TOWARDS AN EFFICIENT HAPTIC RENDERING USING DATA-DRIVEN MODELING

by

Wael Abdelrahman
M.Sc, B.Sc

Submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

Deakin University
March, 2012
I am the author of the thesis entitled

Towards an Efficient Haptic Rendering Using Data-Driven Modeling

submitted for the degree of Doctor of Philosophy

This thesis may be made available for consultation, loan and limited copying in accordance with the Copyright Act 1968.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name ...........................................Wael Abdelrahman.................................
(Please Print)

Signed ..............................................Signature Redacted by Library

Date .................................................22/5/2012..............................................
I certify that the thesis entitled (10 word maximum)

Towards an efficient haptic rendering using data-driven modeling

submitted for the degree of

Doctor of Philosophy

is the result of my own work and that where reference is made to the work of others, due acknowledgment is given.

I also certify that any material in the thesis which has been accepted for a degree or diploma by any university or institution is identified in the text.

'I certify that I am the student named below and that the information provided in the form is correct'

Full Name.................................Wael Abdelrahman........................................
(Please Print)

Signed .................. .................................................................

Date ................. 25/3/2012 .................................................................

Signature Redacted by Library
To my wife, son and parents.
Table of Contents

Table of Contents v

List of Tables ix

List of Figures x

Acknowledgments xiv

Abstract xv

Acronyms xvii

List of Publications xix

1 Introduction 1

1.1 Challenges in data-driven modeling ....................... 2

1.1.1 Data acquisition ....................................... 3

1.1.2 Model training and data storage ........................ 3

1.1.3 Data generalization ..................................... 4

1.2 Research contribution ..................................... 5
2 Literature review

2.1 Haptic rendering .................................................. 8

2.2 Haptic rendering of deformable models .......................... 12

2.2.1 Modeling approaches ........................................... 12

2.3 Data-driven methods ................................................. 20

2.3.1 Data-driven model ................................................. 20

2.3.2 Current data-driven methods in haptic rendering .......... 22

2.3.3 Current data-driven modeling vs. parametric modeling ... 24

2.3.4 Limitations in data-driven haptic rendering .............. 26

2.4 Conclusion ............................................................ 29

3 Optimizing the data collection phase of the data-driven simulation 30

3.1 Overview ............................................................. 30

3.1.1 The need for high volume of data ............................. 31

3.1.2 Parameterized storage of curves ............................... 33

3.2 Related data collection methodologies .......................... 34

3.3 Proposed data collection techniques ............................. 35

3.3.1 Stage 1 - Object segmentation ................................. 38

3.3.2 Stage 2 - Points selection from segments .................. 39

3.3.3 Stage 3 - Force effect linearization .......................... 41

3.4 Simulation results of the proposed data collection methods ... 43
4 Selection of suitable learning methods and data formats for the haptic interaction simulation

4.1 Overview .................................... 53
4.2 Definitions and previous work ...................... 57
  4.2.1 Data-driven haptics .......................... 57
  4.2.2 Supervised learning techniques ................ 58
4.3 Comparative results ................................ 61
4.4 Conclusion ...................................... 68

5 Generalization of data-driven methods ............... 69

5.1 Overview ....................................... 69
5.2 Multi-point interactions modeling .................... 71
  5.2.1 Related literature .......................... 71
  5.2.2 Data-driven representation of multiple contacts .... 73
  5.2.3 Data-driven computation of contact dynamics ....... 78
5.3 Simulation results of the multi-point model ............. 81
5.4 Deformation transfer to similar objects ............... 88
  5.4.1 Shape correspondence ........................ 88
5.5 Conclusion ...................................... 90

6 Virtual grasp - case study .......................... 91
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.1</td>
<td>Overview</td>
<td>91</td>
</tr>
<tr>
<td>6.2</td>
<td>Related works</td>
<td>92</td>
</tr>
<tr>
<td>6.3</td>
<td>System description</td>
<td>94</td>
</tr>
<tr>
<td>6.3.1</td>
<td>Multi-point haptic gripper - Hardware</td>
<td>94</td>
</tr>
<tr>
<td>6.3.2</td>
<td>Design objectives</td>
<td>95</td>
</tr>
<tr>
<td>6.3.3</td>
<td>Design descriptions</td>
<td>96</td>
</tr>
<tr>
<td>6.3.4</td>
<td>Data collection and replay frameworks - Software</td>
<td>98</td>
</tr>
<tr>
<td>6.4</td>
<td>Model building</td>
<td>101</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Data collection and system calibration</td>
<td>101</td>
</tr>
<tr>
<td>6.4.2</td>
<td>ANN training</td>
<td>104</td>
</tr>
<tr>
<td>6.5</td>
<td>Results</td>
<td>106</td>
</tr>
<tr>
<td>6.6</td>
<td>Conclusion</td>
<td>107</td>
</tr>
<tr>
<td>7</td>
<td>Conclusion</td>
<td>109</td>
</tr>
<tr>
<td>7.1</td>
<td>Research outcomes</td>
<td>109</td>
</tr>
<tr>
<td>7.2</td>
<td>Potential applications</td>
<td>111</td>
</tr>
<tr>
<td>7.2.1</td>
<td>Haptically enabled medical atlas</td>
<td>112</td>
</tr>
<tr>
<td>7.2.2</td>
<td>Low-cost low-processing power applications</td>
<td>113</td>
</tr>
<tr>
<td>7.2.3</td>
<td>Hybrid systems</td>
<td>114</td>
</tr>
<tr>
<td>7.3</td>
<td>Future research directions</td>
<td>115</td>
</tr>
</tbody>
</table>

References | 117 |
# List of Tables

2.1 A comparison between current data-driven techniques and parametric techniques in haptic simulation of deformable objects. ........................................ 26

4.1 The performance data of the different supervised learning techniques for different training data percentages. .................................................. 63

4.2 The performance data of the different supervised learning techniques for different training data percentages (contd.). .................................... 64

6.1 The data structure employed. ................................................................... 103
## List of Figures

2.1 Haptic process components and data flow. The proxy is the representation of the haptic device in the virtual environment. .................. 10

2.2 Examples of commercial haptic devices that have only one point of contact. .................................. 11

2.3 Object deformation model where \( O' = def(O, f) \). ................................ 13

2.4 Modeling approaches of deformable models. .................. 14

2.5 Non physical modeling example, where 3D mesh is reduced to a skeleton and the skeleton is controlled by selected control points. ... 15

2.6 Lagrangian mesh vs. Eulerian mesh. .............................. 17

2.7 Data-driven modeling approach for haptic simulation of deformable models. ........................................ 21

2.8 Empirical data are used in calibrating the object material parameters in parametric modeling techniques. .................. 25

2.9 Multi point haptic device examples. .............................. 28

3.1 The dynamic simulation data represented in an entity relationship diagram (ERD). .......................... 32

3.2 Parameterized storage of a 3D liver object into a set of planes. ... 34

3.3 Flowchart of the data collection phase of the data-driven modeling approach for deformable models. .................. 37
3.4 Segmenting 3D liver mesh into flat surfaces.

3.5 Selection of vertices (orange are selected) reduce the volume of data to nearly half.

3.6 3D object simulations. The region at the base of the object (green) represents constraints to fix vertices. The singular arrows represents an external force.

3.7 The configuration of neural network that is used in the simulation.

3.8 The number of segments is inversely proportional to the average number of vertices per segment.

3.9 Different values of θ\text{threshold} and the corresponding number of segments (solid green), average number of vertices per segment (dotted blue) and average number of selected vertices (dashed blue).

3.10 The performance of the artificial neural network (ANN) training of the 3D liver model. The difference in validation performance between choosing all vertices and selecting a subset is minimal when θ = 0.74.

3.11 Comparison between the mean ANN validation error when choosing all points (dashed blue) vs. using a subset of points using the proposed algorithm (solid red).

4.1 The diagram of the inputs and outputs of the learning algorithm. Each circle denotes a three dimensional vector.

4.2 3D finite element methods (FEM) simulation in Abaqus™. Two external forces are applied on the deformable object.

4.3 Coefficient of determination (R²) comparison.

4.4 Mean absolute error (MAE) comparison.

4.5 Root mean squared error (\sqrt{MSE}) comparison.
4.6 Relative absolute error (RAE) comparison. ................................. 66
4.7 Root relative square error (\(\sqrt{RSE}\)) comparison. .................. 67

5.1 Example of medical instruments that typically require multiple points of contact. ........................................................................... 70
5.2 Right and left kidneys are not completely identical in size and geometry. .................................................................................. 70
5.3 The effect of applying two simultaneous forces on deformable object is not the same as the linear summation of the effect of each individually. ................................................................. 75
5.4 In both figures the distance between the contact points does not fully characterize the interaction. (a) To solve the ambiguity between the \((\mathbf{p}_1, \mathbf{p}_2)\) and \((\mathbf{p}_3, \mathbf{p}_4)\) pairs, an additional geometric information is added. (b) The angle \(\theta\) between the two displacement vectors is important to differentiate between varying orientations. ........................................ 77
5.5 The usage of axis lengths of an ellipse or an ellipsoid for the local coordinate system (for the 2D and 3D case, respectively). .......... 78
5.6 Diagrams of the employed ANNs. The left one is for the local coordinate and the right one for the global coordinate system. ........ 80
5.7 Simulation of 2D and 3D virtual objects. Two loads were applied on each model. ................................................................. 82
5.8 Example of random trajectories generated on each point. ......... 83
5.9 ANN performance using a global coordinate system on a 3D virtual object for 153 samples. ......................................................... 84
5.10 ANN performance using a local coordinate system on a 3D virtual object for 153 samples. ......................................................... 85
5.11 The relation between the sample size and the resulting error of the calculated results with respect to the actual recordings. 87

5.12 Data can be collected for only the marked areas and not the whole object. 89

6.1 Literature related multi-point haptic systems. 93

6.2 Flow diagram of a the VirtualGrasp application. 94

6.3 Dexterous grasping scenario to rotate a bolt. 96

6.4 The haptic gripper handle. 97

6.5 The gripper handle attached to the Force Dimension Omega device. 97

6.6 Representation of deformable box using CHAI 3D framework. 100

6.7 The data collection system. 102

6.8 Collected forces sample. 102

6.9 Collected displacements sample (right and left). 103

6.10 Motor calibration software interface. 104

6.11 Feature selection using WEKA software. 105

6.12 The ANN performance - Calculated force vs. Actual. 106

6.13 virtualGrasp interface with graphics and haptics frames per second (FPS). 107

7.1 A cross section 3D view of a human brain details in Google body. 112

7.2 Haptic simulation of a human liver in a typical medical application. 113
Acknowledgments

I would like to express my gratitude to my supervisors, Prof. Saeid Nahavandi and Dr. Douglas Creighton, for their unlimited continuous support.

I would like to thank Dr. habil. Matthias Harders from ETH university for letting me visit his lab. My stay over there was fruitful and very enjoyable as well.

I would also like to thank all the members of the Center for Intelligent Systems Research (CISR) at Deakin University. Special thanks to Dr. Zoran Najdovski for his support.

Very special thanks to my family for their encouragement and support. My deepest gratitude to my wife Sara, my beloved son Omar, my parents, and my brothers.
Abstract

Haptic simulation of deformable objects adds a new dimension for user interactions with virtual reality systems. The addition of the sense of touch complements the other senses and enables the user to feel, perceive, and investigate the virtual environment components. This enhanced capability permits the user to perform virtual procedures in a way that is as close as possible to the real world. The added sense of touch transfers additional information that cannot be easily communicated through alternative visual or auditory senses. The additional information is vital where a user needs to rapidly gain professional skills, such as medical training or where the access to the physical objects is prohibitive, as in the case of controlling robots navigating hazardous environments.

This thesis focuses on the planning process for the construction of data-driven models used to haptically simulate user interactions with deformable objects. The data-driven haptic rendering process is optimized and extended to handle complex material behavior, object geometry, and multi-point interaction scenarios. The contributions of this thesis are: a novel technique to optimize the selection of the training samples, a methodology to generalize the data utilization for multi-point, and introducing a multi-point system that has a real-time performance and accuracy close to offline methods such as finite element methods (FEM).

The planning targets different phases of the data-driven modeling method. The
data selection is done on the discretized object vertices and the external applied forces. The algorithms are adaptive with the targeted level of accuracy and they can work on any object topology aiming to optimize the processing power and the storage space. A data-driven model is introduced that can handle multiple points of interaction. The model captures the external applied forces and the mutual effects among them. This enables the modeling of real life tools such as medical ones which mostly needs at least two points of contact. The advantages of the proposed system is that it can operate in a real-time manner and can handle complex geometry and material behavior such as soft human tissues.

