronic Journal of Haptics Research*, vol. 2, 10 2001.

[32] J. Lang, D. Pai, and R. Woodham, "Acquisition of elastic models for interactive simulation," *The International Journal of Robotics Research, v.21, 713-733 (2002)*, vol. 21, 08 2002.

[33] R. A. Srivatsan, E. Ayvali, L. Wang, R. Roy, N. Simaan, and H. Choset, "Complementary model update: A method for simultaneous registration and stiffness mapping in flexible environments," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 924–930.

[34] P. L. Robinson, "The sphere is not flat," *The American Mathematical Monthly*, vol. 113, no. 2, pp. 171–173, 2006. [Online]. Available: http://www.jstor.org/stable/27641870

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[36] R. Antonova, A. Rai, and C. G. Atkeson, "Deep kernels for optimizing locomotion controllers," 2017.

[37] E. Harber, E. Schindewolf, V. Webster-Wood, H. Choset, and L. Li, "A tunable magnet-based tactile sensor framework," in *2020 IEEE SENSORS*, 2020, pp. 1–4.

[38] K. Dai, X. Wang, A. M. Rojas, E. Harber, Y. Tian, N. Paiva, J. Gnehm, E. Schindewolf, H. Choset, V. A. Webster-Wood, and L. Li, "Design of a biomimetic tactile sensor for material classification," 2022. [Online]. Available: https://arxiv.org/abs/2203.15941

[39] T. Fletcher, "Geodesic Regression on Riemannian Manifolds," in *Proceedings of the Third International Workshop on Mathematical Foundations of Computational Anatomy - Geometrical and Statistical Methods for Modelling Biological Shape Variability*, Pennec, Xavier, Joshi, Sarang, Nielsen, and Mads, Eds., Toronto, Canada, Sep. 2011, pp. 75–86. [Online]. Available: https://hal.inria.fr/inria-00623920

[40] R. Chakraborty, J. Bouza, J. Manton, and B. C. Vemuri, "Manifoldnet: A deep network framework for manifold-valued data," 2018.

[41] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari, "Accelerating 3d deep learning with pytorch3d," *arXiv:2007.08501*, 2020.

[42] G. Gkioxari, J. Malik, and J. Johnson, "Mesh r-cnn," 2020.

[43] N. Jaquier, L. Rozo, S. Calinon, and M. Bürger, "Bayesian optimization meets riemannian manifolds in robot learning," 2019.

[44] M. Saha and P. Isto, "Manipulation planning for deformable linear objects," *IEEE Transactions on Robotics*, vol. 23, no. 6, pp. 1141–1150, 2007.

[45] S. Zimmermann, R. Poranne, and S. Coros, "Dynamic manipulation of deformable objects with implicit integration," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4209–4216, 2021.

[46] J. Huang, P. Di, T. Fukuda, and T. Matsuno, "Dynamic modeling and simulation of manipulating deformable linear objects," in *2008 IEEE International Conference on Mechatronics and Automation*, 2008, pp. 858–863.

[47] O. Roussel and M. Taïx, "Deformable linear object manipulation planning with contacts," in *IROS 2014*, 2014.