The new approaches are demonstrated to significantly improve speed and generality of deformable material haptic simulation model. Rapid simulation featuring multi points of interaction with deformable objects allows future applications in skill transfer, remote operation, fast product prototyping and data visualization.
Acronyms

ANN  artificial neural network
BEM  boundary element method
CSG  constructive solid geometry
CT   computational tomography
DoF  degrees of freedom
ERD  entity relationship diagram
FDM  finite difference method
FEA  finite element analysis
FEM  finite element methods
FFD  free form deformation
FPS  frame per second
FVM  finite volume method
GA   genetic algorithms
LMA  linear modal analysis
MSE  mean square error
MSS  mass spring systems
PCA  principle component analysis
PUM  partition of unity method
RBF  radial basis functions
RoI  region of interest
SA   simulated annealing
SPH  smoothed particle hydrodynamics
SVM  support vector machine
WEKA Wakaito environment for knowledge acquisition
XFEM Extended finite element analysis
List of Publications


Chapter 1

Introduction

The simulation of haptic interactions with deformable models in virtual environments complements the user experience within the virtual space. Robles-De-La-Torre [1] demonstrated that the addition of the sense of touch provide more information about the environment objects and their characteristics. This extra information when paired with graphics and audio create a space that is very close to reality. The more the realism, the more the user can perform the desired tasks with ease and in an efficient manner.

There exist a lot of applications for such immersive environments such as the list compiled by Saddik [2]. Examples include, but are not limited to, medical training, military robotics, education and entertainment. Some applications can utilize the haptic sense and transfer the force to places that no human can reach such as a radioactive cell or a different planet. Other applications enrich the skills transfer process and make training actions recordable and can be replayed as many times as required without extra cost. This reduces the cost of training by introducing non-destructible environment and allows remote connectivity to go for advanced levels of quality.
Data-driven simulation methods of haptic feedback emerged to overcome the limitations of the parametric methods. Parametric methods suffer primarily from an inability to easily run in real-time mode and the difficulty of modeling complex objects and nonlinear material behavior. The haptic loop demands a refresh rate of approximately 1 kHz to run smoothly while a graphics loop would need between 30 to 60 Hz [2]. Bickel et al. [3] identified that the material behavior is usually nonlinear for realistic scenarios such as the simulation of human tissues in the medical domain. Thus, data-driven methods emerged to handle these situations where high processing power is not available or the underlying material behavior is hard to model. In data-driven methods, the required processing is divided into two phases: offline and online. Empirical data are collected offline to train a learning algorithm, which can calculate the desired outputs in the runtime. This reduces the number of calculations during runtime and enables the ability to handle complex scenarios.

1.1 Challenges in data-driven modeling

There are challenges in the data-driven methodology when simulating haptic feedback. The methodology is not intended to work where parametric methods are sufficient, but it needs to handle complex situations and still be efficient and competitive. The current challenges can be broadly classified into: challenges with the data acquisition process; challenges with the learning model training and storage; and challenges with the re-use of the data or generalization to avoid or minimize the re-building of models.
1.1.1 Data acquisition

Data acquisition is the first phase of building the model. The more efficient the collected data, the more the usability of the whole system.

Data acquisition time The first challenge is the time to collect data. Without proper planning, this can be a time consuming process. The number of data to be collected is too large as for every point defined on the underlying object, there are many scenarios of interactions by varying the force magnitude and orientation.

Adaptive selection of data The other challenge in the data acquisition is the complexity of the objects. Apart from simple primitive objects such as cuboids, cylinders, or spheres, the object needs preprocessing before the selection of the data. The data selection needs to be carefully done to maintain an acceptable error level while reducing the data size.

1.1.2 Model training and data storage

The selection of a learning algorithm is important to achieve better results. If the chosen learning algorithm is slow to use or not diverse enough, it will remove the benefits of the entire system.

Data storage size The collected data can be very large and the storage and retrieval processes might be inefficient. The data needs to be filtered or stored in a compressed manner. This is particularly significant in the visual rendering process. For a dynamic simulation, the object vertices positions have to be recorded for every time step. This, with the added data needed for the haptic rendering, has a large size that needs to be reduced or approximated.
Learning technique selection  There are many learning methods, but not all can be applied to this domain as demonstrated by Breiman [4]. The problem is non-linear and has complex interactions among features. The algorithms based on linear functions such as support vector machine (SVM) will not perform as well as artificial neural networks ANN for instance. The selection decision needs to consider whether to use a single learning technique or an ensemble of multiple ones. Another decision needs to be on the data pre-processing on whether to reduce the data dimensionality or not. The data dimensions reduction directly reduces the training time and the storage requirements. This is not the only benefit, but this reduction can lead to a better learning by increasing the model diversity by choosing the high contributing features.

1.1.3 Data generalization

The collection of data and model creation requires effort and time. The inability to re-use the data is one of the main features that discourage the use of data-driven models when comparing with the parametric ones.

Multiple points of contact  A main area of generalization is the extension of the data-driven modeling to multiple points of interaction scenarios. This needs a model that can handle all the external interactions and their mutual effects. The extension to multi-points interactions is very useful to simulate real life tools and the advances in multi-point haptic hardware.

Deformation transfer to similar geometry  This is useful for the cases when a modeled instance needs to be related to its class of objects, such as different human livers, where they differ slightly in shape. Deformation transfer will save a lot of time if the same dataset or part of it can be used for a family
of related objects.

1.2 Research contribution

The thesis addresses the challenges identified in the preceding sections through the proposal of heuristics and models that can overcome the mentioned challenges. This lays the foundation of moving the data-driven methods from theory to practical applications and increases their possible application domain as well. The contributions of this work are:

Data Selection A set of heuristics and algorithms are proposed to guide the data selection. The selection is done on the discretized object vertices and the external applied forces. The algorithms are adaptive with the targeted level of accuracy and they can work on any object topology.

Learning Model Selection Different supervised learning techniques are tested and their performance is benchmarked. Feature reduction is used to reduce the data size and improve the trained model diversity.

Data Generalization A data-driven model is introduced that can handle multiple points of interaction. The model captures the external applied forces and the mutual effects among them.

Multi-point System Implementation A real time multi-point enabled haptic and visual rendering system is introduced. This combines the advances of hardware and software optimization in physically-based manner and with plausible speed rates.

Previous research has focused essentially on the hardware tools used for the data collection [5][6] and little attention has been directed towards the data manipulation
and usage. Without the proper planning of the data collection and usage the data-driven methods will not be able to compete against parametric methods and will lose their advantage of handling complex objects and not only a simple surface or single fixed point of interaction. The main target of this thesis is to improve the data-driven methods abilities in simulating diverse complex scenarios.

1.3 Thesis organization

This thesis is organized to follow the sequence of the data-driven process phases and show the optimization in each individually. Thus, it is organized into the following chapters:

Chapter 2 Presents an overview of the haptic rendering process, rendering techniques of deformable models, and data-driven methodology. The haptic rendering process and the difference between it and graphics rendering are described. A literature review of the important rendering techniques and their pros and cons follows. This discussion lays the foundation on why data-driven methods are promising. Finally, a zoom on the data-driven methodology, related literature, comparison with the corresponding parametric methods, and the challenges that need to be tackled to improve the methodology usage and potential is introduced.

Chapter 3 This chapter targets the optimization of the data collection phase. A set of proposed adaptive methods are introduced to reduce the data volume while keeping accuracy. This enables the handling of complex objects with irregular shapes and material behavior. These methods are then tested with regular and irregular objects in a simulation environment.
Chapter 4  Introduces a comparison between supervised learning methods and the effect of feature selection on the model quality. This gives an insight of the effect of the chosen techniques on the simulation performance. A comparative study was made also to show whether to preprocess the data with feature reduction techniques. The conclusions from this chapter help in the learning method selection and data formatting.

Chapter 5  Discusses how to generalize the data-driven methods to extended scenarios. It introduces a model for multiple points of interactions. This model handles the external loads effect on the object and the mutual effect between the loads. The design is simulated and shows an acceptable error margin between 1% and 3% between the calculated values and the actual recordings.

Chapter 6  Here an implementation is provided of the optimization heuristics in a multi-point interaction system. The system uses a novel gripper system that can interact with deformable models accurately and in a real-time manner. The system utilizes the proposed methods from the other chapters and collects the data samples using physical objects and sensors.

Chapter 7  Presents the main conclusions of this research, potential applications, and indicates the future research directions of the presented work.
Chapter 2

Literature review

Haptic simulations provide the user with more information that facilitates efficient interaction with the underlying virtual objects. The extra information can enhance the user perception abilities or in some cases can be the only source where other visual and audio senses are not applicable. Therefore, haptics is currently found in several areas such as military training, medical training, teleoperation and many daily used products like mobile phones and cars. A typical haptic system consists of three components: human, hardware, software. The software part, which is the focus of this work tries to simulate the user interaction with the virtual environment and calculate the appropriate force feedback to be transferred by the hardware to the user. This simulation needs to be dynamic, realistic and at interactive rates as demonstrated by Saddik [2] to be meaningful and stable to the user.

2.1 Haptic rendering

Haptic rendering of the feedback force to the user is a multi-stage process. As shown in Figure 2.1, the process has two loops: haptic and graphics. Each process has a module or more as follows:
1. Haptic device: this is the interface between the user and the virtual environment. Through it the user moves the device proxy or representative in the virtual space and calculates the force feeds when collisions happen with the virtual objects. There exist several commercial models that vary in design and their abilities of handling force ranges. Examples are shown in Figure 2.2. Two of them are entry level quality and one is premium.

2. Collision detection module: this module is responsible of defining the collision model and detecting the collisions between the haptic device proxy and virtual objects. Examples of collision models are bounding volume hierarchies, stochastic methods, distance fields, spatial subdivision, and imagespace techniques. A more details about collision detection algorithms literature can be found in the detailed survey by Teschner et al. [7]

3. Visual rendering module: this module is responsible for computing the visual deformation or displacement of the objects in case of deformable or rigid objects respectively.

4. Force rendering module: this complements the visual rendering module and calculates the feed force from the elements of the objects and return an augmented force that can be propagated to the user.
Figure 2.1: Haptic process components and data flow. The proxy is the representation of the haptic device in the virtual environment.
Figure 2.2: Examples of commercial haptic devices that have only one point of contact.

The haptic rendering process has higher requirements than graphics rendering. Salisbury et al. [10] demonstrated that the haptic loop usually requires 1 kHz refresh rate while graphics loop requires 30 to 60 Hz. The haptic rendering needs to be realistic and help the user perceive the virtual scene. Thus, the use of physically loyal methods in the calculations of the force feedback and the visual changes is desirable and produce an efficient system. By physically loyal it is meant that the force and visual calculations follow the constitutive laws of physics. This leads to
the need of heavy calculations and with the requirement of real time performance, the system needs to have a trade-off between these contradictory goals to satisfy the application domain needs.

2.2 Haptic rendering of deformable models

Haptic interaction can be with either rigid objects to feel the curvature and the friction or with deformable models to feel the stiffness of the object material. For rigid objects, the challenge is in detecting collisions and the main research focus is on handling large number of objects with high details, as demonstrated by the work of Moustakas et al. [11]. However, for deformable objects, deformation rendering is also needed to reflect the material properties as the force field is not constant like the rigid case.

2.2.1 Modeling approaches

The deformation function $def(O,f)$ transforms the object $O$ to $O' = def(O,f)$, as shown in Figure 2.3. In dynamic simulation, the difference between each time step is the change in object topology and stress values based on the applied external force or load.
Figure 2.3: Object deformation model where $O' = def(O, f)$.

There are many techniques that are used for deformable model simulation. These techniques can be classified broadly into two categories: physically-based and non-physically-based techniques. The difference between the two categories is that physically-based techniques use physical principles and constitutive laws in computing deformations while the other category techniques focus more on plausible visual simulations.

Excellent detailed surveys by Gibson and Mirtich [12] and Nealen et al. [13] have considered the topic from computer graphics point of view with a focus on the visual rendering loop. Figure 2.4 shows the common used techniques in the literature for modeling deformable models. The figure shows the position of the data-driven methods in the literature classification. Also a complete list of the basic general equations of elasticity theory is addressed in [14] by Love. Other introductions exist in [15] by Baraff and in books such as [16, 17, 18, 19, 20].
Figure 2.4: Modeling approaches of deformable models.
Non-physical deformable modeling

Non-physical modeling techniques produce visually plausible results. An example of this category is shown in Figure 2.5. The skeleton, which is a 1D structure, [21] of the 3D mesh is calculated. Each line of the skeleton is then mapped to a B-spline curve [22] that is controlled by certain control points. The deformation of the 3D mesh in this case is reduced largely and interactive refresh rates can be achieved easily. However, this suffers from many disadvantages. Aside from the geometry reduction and even if the 3D mesh was originally a tube or a cable, the method is not convergent. Nealen et al. [13] showed that the mapping between this method and physical laws is not easy and large deformations will produce unrealistic behavior.

Figure 2.5: Non physical modeling example, where 3D mesh is reduced to a skeleton and the skeleton is controlled by selected control points.

There are many other methods for non-physical modeling. Examples include
free form deformation (FFD), which is a fast algorithm where a coarse mesh is controlling a finer one [23, 24, 25, 26]. T-splines that are non-uniform B-spline surfaces with T-junctions [27, 28] can control a mesh with defined points. Others also include space warping [29, 30], differential surface properties [31, 32, 33, 34, 35, 36], and implicit functions [37, 38]. All of these methods suffer from the difficulty in relating the model to constitutive laws of physics and thus are used more in applications that the physical loyalty is not a high priority.

Physical deformable modeling

For haptic applications, physically-based techniques are preferred because they can better describe an object behavior and hence realistic experience. The physically-based methods again can be broadly classified into parametric methods and data-driven ones. The parametric methods build an explicit model that is defined through a set of parameters. The more complex the model, the more parameters and number of relating equations required. Thus, parametric methods has proven its capability in linear simulations, as demonstrated by Müller and Gross [39] but still it is not quick enough for nonlinear real-time applications, as shown by Zhong et al. [40]. On the other hand, data-driven methods do not have an explicit model. The model is defined through training using empirical data and a suitable learning algorithm.

The main parametric physically-based methods are Lagrangian [41] and Eulerian [42] methodologies. The difference between these methods is shown in Figure 2.6 [20]. The Lagrangian mesh is drawn on the object and the object nodes and material change but their mutual relative position remains the same. An alternative approach is the Eulerian mesh the nodes do not move while the material position changes. Thus, the mutual relative position between the nodes and the material is variable.
Lagrangian methods for deformable objects simulation can be classified into mesh-based methods and mesh free ones. Mesh-based approaches that are frequently used include finite element methods FEM [43, 44, 45], finite difference method (FDM) [46, 47], and finite volume method (FVM) [48]. These methods all use weighted residuals as follows:

\[ u = \sum_{i=1}^{n} a_i \varphi_i, \quad \tag{2.2.1} \]

where \( a_i \) is element node displacement and \( \varphi_i \) is an element shape function. The
difference between the three methods is that FEM uses Galerkin [49] shape function, FDM uses Dirac delta function, and FVM uses a uniform shape function. The boundary element method (BEM) [50, 51, 52] is a useful alternative to the standard FEM because all computations are done on the boundary of the object instead of on its volume. The Extended finite element analysis (XFEM) [53] or partition of unity method (PUM) is considered in cases of cracks and discontinuities. Other than these continuum mechanics methods, there are mass spring systems (MSS) [54, 55, 56], which are fast and easy to implement but not convergent.

The second set of Lagrangian approaches are the mesh-free ones [57]. These methods include loosely coupled methods, smoothed particle hydrodynamics (SPH), and point-based methods. Loosely coupled methods [58, 59, 60], which use graphical primitives such as points or spheres, are suitable for fuzzy objects, which have variable boundaries, like sand and clouds. SPH methods [61, 62, 63] are usually used to simulate fluid flows because the coordinates move with the fluid and the resolution of the method can easily be adjusted with respect to variables. Point-based methods [64, 65, 66] are more suitable for animation and non-realistic materials such as elastic fluids. The drawback with mesh-free methods is the increased computational requirements due to the large number of components and the fact that the surface is not explicitly defined.

The Eulerian [67, 68, 69] method is more suitable for fluids and fuzzy objects. Carlson et al. [70] used this to simulate melting objects while Stam [71] used it to model smoke. A drawback in the Eulerian method is that the object boundary is not defined explicitly, as discussed by Nealen et al. [13].

There is a trade-off in haptic simulation between accuracy and speed because of the complexity of the simulation and the limitation of current methods. Considering the most commonly used parametric approach, which is a Lagrangian mesh-based
method, the two extreme approaches are FEM and MSS. FEM are accurate with respect to other methods when the material properties are known. However, FEM are time consuming and not suitable for real time applications. Besides, the material properties have to be estimated through empirical data. This prevents the FEM from being purely parametric. MSS is the opposite of FEM; as they are quick, easy to implement, and use much less parameters. However, their physical accuracy cannot be easily validated and most of them are not convergent [13]. Therefore, other techniques were introduced to handle the trade-off between these two extremes. Examples of these techniques are:

- Methods that improve the speed of FEM, as in the work by Kikuuwe et al. [72] or the accuracy of MSS such as the work of Corso et al. [73];

- Using MSS parameterized from FEM, such as the work by Etzmuss et al. [56], Bianchi et al. [74], and Morris [75];

- Limiting the user experience by providing a subspace or reducing the object movement with focus on important user requirements such as the work by James and Fatahalian [76] and the work by Barbič and James [77]. This enables using high accurate methods with respect to the application nature; and

- Using a learning technique, such as artificial neural networks ANN, as in the work of Morooka et al. [78] and Petriu et al. [79] to build a model that is based on training sets acquired from interactions with the underlying object.

Data-driven techniques always exist in the haptic simulation process. This can be either in the preliminary stages of defining material properties and feeding the parameters to a parametric method or to do both the tasks of material definition
and material behavior description. In either cases, the data-driven techniques has the ability to describe complex scenarios and to act with a higher speed than the corresponding parametric methods.

2.3 Data-driven methods

In this section an overview of the data-driven methodologies in haptic rendering is presented and its status compared to the corresponding parametric techniques. The data-driven methods are essential and unavoidable either as a part of the parametric methods or in a standalone mode. At the end of this section, the current limitations that decrease the possibility of favoring the standalone mode and what is the importance of removing or easing them are shown.

2.3.1 Data-driven model

Data-driven models define complex phenomena by collecting a set of observations regarding its behavior. This does not form an explicit model but helps in grasping the complexity and simulate it in a real-time manner. Data-driven modeling is common in many fields and is used extensively when the parametric or explicit solution is not feasible. Figure 2.7 shows the proposed data-driven modeling approach in haptics.
Figure 2.7: Data-driven modeling approach for haptic simulation of deformable models.

**Data-driven modeling steps**

Most data-driven models share common steps as identified by Harris and Hong [80]. The steps are:

- **Data Gathering**: This step includes using hardware equipment or simulation software to generate the data. In addition, the amount and sources of data are studied. For instance, all possible data can be considered [81], random amount of data [78] or a planned selection as provided in this thesis. The latter technique guarantees that each piece of the gathered data is contributing to the model and largely decrease the collection time.

- **Pre-Processing**: The collected data might not be ready to use as is in next steps due to noise or high dimensionality. In this step, data are processed and filtered according to the user requirements and needs.

- **Model Architecture Selection**: The data are used to construct a model. This model decides how the data will be manipulated. Usually the architecture is an interpolating mechanism such as ANN [82] or radial basis functions (RBF) [83].
• *Learning:* This leads to building an online quick system that can simulate the phenomena based on the observations. This should work for new behaviors and recorded ones as well.

• *Parametric Evaluation:* This include checking the accuracy, feasibility, and completeness.

### 2.3.2 Current data-driven methods in haptic rendering

The usage of data-driven methods to describe the material behavior emerged in recent years to handle complex scenarios and to satisfy the real-time performance requirement. Previous research in data-driven modeling can be classified into two categories. Firstly, research that experimented to estimate $K$ in the equation such as the work by Pai et al. [84]

$$
\mathbf{u} = \mathbf{Kp},
$$

(2.3.1)

where $\mathbf{u}$ is the vector of vertices displacements, $\mathbf{K}$ is the stiffness matrix and $\mathbf{p}$ is the applied external forces vector. If some of the vertices are fixed the $\mathbf{p}$ vector will contain force feedback. They use physical and simulation techniques to help in estimation. Other researchers such as Krysl et al. [85] focused on reducing the space of movement of the 3D object or considered only a subset of the available degrees of freedom DoF. They also used pre-computed data to determine the choice of the subset.

Both categories focus on the utilization of the training data to create a direct calculation method. The number of data or the selection criteria was not given much attention. Mostly the data usage was limited to specific object and for static simulation purposes. The employed equipments or the mapping of the data to architecture
were the main focus. For the actual extraction of the data some used random methods or tried all vertices in specific discretization. Thus, in this thesis new heuristics on the selection basis are introduced that can help in reducing the training time, improve coverage of all cases, enable increased usage of the collected data and can be adapted to user needs and region of interest (RoI).

Building a direct deformation model

Pai et al. [84] describes how to capture observations of a deforming object to be able to estimate a deformation mode using linear elasto-static models. Their approach is to estimate $u$ in equation 2.3.1 using the observations they acquire from physically interacting with the object and used the least square method for estimation. The authors did not consider how many samples are sufficient for estimation, but noted that they require redundant images. Other researchers have tried to improve Pai’s by introducing neural network techniques, as in the work of Cretu et al. [86], to decrease number of the points used. Morooka et al. [78] introduced an ANN simulator for FEM. They use random samples of an arbitrary size to train their network. A similar approach to Morooka was adopted by Deo and De [87] that is called PhyNeSS but used RBF and considered all the nodes instead. Some of the recent work that covered many areas of the acquisition process is by Fong [81] and Hoever et al. [83]. Most of the research in this area is directed towards the equipment used to obtain the data and the interpolation mechanisms of the collected data.

Model reduction

This category of research focuses on reducing the DoF of the 3D object to enable faster utilization of differential equation methods such as FEM. Most of the work follows the idea of model reduction through modal analysis as demonstrated by
Krysl et al. [85] and James and Fatahalian [76]. The most recent work is by Barbič and James [77], where they introduce two methods for obtaining the training data. They introduce an interactive method based on the user feedback and another automatic way based on linear modal analysis (LMA) that extracts the modal basis vectors of an object.

2.3.3 Current data-driven modeling vs. parametric modeling

There is no pure parametric method that can simulate the interactions with arbitrary deformable models. If the material parameters are not known (e.g. Young’s modulus and Poisson ratio), then empirical data are required to deduct these parameters values such as the work by Morris [75]. An example is calculating stress $\sigma$ values in FEM for elastic materials:

$$
\sigma = \begin{bmatrix}
\sigma_x \\
\sigma_y \\
\sigma_z \\
\sigma_{yz} \\
\sigma_{zx} \\
\sigma_{xy}
\end{bmatrix} = \begin{bmatrix}
\lambda + 2\mu & \lambda & \lambda & 0 & 0 & 0 \\
\lambda & \lambda + 2\mu & \lambda & 0 & 0 & 0 \\
\lambda & \lambda & \lambda + 2\mu & 0 & 0 & 0 \\
0 & 0 & 0 & \mu & 0 & 0 \\
0 & 0 & 0 & 0 & \mu & 0 \\
0 & 0 & 0 & 0 & 0 & \mu
\end{bmatrix} \begin{bmatrix}
\varepsilon_x \\
\varepsilon_y \\
\varepsilon_z \\
\gamma_{yz} \\
\gamma_{zx} \\
\gamma_{xy}
\end{bmatrix}
$$

(2.3.2)

Where

$$
\gamma_{yz} = \varepsilon_{yz} + \varepsilon_{zy} = 2\varepsilon_{yz}, \quad \gamma_{zx} = \varepsilon_{zx} + \varepsilon_{xz} = 2\varepsilon_{zx}, \quad \gamma_{xy} = \varepsilon_{xy} + \varepsilon_{yx} = 2\varepsilon_{xy},
$$

$$
\varepsilon_x = \varepsilon_{xx}, \quad \varepsilon_y = \varepsilon_{yy}, \quad \varepsilon_z = \varepsilon_{zz}
$$
and

\[
\lambda = \frac{E \nu}{(1 + \nu)(1 - 2\nu)} \quad (2.3.3)
\]

\[
\mu = \frac{E}{2(1 + \nu)} \quad (2.3.4)
\]

Where \( \nu \) is Poisson’s ratio, \( E \) is Young’s modulus, and \( \varepsilon \) is the strain.

This is especially the case when the material is heterogeneous or layered. Thus, parametric modeling process can be generally split into two phases: material representation and material behavior description. Only the material description phase can be done always using a parametric explicit model, as shown in Figure 2.8.

![Diagram](image.png)

Figure 2.8: Empirical data are used in calibrating the object material parameters in parametric modeling techniques.

In light of the previous sections, table 2.1 shows a comparison between the current data-driven techniques and the material behavior describing parametric techniques in haptic simulation of deformable objects. In the light of this comparison, the last three points are not enabled directly in the current developed techniques. This leads to limiting the usage of the data-driven methods in the simulation and even may make it not applicable.
Table 2.1: A comparison between current data-driven techniques and parametric techniques in haptic simulation of deformable objects.

<table>
<thead>
<tr>
<th>Table</th>
<th>Parametric methods</th>
<th>Literature data-driven methods</th>
<th>Favored for simulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model description</td>
<td>Explicit through parameters</td>
<td>Implicit through empirical data</td>
<td>N/A</td>
</tr>
<tr>
<td>Required online processing power</td>
<td>High</td>
<td>Low</td>
<td>Data-Driven</td>
</tr>
<tr>
<td>Required storage space</td>
<td>Low</td>
<td>High</td>
<td>Parametric</td>
</tr>
<tr>
<td>Interactivity</td>
<td>Variable point of contact &amp; Multiple points of contact</td>
<td>N/A</td>
<td>Parametric</td>
</tr>
<tr>
<td>Generalization</td>
<td>Available</td>
<td>N/A</td>
<td>Parametric</td>
</tr>
</tbody>
</table>

The focus of the current research is largely towards the collection equipment of the empirical data. The research in this direction is important to reduce the noise in the collection and to accurately capture the material properties and behavior. However, without planning, the collection will take a long time, without enabling high interactivity and data reuse the data will be recollected every time, and without storing the data effectively the storage requirement will not be applicable. In the following section, the main limitations of the current data-driven techniques are discussed.

### 2.3.4 Limitations in data-driven haptic rendering

The current data-driven methodology in haptic simulation of deformable models seems to have some limitations. These limitations hinders the dependability on these kind of modeling technique as a rival to the parametric methodology. The limitations can be listed as follows:
1. **Simulation of complex objects:** There are no straightforward methods defined in the literature that effectively allow the haptic manipulation of complex objects such as a human kidney or liver. Data collection needs to cover the object and handle as much points of contact as possible or locate the ones that can be interpolated for others.

2. **Lack of variable points of contact:** The training for a fixed point of contact is not enough because it assumes that the underlying object, that is dealt with, is a finite homogeneous 2D mesh object where all points are the same. This is not true in real world and limit the usage of the training data to maybe one usage, which is the ability to deferentitis between different materials. The ability to interact from different points enable the recognition of the material and the geometry and improve the overall user perception.

3. **Large training data size:** The collection methods in the literature suggested collecting the data randomly or collecting all data if possible. This obviously take long periods that can be days for a simple object [5] and leads to long training time as well. Without planning the data selection based on the object geometry and user needs, the collection process will be usually inefficient.

4. **High data storage requirements for complex objects:** The data that will be trained size can grow enormously. For each vertex of the object, there are many interaction scenarios with different external force magnitude and orientation. Besides, there can be simultaneous multiple points of contact. Thus, for complex objects, with large number of vertices and irregular geometries, a parameterized storage method is required to store a reduced version of the object. This needs to maintain the accuracy as well to preserve the original curvature and topology of the object.
5. **Machine learning model selection:** Hover et al. [88] identified the importance of choosing an appropriate learning method as some learning algorithms might not be applicable and others might need modifications and tuning. The preprocessing of the data can also help in improving the performance by reducing the model bias [89].

6. **Multi-point interaction:** There is a growing trend in the development of multi-point haptic devices. Examples include a gripper device developed by Najdovski and Nahavandi [90] and five fingered hand device developed by Endo et al. [91] as shown in Figure 2.9. The data-driven methods need to enable such kind of interactivity to improve the user experience and keep up with the new haptic devices design.

7. **The re-use of the data:** The collection and processing of data is a long process. It will be more practical if the same data can be reused for similar related objects with minimal recollection and reprocessing through the identification of the differences between the objects.

(a) Two points of contact gripper [92].  
(b) Five points of contact hand [91].

**Figure 2.9:** Multi point haptic device examples.
2.4 Conclusion

There are several methods that are used in simulating haptic interactions with deformable models. Only physically-based methods can be considered to achieve accuracy. These physically-based methods can be classified broadly into parametric and data-driven ones.

The data-driven haptic rendering of interactions with deformable models is a powerful technique. It has many advantages over the corresponding parametric methods specially in speed and the ability to model complex objects and material behavior. Besides, the accurate parametric techniques such as FEM need empirical data to estimate the parameters which make the data-driven methodology a basic component of them.

Existing data-driven methods suffer from limitations, such as inefficient data collection processes, the inability to capture global deformations and to simulate complex interaction scenarios. These limitations prevent the usage of the data-driven methods in real life applications and limit it only to simple scenarios.

The objective of this thesis is to address these limitation through theoretical contributions and move the data-driven technique to a practical level that can be used in different applications such as medical training and web and mobile haptics. This will make the data-driven modeling tool available to the users with all its features enabled, which will widen their choice and enable hybrid solutions to take place in the future.
Chapter 3

Optimizing the data collection phase of the data-driven simulation

3.1 Overview

This chapter targets the data collection phase of the data-driven modeling process. This is one of the most important phases of the modeling process. The quality of the collected data determines the accuracy of the resultant model when compared with offline methods. There are two components of this phase: the collecting tool whether it is hardware or software and the selection algorithm that plans the collection process. The collection tool has been the primary focus of researchers for the last decade and it still needs further research. The problem with the recording tools is how to capture the deformation of the whole object. This is challenging as each node needs to be observed and the existence of occlusions due to complex geometry. The other component, which has been neglected is the planning of the collection phase. The planning needs to clearly sort the data sources and decide on which items to consider and which to neglect to ensure a maximized contribution
with a small sample size.

### 3.1.1 The need for high volume of data

For complex objects that have large number of vertices and irregular geometry, the size of the empirical data that needs to be fed to the learning algorithm can grow in a dramatic way. Given a 3D object $O$ and an external force $F$, the static deformation function $\bar{O} = def(O,F)$. $\bar{O}$ is the deformed object that resulted from applying $F$ over $O$. In reality, 3D objects are continuous and $F$ can be of any magnitude and direction. Thus, an infinite number of possible $\bar{O}s$ exists. For haptic simulation and interactive applications, the object is usually discretized into a set of primitives (e.g. triangles), which are called faces, in order to reduce the number of interactions. Let $O = \{V,F\}$, where $V$ and $F$ are the sets of vertices and faces respectively, and $N_\nu$ is the number of vertices of $O$. To compute the data of one instance of forces for all vertices, $N_\nu$ data sets are needed. For coverage of more than one force magnitude and more force directions in an admittance-based rendering (i.e. the force is provided to the system), the following equation will represent the total data size required for haptic rendering:

\[
S_{\text{haptic}} = N_\nu \times N_{\text{mag}} \times N_{\text{dir}}
\]  

(3.1.1)

$S$ is the data size and is the main parameter of the cost function of the data collection process. $N_{\text{mag}}$ and $N_{\text{dir}}$ are the number of the required magnitudes and directions of the applied external force. The data size depends heavily on the object size and the required data coverage.

In addition to the haptic simulation, the deformable object can be also simulated visually. In order to do that, new positions of the object vertices need to be identified. In a dynamic simulation environment with multiple deformation functions
\(def_i\) through time:

\[
O_n = def_{n-1}(O_{n-1}, F_{n-1}) = def_{n-1}(def_{n-2}(O_{n-2}, F_{n-2}))
\]

\[
= ... = def_{n-1}(...def_1(O_1, F_1)...)
\] (3.1.2)

The size of the data dedicated for the visual simulation will then be multiplied by the number of the time steps \(N_\Delta\) as follows:

\[
S_{visual} = V_O \times N_\Delta
\] (3.1.3)

Where \(V_O\) is the number of vertices of the deformable object.

The dynamic simulation case can be represented in an entity relationship diagram ERD, as shown in Figure 3.1. Here, only forces are considered and not torques as for simple actions like poking (single point) or grasping (double points) they have no much effect.

Figure 3.1: The dynamic simulation data represented in an ERD.
Without proper planning for the data collection, the process of building the data-based models will be very time consuming. In addition, there are some disadvantages that limit the further usage of the generated data-driven models in practice such as:

- The handling of complex objects such as human body parts;
- The retrieval of the data might be slow and prevent the advantages of data-based modeling speed; and
- The limited portability of the data over networks and limited storage media.

In the following sections, new techniques of data collection methods are proposed. This is done by getting more information about the underlying 3D object and the external force.

### 3.1.2 Parameterized storage of curves

Parameterized storage of objects is vital for large objects where limited transmission capabilities are used or small storage media are only available. The transformation from raster format to a vector one provides the ability to have the information required for haptic and visual rendering in a file that can be portable. By acquiring extra information about the underlying object and the external forces, a parameterized representation of the object can be stored in a reduced format. As will be shown in the proposed algorithms, the data will be discretized in a way that allows a vector storage format. For visual simulation, rather than storing the positions of all vertices, a set of vectorial planes is stored instead. For each plane all the required is the plane equation, dimensions of the best enclosing rectangle, and a set of missing vertices from the rectangle. This enables dynamic reduction of the data according
to the available space and minimizes the error between the restored versions and the original one. Figure 3.2 shows an example of the parameterized storage. The bigger the size of the planes, the less data required to present it.

Figure 3.2: Parameterized storage of a 3D liver object into a set of planes.

### 3.2 Related data collection methodologies

The haptic simulation using empirical data literature focused on the equipment and the learning technique to be used to make use of the collected data. This work lies in between these two topics. The target is to guide the data collection to enable the handling of complex objects and faster training and simulation of the learning technique.

Pai et al. [84] describe how to capture observations of a deforming object to be able to estimate a deformation mode using linear elasto-static models. Their approach is to estimate the displacements vector $u$ in equation $Ku = p$, where $K$ is the stiffness matrix and $p$ is the force vector, using the observations they acquire from
physically interacting with the object. The least square method was used for estimation. Other work involving the design of collection equipment were introduced by Fong [81] and Hoever et al. [83]. The data collection process in these approaches lacked the planning phase and point selection algorithms were not clear which led to a prolonged data collection process.

Other researchers focused on the usage of the collected data. Some introduced neural network techniques, as in [86], to decrease number of the points used. Morooka et al. [78] introduce artificial neural networks ANN simulator for finite element methods FEM. They use random samples of an arbitrary size to train their network. Also radial basis functions RBF were used by Hoever et al. [83] and Deo and De [87].

The planning of the data collection has been nearly absent in the literature. The most common used methods were to consider a fixed number random data [78] or to limit the scenarios to a certain set and collect all the elements of this set [87]. The collection time in [87] for one scenario had an average of 29.17 sec for an object with 734. With a number of scenarios equal to 17940 the approximate collection time for all scenarios is 6 days, which is a relatively long period for such number of nodes.

### 3.3 Proposed data collection techniques

In this section the data collection techniques that provide minimal efficient set of data are provided. The process of interacting with deformable models has two main components, the deformable object itself and the applied external force. The external force has a direction and a magnitude. The object model is represented as a 3D
mesh composed of a set of vertices and faces. To represent real materials behavior, the object is described in a parametric format or using data driven models. In parametric descriptions, the object is assigned a set of equations to govern its motion and response to the external forces. These equations correspond to the desired material properties. Examples of the material properties are:

- Homogeneous, which implies all vertices have the same properties, or the opposite, which is heterogeneous;

- Isotropic, which implies vertices move in the same way in all directions and in this case only two parameters are needed to define this property, namely the Young’s modulus and the Poisson ratio. Conversely, the material can be anisotropic and in that case the number of required parameters will be much more after considering symmetry and other reductions; and

- Linear, which implies the relation between the stress and the strain, is linear. The opposite is nonlinear and in most cases it is the most realistic relation.

Using data models that are not explicit describes the model based on trials and simulations. The proposed techniques through different iterations will attempt to automatically propose a set of training data for an input object. The input object will have known material properties as stated above. In all stages the algorithm will use FEM software to define experiment responses. This can also be done physically if certain equipment is available for measurements. A description of the algorithm flowchart is shown in Figure 3.3. The assumptions of the algorithm are:

- Some faces/vertices are always fixed so they show no movement and this is an input to the program; and
• The collection of the data will be done via simulation using FEM. Otherwise, the material properties is not needed if the data acquiring phase needs to be done physically.

Figure 3.3: Flowchart of the data collection phase of the data-driven modeling approach for deformable models.
3.3.1 Stage 1 - Object segmentation

In the first stage, the algorithm defines vertices that lead to distinguished deformation shapes by segmenting the object outer shell into flat surfaces. The next stage tries a set of force magnitudes for each surface in both pure pushing and pure shearing directions to compute the certain forces that are required for training. Finally, the combination of the results from the previous stages produces a set of training data that can be fed to a learning module such as ANN to form a black box that compute deformations instantly in real time.

The 3D objects that are provided to the algorithm are boundary objects. This means they have an outer mesh only and are not made of voxels as in the case of 3D volumetric meshes. The segmentation process aims to split the input mesh into a set of sub meshes. The underlying object can be a flat surface, a box or a complex object like a human organ. The simple topology of a flat surface enables a more straightforward training process and natural resemblance in behavior between the flat surface vertices. Thus, linear interpolation techniques produce fewer errors because all vertices are nearly sharing one plane.

Using this fact, the first step in the algorithm is to segment the object or the input mesh into set of surfaces that are as flat as possible. The input to the segmentation algorithm is a 3D mesh and a threshold angle. The threshold is calculated considering the available resources and the required accuracy. The algorithm uses a greedy growing region technique, which uses an iterative approach where in each iteration; an unclassified vertex (seed) is chosen to build a surface around it. The results are a set of surfaces that share the same criteria. Better results can be achieved using more sophisticated segmentation techniques but will need more computation time. A recent survey by Shamir [93] lists the latest advances in segmentation methodologies. Figure 3.4 shows results of segmenting a liver. The output of the algorithm
is a set of surfaces that are as flat as possible based on the provided threshold.

Assuming the 3D object \( O \) is defined as \( \{V, F\} \) with vertices \( V \) and faces \( F \) can be defined as a set of triangles or rectangles or any other form. The target is to generate \( n \) surfaces \( \{S_1, S_2, \ldots, S_n\} \) that are disjointed and subsets of \( O \). The criteria for the segmentation process are the vertex normal variation. For a vertex \( V_i \) with normal \( N_i \) to be segmented into a surface \( S_i \) that started with a seed vertex \( V_{seed} \), the angle between \( N_i \) and \( N_{seed} \) should be less than \( \theta_{threshold} \). Here \( N_{seed} \) is the normal of the vertex \( V_{seed} \) and \( \theta_{threshold} \) is the threshold given to the algorithm.

![3D liver mesh and liver after segmentation](image)

(a) The 3D liver mesh  
(b) Liver after segmentation

Figure 3.4: Segmenting 3D liver mesh into flat surfaces.

### 3.3.2 Stage 2 - Points selection from segments

The output of the segmentation process is a set of near flat surfaces. The next task will be to select certain points from each surface in order to decrease the number of training sets. The selection process chooses the vertices that have large difference in the resulting deformation topology in order to improve the interpolation results. Thus, all the boundary points are selected and the approximate middle point of the plane is added, as shown in Figure 3.5. This reduce the number of data from \( n \) to
Algorithm 1 Segmentation algorithm

1: procedure SEGMENT($O$, $\theta_{\text{threshold}}$) 
2:     \[ \triangleright O \text{ is a 3D rectangular mesh} \]
3:     for all $v \in O$ do 
4:         Add $v$ to nonClsVts 
5:     end for 
6:     while \( \text{count}(\text{nonClsVts}) \neq 0 \) do 
7:         $v_{\text{curr}} \leftarrow \text{nonClsVts}[0]$ 
8:         $S_{\text{curr}}(v) \leftarrow \text{new surface}$ 
9:         Add $S_{\text{curr}}(v)$ to surfaces 
10:        $N_{v_{\text{curr}}} \leftarrow \text{calcNormal}(v_{\text{curr}})$ 
11:        for all $v_n \in \text{nonClsVts}$ do 
12:            $N_{v_n} \leftarrow \text{calcNormal}(v_n)$ 
13:            $\theta \leftarrow \text{getAngle}(N_{v_{\text{curr}}}, N_{v_n})$ 
14:            if $\theta < \theta_{\text{threshold}}$ then 
15:                Add $v_n$ to $S_{\text{curr}}(v)$ 
16:                Remove $v_n$ from nonClsVts 
17:            end if 
18:        end for 
19:     end while 
20: Split discontinuous surfaces 
21: return surfaces 
22: end procedure
approximately $\frac{n}{2}$ where $n$ is the number of object vertices.

Figure 3.5: Selection of vertices (orange are selected) reduce the volume of data to nearly half.

### 3.3.3 Stage 3 - Force effect linearization

Barbič [94] demonstrated that the external force effects on materials are generally non-linear especially for large deformations. In order to obtain a good approximation of the applied external forces, the range of the force magnitude needs to be piecewise linearized.

Suppose that the force range is from $f_1$ to $f_2$, a binary search like technique can be used to divide the nonlinear curve to a set of linear regions. This is valid, assuming that the underlying material is homogenous and isotropic. This enables the data to be collected using only one vertex, generalizing the behavior for others. The assumption of homogeneity can be relaxed if the object is piecewise homogeneous too. This technique uses the advantage of the fact that the nonlinear curve is
monotonically increasing.

The other component of the force is the force direction. Using spherical polar coordinates any force has an angle $\theta$ that ranges from zero to 180 degrees and $\phi$ angle. The force can be transformed with projection into the basic three dimensions. The forces are namely the shear forces and the push forces that are normal to the vertex. In order to handle the infinite possible force directions, incoming forces to the orthogonal three force directions are projected as follows:

\[
F_x = r \sin \theta \cos \phi,
\]
\[
F_y = r \sin \theta \sin \phi,
\]
\[
F_z = r \cos \theta
\]  

Thus, the force linearization algorithm, shown in algorithm 2, is performed three times to capture the force effects in all the three cartesian dimensions.

---

**Algorithm 2** Force linearisation

1: procedure LINEARISE($F_{\min}, F_{\max}, Dir, Err_{threshold}$)
2:   ▷ Direction(Dir) can be X, Y or Z
3:      $F_1 = (F_{\max} - F_{\min})/4$
4:      $F_2 = 3 \times (F_{\max} - F_{\min})/4$
5:      line = linRegressEffects($F_1, F_2, F_{\min}, F_{\max}$)
6:      $Err = maxError(line, F_1, F_2, F_{\min}, F_{\max})$
7:      if $Err < Err_{threshold}$ then
8:          Add $[F_{\min}, F_{\max}]$ to Ranges
9:      else
10:          Add Linearise($F_{\min}, F_1$) to Ranges
11:          Add Linearise($F_1, F_2$) to Ranges
12:          Add Linearise($F_2, F_{\max}$) to Ranges
13:      end if
14:  return Ranges
15: end procedure

---
3.4 Simulation results of the proposed data collection methods

This section describes stage 4 where the validation process takes place. The usefulness of the proposed techniques is demonstrated by applying them to an experimental 3D object and an example of a learning technique, which is ANN. The results are obtained using a simulation environment to compute quick training data without the need for dedicated hardware but the same idea is applicable if physical hardware is to be considered as in [81]. For the learning process the Matlab toolbox of ANN was used. Any other learning method could be used as the proposed techniques are general for any data usage method. The ANN $N = (W, T)$ has the advantage of being always convergent to a stable state [95] [96] where the matrix $W$ is symmetric and the element of its diagonal are nonnegative. Figure 3.6 shows 3D liver and 3D ellipsoid that were used in simulation. The chosen objects represent examples of a regular and an irregular geometry. The experiment shows that the performance of the ANN and its validation error is marginally affected by reducing the training data using the proposed techniques.
Figure 3.6: 3D object simulations. The region at the base of the object (green) represents constraints to fix vertices. The singular arrows represent an external force.

The employed ANN shown in Figure 3.7 has three layers; one for input and contained five neurons. Inputs are the contact point, the magnitude of the force and its direction along one of the main three axes. The hidden layer is chosen to be the default of 20 neurons as the tool suggested. The last layer is the output layer that has $3n$ neurons given that the 3D object contains $n$ vertices. The neurons in the last layer represent the displacement of the 3D object vertices on the three dimensions ($x$, $y$, and $z$). The ANN is a feed-forward back-propagation network that uses mean square error (MSE) as the performance function. This is used because it is direct to implement and converges, as proved by Bruck and Goodman [95]. There are three notes on the ANN:

- For complex objects with many vertices, the number of neurons in the output layer will be very large. A practical approach is to build more than one ANN that match the available resources given that the outputs are not correlated.
• The direction input is for the basic three axes. Thus, to capture any other orientation, the force direction is factorized into the three basic axes.

• The aforementioned ANN is for output displacements only and similar ones can be built for the force feedback.

![Diagram](image)

Figure 3.7: The configuration of neural network that is used in the simulation.

Increasing the $\theta_{threshold}$ value in the segmentation phase decreases the number of surfaces. Fewer surfaces reduce the processing time, but increase the interpolation error as the number of segments is inversely proportional to the average number of vertices per segment. This is shown in Figure 3.8. Figure 3.9 shows the relation between $\theta_{threshold}$, the number of surfaces and the average number of vertices per segment for the liver 3D object. The curves are not monotonically increasing or decreasing due to the greediness nature of the segmentation algorithm. This is more visible in the dashed curve as the Y-axis range is less than the other solid one.
Figure 3.8: The number of segments is inversely proportional to the average number of vertices per segment.
Figure 3.9: Different values of $\theta_{threshold}$ and the corresponding number of segments (solid green), average number of vertices per segment (dotted blue) and average number of selected vertices (dashed blue).
The results show that the performance of the neural network training was only slightly affected when considering a planned subset of the vertices rather than considering all of them. Figure 3.10 shows the difference in MSE of the ANN between considering all points in Figure 3.10a and using the proposed heuristics to select a subset of the vertices in Figure 3.10b. The comparison is done for $\theta$ value of 0.74 because, as shown in Figure 3.8, this value provides a balance between the number of segments and number vertices per segment.
Best Validation Performance is 4.4543 at epoch 9

(a) ANN Performance when all points of the surface are selected.

Best Validation Performance is 6.6362 at epoch 7

(b) ANN Performance when a subset of the points of the surface are selected using the proposed algorithm.

Figure 3.10: The performance of the ANN training of the 3D liver model. The difference in validation performance between choosing all vertices and selecting a subset is minimal when $\theta = 0.74$. 
For comparison between the performances on different $\theta$s, Figure 3.11 shows the difference in the validation error of the ANN in both cases. This shows that by planning the data collection, using the proposed algorithm the data collection time is decreased, whilst maintaining acceptable MSE. Besides, the convergence of the interpolation algorithm is faster due to the reduced training set size.
Figure 3.11: Comparison between the mean ANN validation error when choosing all points (dashed blue) vs. using a subset of points using the proposed algorithm (solid red).
3.5 Conclusion

In this chapter, a presentation of a new heuristic that can be utilized to auto generate deformation bases is provided. These bases can then be input into machine learning algorithms to simulate real time and accurate interactions with deformable objects. The advantages of the proposed planning techniques are that the data collection time is reduced and being adaptive with respect to user requirements. The techniques are independent of the data usage methods or the nature of the input 3D object. FEM and ANN were used as examples of a complete data-driven mode. The target of this work was toward static simulations. A natural extension is to check the possibilities of having similar heuristics for dynamic simulations. Generalizing for multi-point interaction given a set of single point interactions is interesting because multi-point interactions are very resource consuming with parametric models.
Chapter 4

Selection of suitable learning methods and data formats for the haptic interaction simulation

4.1 Overview

This chapter targets the problem of selecting an appropriate learning algorithm and how the data filtering and preprocessing can improve the model performance. Shall a single learning technique be used or an ensemble of multiple techniques? Shall the data be preprocessed or not? These questions are important for the quality of the training process. The model needs to be stable and flexible [4] in order to have a subset of all possible values that can be trained and generalized for possible scenarios. The smaller the subset needed, the faster the data collection and training and the less storage requirements.

This chapter provides a comparative study between different supervised learning methods to help in the selection of the appropriate method. This is done by
comparing the performance of different learning methods in the domain of multiple point haptic simulation of interactions with deformable models. This domain has the characteristic of large dimensionality and high volume of data as shown in the previous chapter. This is proportionally related to the number of contact points and proper selection of a learning method.

The components of the simulated system here are the deformable object and the external contacting tools. The simulation is done using an impedance-based rendering, which means that the displacement of the contacting tools is the input and the output is the haptic force feedback. Thus, the data that describe the system includes the external tools displacements $\mathbf{u}$, velocities $\mathbf{v}$, accelerations $\mathbf{a}$, and deformable object contact points $\mathbf{p}$ as an input. The output is the haptic force feeds $\mathbf{rf}$. All the features here are plural because the interactions are done through simultaneous multiple points of contact. It should be noted that each item of the data is a vector of three dimensions ($x, y, \text{and } z$). Figure 4.1 shows a typical learning algorithm diagram.
Figure 4.1: The diagram of the inputs and outputs of the learning algorithm. Each circle denotes a three dimensional vector.

The data size and features here are of a high volume and dimensionality nature. The size of the data $S$ can be calculated for two point of contact with a 3D object as follows:

$$S = N_p \times N_u \times N_v \times N_a$$  \hspace{1cm} (4.1.1)

Where $N_p$ is the the number of different points of contacts. $N_u$, $N_v$, and $N_a$ are the numbers of different displacements, velocities and accelerations of the external contacting tool. These are doubled as each parameter here is doubled for two points of contact and for each point there are three dimensions. The input features here are 24, as there are four parameters, two points of contact and three dimensions. These
are large numbers and for a higher number of contact points they will increase enormously.

The data collection can be done using an automated physical sensors or simulation environment. In this work, a simulation environment is used. The simulation is done using finite element methods FEM package (Abaqus™). The FEM is chosen due to its accuracy [13] and that the data collection will be made offline. The data is then fed to different learning techniques for performance comparison. Figure 4.2 shows a 3D object simulated in Abaqus™ with two external forces.

![3D FEM simulation in Abaqus™](image)

Figure 4.2: 3D FEM simulation in Abaqus™. Two external forces are applied on the deformable object.
4.2 Definitions and previous work

The haptic simulation of deformable objects problem has been addressed several times. To be able to group related work, a classification of the literature based on, which parts of the data-driven modeling process being addressed, is required. The data-driven modeling process has the following stages [80]:

1. Data Selection: This includes the process of what data to gather and what to neglect.

2. Data acquisition: This is where the data is collected using physical sensors or computer simulation.

3. Learning: This where the learning algorithm is trained using the collected data.

4. Evaluation: Checking performance against parametric methods or conducting user studies.

A summary of related work in the area of data-driven haptics is provided, followed by a review of the supervised learning techniques categories that will be in the comparison.

4.2.1 Data-driven haptics

Data selection and planning were the focus of [97]. The authors discussed selection criteria that can reduce the number of considered data while maintaining a satisfactory level of accuracy. The data acquisition was one of the hot areas of research as it needs the aid of diverse research fields. Pai et al. [84] and Lang et al. [98] described a system that includes visual aids to capture the haptic feedback and also
the texture properties of 3D objects. Fong [81] described an automated system that uses a range sensor with the ability to detect contact with the 3D object and slips on deformable surfaces. Hover et al. [83] designed a system that can capture the force field of viscous fluids and solids via a sensor equipped probe. An acquisition system can capture deformations globally or locally. Local deformation capturing such as [83] assumes flat infinite mesh topologies, which is an ideal case. Global deformations require a large amount of data and hence a robust learning algorithm.

The learning phase is an important part of the modeling process. The most used in data-driven haptics is artificial neural networks ANN as in [99][78][100]. Also other variations of ANN were used such as Gas-Elman network [86][101]. Other researchers used radial basis functions RBF [88][81]. To address the problem of large data volumes, Moorka et al. [78] suggested the use of principle component analysis (PCA) where important dimensions are only considered. No work, as far as the knowledge of the authors, has addressed the use of ensemble of learning methods in the data-driven haptic modeling domain.

The evaluation of the data-driven model is usually done through two methods: a comparison with parametric models such as FEM, mass spring systems MSS [78] or user acceptance studies [102][103]. The literature shows that data-driven models achieves high scores in this comparisons while minting the advantage of speed and applicability to real-time needs.

4.2.2 Supervised learning techniques

Supervised learning is the process of building a function from training data, which are collected using a human expert or automatic sensors [104]. Given a set of training examples \( \{(x_1, y_1), ..., (x_N, y_N)\} \), then a supervised learning algorithm produce the function \( g : X \rightarrow Y \), where \( X \) is the input vector and \( Y \) is the output vector. If the
output function is discrete, then it is a classification function, while if it is continuous, then it is a regression function, which is the case of haptic simulation. There are several algorithms in the literature such as support vector machines SVM, ANN, and RBF with diverse properties and application domains.

Several learning algorithms that handle regression have been used in the literature. They can be broadly classified into:

1. Single learning mechanism without data reduction [86]: this is the most popular approach in the literature. Some researchers also focused on the tuning of the learning method parameters [105].

2. Single learning mechanism with data reduction or feature selection [78]: this is suggested to reduce the number of features to reduce the learning algorithm variance.

3. Ensemble of multiple learning mechanisms and apply a selection criteria on the output such as averaging [106], bagging [107], or stacking [108].

4. A combination of the above.

It should be noted that not all supervised learning mechanisms can be used in the data-driven haptic simulation due to the presence of interactions and nonlinearities [83]. Methods such as artificial neural network (ANN) are favored over the ones that use linear functions such as linear SVM [109]. Besides, The speed of the learning process is irrelevant here as it is done offline.

The most important features of a learning algorithm in the domain of haptic simulation is the efficiency of the prediction and the ability to handle a high number of features and data volumes. Thus, a comparison is provided among the performance of the widely common approach of using a single learning algorithm against the use of feature reduction, ensemble of learning methods, and using both of them.
Feature reduction

This is the process of selecting a subset of the features or dimensions of the training data [110]. The objective is to reduce storage requirement, to focus on the unique important features, and to enhance the learning algorithm generalization. Common methods for selection are exhaustive search, stimulated annealing and greedy selections. Other methods also principal component analysis PCA and mass-PCA, which were used to increase the speed of data-driven [78] approaches and parametric methods such as FEM [77] as well.

Ensemble learning

Ensemble methods use multiple learning models to improve the prediction performance over singular models [111]. Ensembles require more computation time but enjoy more flexibility and more tuning parameters. This enable them, in theory, to over-fit the data more than single techniques. The famous types of ensembles are:

- Averaging: combine the multiple methods using Bayes’ law [106].
- Bagging or bootstrap aggregating: this method has two stages: bootstrapping and aggregating. The bootstrapping samples the data set randomly with replacement. Then, these samples are fed into a classifier. Each classifier has a vote with equal weights [107].
- Stacking or stacked generalization: partition the data into held-in and held-out sets. The training of the models is done on the held-in set and the decision is based on the held-out set in a similar manner as cross-validation method [108].
4.3 Comparative results

In this comparison, the standard implementation of the learning algorithms in Wakaito Environment for Knowledge Acquisition (WEKA) [112][113] is used. The tool has a wide variety of supervised learning and feature reduction algorithms implementations. In this work the comparison specifically is between the following algorithms:

1. ANN or multi layer perception

2. Attribute selection (AS) followed by ANN

3. Resampling-based ensemble using bootstrap aggregating (bagging) algorithm (Ens)

4. Attribute selection (AS) followed by the resampling-based ensemble (bagging) algorithm (Ens)

The comparison between the above algorithms will be on the coefficient of determination (the square of the correlation coefficient) ($R^2$), mean absolute error ($MAE$), root mean squared error ($\sqrt{MSE}$), relative absolute error ($RAE$), and root relative squared error ($\sqrt{RSE}$). These measures are calculated as follows:

\[
R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - f_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2} \tag{4.3.1}
\]

\[
MAE = \frac{1}{n} \sum_{i=1}^{n} |f_i - y_i| \tag{4.3.2}
\]

\[
\sqrt{MSE} = \frac{1}{n} \sqrt{\sum_{i=1}^{n} (f_i - y_i)^2} \tag{4.3.3}
\]

\[
RAE = \frac{\sum_{i=1}^{n} |f_i - y_i|}{\sum_{i=1}^{n} |y_i - \bar{y}|} \tag{4.3.4}
\]
And
\[ \sqrt{RSE} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - f_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}} \]  \hspace{1cm} (4.3.5)

Where, \( f_i \) is the prediction, \( y_i \) is the true value, and \( \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y_i \).

Tables 4.1 and 4.2 show the performance results of the mentioned algorithms and the figures 4.3, 4.4, 4.5, 4.6, and 4.7 show them visually.
Table 4.1: The performance data of the different supervised learning techniques for different training data percentages.

<table>
<thead>
<tr>
<th>Training %</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
<th>55</th>
<th>60</th>
<th>65</th>
<th>70</th>
<th>75</th>
<th>80</th>
<th>85</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training size</td>
<td>66</td>
<td>88</td>
<td>110</td>
<td>132</td>
<td>154</td>
<td>176</td>
<td>198</td>
<td>219</td>
<td>241</td>
<td>263</td>
<td>285</td>
<td>307</td>
<td>329</td>
<td>351</td>
<td>373</td>
</tr>
<tr>
<td>ANN R²</td>
<td>0.97838</td>
<td>0.9931</td>
<td>0.99514</td>
<td>0.9955</td>
<td>0.9949</td>
<td>0.99722</td>
<td>0.99688</td>
<td>0.99634</td>
<td>0.99652</td>
<td>0.99488</td>
<td>0.99412</td>
<td>0.9956</td>
<td>0.99646</td>
<td>0.9976</td>
<td></td>
</tr>
<tr>
<td>MAE</td>
<td>2.03202</td>
<td>1.3665</td>
<td>1.08086</td>
<td>1.06838</td>
<td>1.1066</td>
<td>1.11033</td>
<td>0.89036</td>
<td>0.81846</td>
<td>0.72984</td>
<td>0.59828</td>
<td>0.95778</td>
<td>0.88802</td>
<td>0.81428</td>
<td>1.05614</td>
<td></td>
</tr>
<tr>
<td>√MSE</td>
<td>3.25396</td>
<td>1.87472</td>
<td>1.6173</td>
<td>1.48216</td>
<td>1.51938</td>
<td>1.33868</td>
<td>1.46476</td>
<td>1.31388</td>
<td>1.26446</td>
<td>1.5026</td>
<td>1.54868</td>
<td>1.35013</td>
<td>1.31184</td>
<td>1.35136</td>
<td></td>
</tr>
<tr>
<td>Ens R²</td>
<td>0.87342</td>
<td>0.89788</td>
<td>0.92382</td>
<td>0.93062</td>
<td>0.96256</td>
<td>0.95066</td>
<td>0.96306</td>
<td>0.9754</td>
<td>0.9761</td>
<td>0.9762</td>
<td>0.97313</td>
<td>0.9667</td>
<td>0.97852</td>
<td>0.97568</td>
<td>0.98184</td>
</tr>
</tbody>
</table>
Table 4.2: The performance data of the different supervised learning techniques for different training data percentages (contd.).

<table>
<thead>
<tr>
<th>Training %</th>
<th>15</th>
<th>20</th>
<th>25</th>
<th>30</th>
<th>35</th>
<th>40</th>
<th>45</th>
<th>50</th>
<th>55</th>
<th>60</th>
<th>65</th>
<th>70</th>
<th>75</th>
<th>80</th>
<th>85</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training size</td>
<td>66</td>
<td>88</td>
<td>110</td>
<td>132</td>
<td>154</td>
<td>176</td>
<td>198</td>
<td>219</td>
<td>241</td>
<td>263</td>
<td>285</td>
<td>307</td>
<td>329</td>
<td>351</td>
<td>373</td>
</tr>
<tr>
<td>$R^2$</td>
<td>0.97092</td>
<td>0.9636</td>
<td>0.97066</td>
<td>0.97926</td>
<td>0.97962</td>
<td>0.97926</td>
<td>0.98144</td>
<td>0.97884</td>
<td>0.98138</td>
<td>0.97816</td>
<td>0.97954</td>
<td>0.97906</td>
<td>0.9753</td>
<td>0.96924</td>
<td></td>
</tr>
<tr>
<td>$R^2$</td>
<td>0.89282</td>
<td>0.90038</td>
<td>0.93822</td>
<td>0.92928</td>
<td>0.9547</td>
<td>0.94888</td>
<td>0.95432</td>
<td>0.96002</td>
<td>0.96866</td>
<td>0.96712</td>
<td>0.96214</td>
<td>0.95004</td>
<td>0.96032</td>
<td>0.96866</td>
<td>0.96866</td>
</tr>
<tr>
<td>$\sqrt{RSE}$</td>
<td>45.10204</td>
<td>44.13366</td>
<td>37.04316</td>
<td>38.4679</td>
<td>31.77922</td>
<td>34.41</td>
<td>30.94096</td>
<td>27.35584</td>
<td>26.01334</td>
<td>25.51482</td>
<td>27.0156</td>
<td>30.08886</td>
<td>25.04438</td>
<td>25.57992</td>
<td>25.26194</td>
</tr>
</tbody>
</table>
Figure 4.3: Coefficient of determination ($R^2$) comparison.

Figure 4.4: Mean absolute error (MAE) comparison.
Figure 4.5: Root mean squared error ($\sqrt{MSE}$) comparison.

Figure 4.6: Relative absolute error (RAE) comparison.
In the light of these results, it is clear that single supervised learning techniques such as ANN are sufficient and can outperform ensemble of multi techniques. This is due to the data volume as Ensemble learning can perform better with small data size and high dimensionality [114]. Thus, the contact points number can have a large effect on this result as the number of features to consider increases. Also, the number of object vertices affects the data size. However, material behavior or the object topology do not have a direct effect on dimensionality and hence cannot affect the result. The usage of a single learning technique also has the advantage of having less parameters to tune.

The feature reduction process is efficient as it saves storage and the results shows that the error margin is not high compared to learning without them. The amount of training data size required for efficient results cannot be estimated easily but the results show that over training is as bad as under training. Thus, the most important
characteristics of the sample is the coverage of the diverse possible scenarios [99].

4.4 Conclusion

In this work, a comparison between four techniques (ANN, attribute selection followed by ANN, an ensemble of learning methods, and attribute selection followed by an ensemble of learning methods) of learning a model for multiple points haptic interactions with 3D deformable models is presented. The problem is characterized by the high number of dimensions and data instances, which grow enormously as the number of contact points increases. The addition of an ensemble of learning techniques is not necessary to produce better results, especially in the case of two or less contact points, and the feature reduction preprocessing of the data is efficient and can reduce the storage requirements of the system. In future, an extension would be to test the performance for more contact points (e.g. 5 fingers) as this is a potential application for haptic technology.
Chapter 5

Generalization of data-driven methods

5.1 Overview

The real life human-object interactions involve multiple points of contact. Excluding poking, there are pulling, pushing, cutting, stitching, and other interactions that require at least two fingers to be accomplished. For example, in the medical field nearly all of the surgical tools, as shown in Figure 5.1 [115], require more than one finger to use. To haptically simulate surgical procedures involving such tools requires the data-driven modeling techniques to be scalable for more than one point interactions simulation.
Another generalization feature of the trained data model is the ability to be used for similar objects. By similar it is meant that they share a high percentage of their 1D curve-skeleton. Curve-skeletons are 1D structures that represent a simplified version of the geometry and topology of a 3D object [21]. Examples of this case may be found in human organs from different human bodies as our kidneys, livers and hearts are not identical but have nearly the same structure. Figure 5.2 [116] shows that even in the same person, left and right kidney can be a bit different.

Figure 5.1: Example of medical instruments that typically require multiple points of contact.

Figure 5.2: Right and left kidneys are not completely identical in size and geometry.
In the previous chapters, a new method was proposed to optimize the data acquisition and training phases of the data-driven modeling process. In the stage of data usage, the literature of the data-driven methods has been focusing so far on single point interactions and the usage of the trained model for one instance of deformable objects only. The possibilities of extending data-driven methodologies to multiple points and transfer deformations to similar instances of deformable models with minimal required extra training are investigated.

5.2 Multi-point interactions modeling

The objective is to find a suitable representation and computation model for handling multi-point interactions with deformable models. The presented model is for two points of contact but similar techniques can be applied for more points if they are considered pairwise. The more the number of contact points, the more parameters are required and hence the larger the problem dimensionality. Techniques such as principal component analysis PCA from the previous chapter come handy in these situations to sort the parameters by importance.

5.2.1 Related literature

Haptic feedback simulation for multiple points of contact has been addressed many times in the literature as in the work by [117] and Ferre et al. [118]. However, this research introduces the first time that it has been modeled using a data-driven technique. In this section, a summary of the literature of multiple points of contact simulation and the current limitations of existing methods is given. An overview of data-driven haptic simulation is then provided.
Multiple points of contact

Being able to compute reaction forces at multiple points of contact allows generating much richer haptic feedback than that obtained from commonly used single point approaches. This can be verified when the number of possible actions using multi-points is compared with their corresponding ones using only single point of contact. Integration of multi-point contact is especially useful for rendering multi-finger interaction with objects. In the following a non-exhaustive overview of some related display environments is given.

To date, most haptic rendering research has focused on single point haptics rendering, as identified in section 2.3.4. This is due to the availability of the hardware, the ease of implementation, and that it is a natural study point. Unfortunately, a single point of interaction cannot provide simulation of the human hand or many of the common tools that are used in the daily life. Thus, multi-point haptic research emerged to enable more realism and a more immersive haptic experience. The term *multi-point haptics* in the literature is not well documented or surveyed. From the hardware point of view, it can refer to one of the following:

- **Multi fingers** refer to grippers that enable users to grasp objects with two or more fingers [119, 118, 90, 92] or tactile multi finger display [120, 121, 122]. These tools also was extended by Endo et al. [123][124] to five fingers. Also researchers have used two haptic devices for multi fingers [125], however haptic forces are synthesized and rendered independently for each finger of the user. In their research deformation with the contact models was based on mass spring systems MSS models. Experiments measuring psychophysical thresholds were surveyed in [126], the ability of users to distinguish objects with different elasticity were demonstrated in [127], and appropriate analysis and design for enabling the multi fingers were discussed in [128]. Example
applications target visually impaired people services and medical tool simulations.

- **Multi hands** refer to using more than one haptic device by the same user. In [129] the author discussed the implementation and evaluation issues of dual haptic devices. A hybrid system merging multi fingers and multi hands (eight fingers), was introduced in [130] that had eight fingertip attachment devices with each fingertip attachment device connected by three strings.

- **Multi users** refer to distributed environments where the users interact with the same virtual objects and can feel the mutual effect on each other [131]. A multi rate approach has been suggested to enable multi users interaction with a single object [132][133]. This separates, the visual, haptic and collision detection loops and give each a different refresh rate.

### 5.2.2 Data-driven representation of multiple contacts

The data-driven simulation of a single point interaction has been discussed in the literature in [78, 81, 83, 134]. The most common approach is to have data samples composed of displacement values and the corresponding haptic feedback force values. This works well if the object geometry is not considered and the point of contact position remains fixed or does not make a difference as in the case of a circle or sphere boundaries. In [97] the authors discuss the general case where geometry is considered and how to reduce the number of required data samples.

In the case when more than one contact location is involved, additional information is required since there is a mutual relationship between the interaction points. Contacting a deformable object at two points simultaneously will usually (unless
the material behavior is linear) cause different force responses than exclusive single-point interaction. Figure 5.3 illustrates this effect. The plots display the X-, Y-, and Z-components of reaction forces resulting at the contact points. The first curve (simultaneous effect) represents the forces at a point resulting from two simultaneous external contacts during two-point interaction. The second curve (linear summation) is the summation of two separate force responses obtained when the contacts are computed individually. Note that the loads applied at the two locations differ in magnitude and orientation.
Figure 5.3: The effect of applying two simultaneous forces on deformable object is not the same as the linear summation of the effect of each individually.

Thus, the relation of the points to each other as well as to the object geometry has to be integrated into the data-driven model. Two different strategies were examined in this context. In the first, contact data are obtained assuming a global coordinate system and knowledge about the object geometry. In this case, contact positions $\mathbf{p}_{1,2}$, displacements $\mathbf{u}_{1,2}$, velocities $\mathbf{v}_{1,2}$, accelerations $\mathbf{a}_{1,2}$, and reaction forces $\mathbf{rf}_{1,2}$ are acquired for both interaction points, respectively. Unfortunately, this
straightforward technique requires acquisition of interaction data for the whole ob-
ject surface. In case of multiple interaction points, this quickly becomes infeasible.
However, segmentation techniques [97] might help in this case, e.g. by grouping
similar points and then selecting a sample from each group.

In the second method the local coordinate systems at the contact points is con-
sidered. This requires additional parameters to describe the mutual interaction ef-
fects. A straightforward extension is the inclusion of the distance between the con-
tact locations. However, this is still not sufficient to discriminate between different
cases of manipulation, as shown in Figure 5.4. In the first example, the contact pairs
(\(p_1, p_2\)) and (\(p_3, p_4\)) have the same distance, but are located at different points on
the surface. This case is addressed by including additional geometrical information
in the parameter set. In the second example, the contact pairs (\(p_1, p_2\)) and (\(p_3, p_4\))
differ with respect to the applied force orientation. This case will be addressed by
including angle information.
Figure 5.4: In both figures the distance between the contact points does not fully characterize the interaction. (a) To solve the ambiguity between the \((p_1, p_2)\) and \((p_3, p_4)\) pairs, an additional geometric information is added. (b) The angle \(\theta\) between the two displacement vectors is important to differentiate between varying orientations.

In order to encode the relation of the contact pair to the object, an ellipsoid is fitted to the local geometry. For the class of convex objects that are currently considered, this heuristic approach is a reasonable approximation of the interrelation between interaction points and geometry. The first ellipsoid axis is determined by the two contact points. The second axis is orthogonal to the first, and goes through the midpoint of the first, as well as through the closest surface point. The third one is orthogonal to, and thus directly defined by, the first two. A 3D (as well as a 2D) example of this is shown in Figure 5.5. The lengths of these axes are added as parameters to the data-driven model. Finally, the orientation of the contact displacements is encoded by the angle between the vectors. Thus, in the case of the local coordinate system the interaction parameters are the displacements \(u_{1,2}\), velocities \(v_{1,2}\), accelerations \(a_{1,2}\), and reaction forces \(rf_{1,2}\), as well as the ellipsoid axes lengths
δ_{1,2,3} and the orientation relation θ, which can be calculated as follows:

\[ \theta = \arccos \frac{\mathbf{v}_1 \cdot \mathbf{v}_2}{|\mathbf{v}_1||\mathbf{v}_2|} \]  \hspace{1cm} (5.2.1)

And

\[ \mathbf{v}_1 = U_{1x} \vec{X} + U_{1y} \vec{Y} + U_{1z} \vec{Z} \]  \hspace{1cm} (5.2.2)

\[ \mathbf{v}_2 = U_{2x} \vec{X} + U_{2y} \vec{Y} + U_{2z} \vec{Z} \]  \hspace{1cm} (5.2.3)

Where \( \mathbf{u}_1 \) and \( \mathbf{u}_2 \) are the displacements of the two points of contact. \( \mathbf{v}_1 \) and \( \mathbf{v}_2 \) are the displacement vectors.

Where \( \mathbf{u}_1 \) and \( \mathbf{u}_2 \) are the displacements of the two points of contact. \( \mathbf{v}_1 \) and \( \mathbf{v}_2 \) are the displacement vectors.

Figure 5.5: The usage of axis lengths of an ellipse or an ellipsoid for the local coordinate system (for the 2D and 3D case, respectively).

The mentioned parameters will be used as inputs to the interpolation system. Employing a local coordinate system can potentially reduce the amount of acquisitions needed for building a data-driven model. In the next section, a description on how this representation can be used for data-driven computation of contact dynamics is given.

### 5.2.3 Data-driven computation of contact dynamics

The complete multi point data-driven simulation is composed of three modules. The first two (data collection and interpolation) are utilized in an offline manner while
the third one (data replay) is used in runtime.

**Data collection module**

This module is responsible of the data gathering. The data represent the interactions with the deformable object. This can be done virtually in a simulated environment or physically with sensors. The data collection here is done using an impedance-based rendering. The opposite to this rendering type is the admittance-based rendering. In the impedance-based rendering, the movement of the tool serves as the input to the rendering algorithm, which calculates corresponding force feedback. In the admittance-based rendering, the force that the user applies to the interface is used as the input to the renderer [83]. The virtual data collection uses an accurate technique in a batch processing mode to collect samples of interactions with the virtual model. The most common technique is finite element methods (FEM) due to its accuracy and applicability to different scenarios. The simulation package used here is Abaqus. It allows different settings of material properties and behavior, to define object elasticity, viscosity and plasticity. The material behavior is described using parameters (e.g. Young’s modulus and Poisson ratio). The package also allows interfacing with Python to manipulate the results. The experiments are done with both 2D and 3D virtual objects, as illustrated in Figure 5.7a and Figure 5.7b respectively. The advantages of the virtual approach are the ability to generate the discretized vertices data (i.e. displacements and forces feedback) for all locations.

**Data interpolation module**

The collected data is input into this module to build the data-driven model. The data are interpolated using an appropriate model such as artificial neural networks,
ANN or radial basis functions RBF. The ANN Matlab toolbox is used in this implementation, which enables training and validation facilities. The structure of the ANN depends on the nature of the rendering (i.e. whether it is impedance-based or admittance-based) and the data-driven model (i.e. whether it is relative or absolute positioning). The ANN input layer structures are shown in Figure 5.6 for an admittance-based rendering of a 3D object. The hidden layers are set as the default suggested by the toolbox while the output layer has the displacements/force feeds. The training of the ANN can be done off line.

![Diagram of the employed ANNs](image)

Figure 5.6: Diagrams of the employed ANNs. The left one is for the local coordinate and the right one for the global coordinate system.

**Data replay module**

This is the module that is used in runtime. The user can use the haptic device to interact with virtual objects. The force feed will be calculated from the ANN or RBF systems. The advantages of this system are the speed as the calculations are linear and the ability to handle different scenarios such as nonlinearity, heterogeneity, or
viscosity. The main disadvantage is the requirement to collect prior data. The number of the collected data is related to the accuracy of the simulation. In [97] the authors show how data can be selected given the required level of accuracy or data size.

5.3 Simulation results of the multi-point model

Using the previously described approach for data-driven computation of contact dynamics an experiment has been carried out using 2D and 3D virtual object, discretized into triangular and tetrahedral elements, respectively (Figure 5.7a and Figure 5.7b, respectively). The 3D object is a half sphere constrained at the circular bottom (the 2D object is a cross-section of the half sphere). The FEM package was used to generate input for interactions at two pre-defined contact points.
Figure 5.7: Simulation of 2D and 3D virtual objects. Two loads were applied on each model.
For both test objects a homogeneous viscoelastic material was assumed, using a Young’s modulus of 10MPa, a Poisson ratio of 0.5, a material density of 5kg/m³, and viscoelastic parameters as $g_i = 0.2$, $k_i = 0.5$, and $\tau_i = 0.1$.

The manipulations of the object simulated in the FEM package were trajectories composed of random displacements of the contact positions (X-, Y-, and Z-components). These had a random length of up to 20% of the base diameter of the half sphere. The displacements were applied consecutively, after each previous one reached equilibrium. Therefore, the time of each displacement is variable and hence the number of resultant samples. Note that a random displacement can lead to pushing or pulling of the contact point. Figure 5.8 shows example trajectories for the two contact points. The input data obtained from these displacements were sampled at 0.1 second intervals. The parameters discussed in Section 3 were obtained for the global and the local coordinate system. In total 153 samples were collected for 22 trajectories. 70% of the collected data was used for training, leaving 15% of the data for validation and 15% for the testing. ANN were trained for both the local and the global case.
Figure 5.9: ANN performance using a global coordinate system on a 3D virtual object for 153 samples.
Figure 5.10: ANN performance using a local coordinate system on a 3D virtual object for 153 samples.

Figure 5.9 and Figure 5.10 show the performance of the data-driven force computation in 3D. The figures include six sub-figures representing the X-, Y-, and Z-components of the reaction forces at the two interaction points for a test trajectory. Each sub-figure has two curves showing the force computations and the predictions from the trained ANN. The average error between the actual and calculated is between 1% and 3% in both cases. Similar results were found for the 2D case.

Finally, the relation between the accuracy and the sample size is also examined. Figure 5.11 shows that the error percentage decreases with additional input data.
This behavior was observed both in 2D and 3D. For the special type of interaction examined in our first test using about 150 samples resulted in reasonable accuracy. Nevertheless, for smaller sample numbers the local approach lead to a larger error than when using the global one.
Figure 5.11: The relation between the sample size and the resulting error of the calculated results with respect to the actual recordings.
5.4 Deformation transfer to similar objects

The data-driven modeling of a deformable object behavior is quite a time consuming process. Fortunately, this is done offline and only the usage of the trained model is done in real-time. However, there is still a need to build a unique model for each individual deformable object. This condition does not necessarily be always there. If two objects share the same material and a sufficient percentage of their topology, deformation transfer ideas and mesh segmentation techniques can be used to build one data-driven model and then extend it to the other with minimal data modification. This approximation can be very handy in cases of the need to deal with different objects, which belong to one class such as a set of livers for example. Also this can be fully utilized if different sizes of the same object is required.

5.4.1 Shape correspondence

The search for a correspondent relationship between objects is a common problem in many disciplines. It even has different names in the literature such as registration, alignment and matching. A recent survey about the topic is done by Van Kaick et al. [135]. The problem can be stated as: given two objects $O_1$ and $O_2$, a relation $R$ is defined between their vertices that lead to them behaving as similar as possible. An example on two livers is shown in Figure 5.12. The two livers share most of their topology but differ in some areas. Once these areas are defined, additional data can be collected to cover them. This increases the process of building data-driven models.
In [136] the authors show how to calculate correspondence between two objects using user guided points selection of the source and destination. Their algorithm followed the work of [137], which is an iterative closest point algorithm that search for the matching pairs between the source 3D mesh and the destination one in a close proximity. The problem with that approach is that it needs a lot of guidance from the user. For instance, the user is required to mark an average of 64 vertices to match two meshes in [136]. The problem with their approach is that it used a dense correspondence without getting more knowledge about the underlying 3D meshes to be matched.

An efficient approach for data-driven haptic rendering is to match the two objects using sparse correspondence. Thus, rather than matching a set of vertices, the match is done between planes segmented using techniques mentioned in previous sections. The two objects still have to follow the assumption that a high similarity exist between their shapes. This method improves the correspondence process by limiting the input needed from the user and directly defines what planes needs to be

Figure 5.12: Data can be collected for only the marked areas and not the whole object.
reconsidered for data collection.

5.5 Conclusion

The ability of data-driven model to be generalized for more scenarios other than single point of interaction is important. Most of real-life interactions are not done through single points. Also, the multi-point interaction scenarios are complicated and computational heavy for rival parametric methods. Thus, it is a promising area for data-driven methods to compete in. In this chapter, an illustration has been shown on how to model the behavior of deformable objects with two points of loads interaction. Linear and nonlinear response could be modeled using extra inputs and parameters. Arbitrary force directions could be also modeled through the transformation into local coordinates. Local coordinates method allows one scenario to be generalized for many. The models can be generalized for more than two points. Also, it has been shown how to re-use a collected model for similar objects, which share similar topology. This is an important feature when multiple instances of one class of objects need to be trained with minimal initialization time.
Chapter 6

Virtual grasp - case study

6.1 Overview

In this chapter, a novel haptic case study is developed and presented that demonstrates the performance and efficiency of the data-driven techniques that have been discussed in the previous chapters. So far, the implementation and usage of haptic feedback has been delayed and even avoided if possible in commercial domains such as medical training [138]. The main reasons for this could be stated, considering a medical training domain example, in the following points:

Modeling nonlinear heterogeneous material complex behavior The soft tissues of human organs are complex objects that have layered heterogeneous materials and sophisticated geometries. Oversimplifying the organs, using a non physically accurate approach, prevent realistic feeling and hence inaccurate training. The ability of the system framework to handle high resolutions and fine details is considered as a main design goal of any virtual reality system to be used and trusted as an alternative to real life experience.
Modeling multiple points of contact to enable real life tools The other component of the system is the tools that the user manipulate the objects through. The more points of contact enabled the more actions and moves that can be performed. One point of contact touching scenario is not common in real life as most of the tools need more than one finger as seen in medical field for instance.

Real time speed The above two items are important design goals for a system to have. However, they come with the price of heavy calculations and extensive analysis, which reduce the speed and stability of the system. Therefore, the system needs to achieve them and show real-time performance in the same time. The high requirement of haptic loop refresh rate (1000 Hz) needs to be met to guarantee a stable user experience.

The developed system design goals are to handle all these three requirements. The system uses a novel haptic gripper beside the proposed haptic rendering algorithms. In the next sections we will describe the system hardware, software and data structures. The operational phases and required configuration experiments are then discussed. This is followed by results section and conclusions.

6.2 Related works

In recent years, there is a trend to increase the number of interaction points and degrees of freedom (DoFs) of the haptic devices. Kawasaki [123] introduced a multi-point system that can be used in future science encyclopedia and breast palpation based on HIRO [91] system, as shown in Figure 6.1a. Handa et al. [117] evaluated a system that uses three Premium PHANToMs 1.5A, with 3 degrees of freedom DoF, (by SensAble Technologies, Inc.), as shown in Figure 6.1b. Also,
Ferre et al. [118] used two MasterFinger-2 (MF-2) [139] for interacting with virtual objects, as shown in Figure 6.1c.

![Multi-finger HIRO and Three PHANToMs](image)

(c) Two MasterFinger-2

Figure 6.1: Literature related multi-point haptic systems.

Of these three multi-point systems, only the one developed by Kawasaki implemented interactions with deformable models. That was done using an "elementary displacement" approach from [140] coupled with finite element methods FEM. The developed system in this chapter provides an accurate haptic rendering using the proposed data-driven methods. This is coupled with the multi-point haptic gripper to enable a realistic interaction with the virtual deformable models regardless of how sophisticated are their composing materials.
6.3 System description

The developed system can be divided broadly into two parts: software interface and a hardware one. The software interface includes the algorithms and methods that collect the data, build learning models, and replay scenarios in real time. The hardware component is represented in the haptic gripper, the force sensors, and the accelerometers. The overall structure of the system make use of the proposed methods and algorithms from the previous chapters but instead of working on a simulation environment such as an FEM package, a physical sensors and real physical materials are used. Figure 6.2 shows the system architecture.

![System Architecture Diagram]

Figure 6.2: Flow diagram of a the VirtualGrasp application.

6.3.1 Multi-point haptic gripper - Hardware

Several approaches have been proposed to increase the number of contact points in haptic interfaces. A more logical method involves the use of multiple single-point of contact devices. Two or more of these devices are used to produce force
feedback to an operator’s hand, as shown in Figure 6.1, ultimately allowing them to grasp and manipulate virtual objects on a multi-point level. However, the technique of interfacing with these systems is not intuitive, and does not allow for a natural interaction method due to constrained hardware motions, and potential mechanical collisions [141].

6.3.2 Design objectives

The haptic grasping system is designed to fit any type of single-point-of-contact commercial or custom haptic interface. The hand interface should be created as a light-weight modular system that can be integrated onto a haptic device to allow for bimanual operation. It should also allow a user to interact with virtual or teleoperated objects by way of a pinch grasp. Easy assembly and disassembly is a requirement for this device. As a result, it must be scalable to suit various sized hands and finger widths. The design objectives for this device were centered on the application of dexterous virtual assembly, which require the device to accommodate various virtual end-effectors such as a human hand, screwdriver, pliers and drill. In bimanual operation, this approach should allow two of these systems to work concurrently without interference. The commercial or custom haptic interface to which the grasping device is attached would benefit the application by having greater than three degrees of freedom capability.

In order to extend the grasping capabilities of the hand interface to accommodate different hand sizes, it must also allow for natural pinch grasp motions. This requires the structure to mechanically encompass this functionality while also maintaining device stiffness, durability and low mass. As shown in Figure 6.3, the task of assembling a mechanical system requires the user to use their thumb and index finger to rotate a bolt, consequently requiring a combination of open/close and vertical
finger motions to complete the task.

Figure 6.3: Dexterous grasping scenario to rotate a bolt.

6.3.3 Design descriptions

The dexterity of a user’s hand requires a hand interface that will not restrict the natural motion of the fingers throughout any grasping procedure. The haptic grasping interface is designed to follow the natural grasp closure of the operator’s hand. This is achieved by use of a serial link structure that allows each finger interface point to move around a central axis and also within a vertical motion as shown in Figure 6.4. To aid in the design and analysis of the hand interface, the kinematic relationship between the joint angles and the position of the finger interaction points is determined. This allows for tracking the position of the operator’s fingertips in any possible pinch grasp configuration or finger motion. Figure 6.5 shows the attachment of the gripper with the commercial haptic Force Dimension Omega™ - 3 DoF [9] device.
Figure 6.4: The haptic gripper handle.

Figure 6.5: The gripper handle attached to the Force Dimension Omega device.
6.3.4 Data collection and replay frameworks - Software

The software part of the system is responsible for the haptic and visual rendering. The interface is developed using an open source haptic framework called CHAI 3D [142], which is developed in C++ programming language. The CHAI 3D framework enable haptic and visual rendering of interactions with deformable models. The package defaults to have a single point interaction and uses mass spring systems MSS for haptic rendering.

To incorporate the proposed system, a custom definition of the connected haptic devices was introduced as in code snippet 3. Also, the force calculation method had to be changed from algorithm 4 to algorithm 5. In both algorithm 4 and algorithm 5 the deformable object is discretized into a set of spheres that serve the purpose of collision detection [143]. It is to be noted that algorithm 4 runs for every sphere that represent the deformable object while algorithm 5 runs only once and produce an augmented force. The assumption for algorithm 5 is that only one sphere intersects with a finger at time and others are neglected.

**Algorithm 3 Custom Device class**

```plaintext
1: Struct cMyCustomDevice Inherits CDeltaDevices
2: {
3:     // Override methods for two points
4:     setForce(cVector3d& a_force_1, cVector3d& a_force_2);
5:     getPosition(cVector3d& a_position_1, cVector3d& a_position_2);
6:     ....
7: }
```
Algorithm 4 MSS Force Computation

Input: finger_Position, sphere_Position, finger_Radius, sphere_Radius, MSS_Stiffness

Output: force

1: force = 0

2: penetration = finger_Position − sphere_Position

3:

4: if penetration.length() < 0.00001 then

5: return force

6:

7: else if (thenpenetration.length() > (finger_Position + sphere_Position))

8: return force

9:

10: else

11: penetration_Distance = finger_Radius + sphere_Radius − penetration.length()

12: force_Direction = penetration.normalize()

13: return (penetration_Distance * MSS_Stiffness) × force_Direction

14: end if
**Algorithm 5** ANN Force Computation

**Input:** finger.Pos_arr, sphere_arr, finger.Radius

**Output:** aug.force_arr

1: for $i = 1$ to $2$ do
2:    [coll.sphere.pos_arr, penet.dist_arr, force.dir_arr] = checkCollision(sphere_arr, finger.Pos_arr[i], finger.Radius)
3:    ▷ Similar to the procedure in the MSS algorithm
4: end for
5:
6: aug.force_arr = simANN(penet.dist_arr, coll.sphere.pos_arr)
7: for $i = 1$ to $2$ do
8:    aug.force_arr[i] = aug.force_arr[i] $\times$ force.dir_arr[i]
9: end for
10: return aug.force_arr[i]

Figure 6.6 shows a rendered box using the CHAI 3D package. The arrows represent the normals on the discretized elements.

![Real box](image1.png) ![Virtual Box](image2.png)

(a) Real box  (b) Virtual Box

Figure 6.6: Representation of deformable box using CHAI 3D framework.
6.4 Model building

The proposed system depends on a pre-computed model to be able to perform accurately. The process of building this offline model is done through the following steps: data collection from a real model, motors calibration, and artificial neural networks ANN training. Each of these steps require a set of hardware and/or software. The input to the whole process is a deformable real model. In this work, a silicone rubber cube of a length of a 5 cm was used. The real model has been discretized with marks to facilitate the data collection phase.

6.4.1 Data collection and system calibration

The data collection phase is done in an impedance-based form where the displacements are the inputs and the feed forces are the output. As shown in the system description section, the haptic gripper has two motors fixed on the haptic device handle. A basic trigonometry is done to calculate the absolute positions of each point in order to convert the circular movement to a linear one utilizing $\theta$, as shown in Figure 6.4. To capture the feed forces, the haptic gripper, the two force sensors and the acceleration sensors were connected to the developed real-time data capture interface to compute the data in the desired format.

Figure 6.7 shows the data collection system. The data is fed to a low pass filter to remove the noise. Figures 6.8 and 6.9 show plot of collected forces and displacements. To facilitate the selection procedure of the data a similar strategy to [97] were used. Each vertex has an index that is composed of (face, row, column) combination. The data structure used to hold the data is shown in table 6.1. The relative displacements are acquired by scaling the recorded displacements from $[\text{disp}_{\text{min}}, \text{disp}_{\text{max}}]$ to $[0, \text{disp}_{\text{max}} - \text{disp}_{\text{min}}]$. This is required for accurate training and
ANN performance.

Figure 6.7: The data collection system.

Figure 6.8: Collected forces sample.
Table 6.1: The data structure employed.

<table>
<thead>
<tr>
<th>Class</th>
<th>Finger_left</th>
<th>Finger_right</th>
</tr>
</thead>
<tbody>
<tr>
<td>Forces</td>
<td>$f_{\text{left}}$</td>
<td>$f_{\text{right}}$</td>
</tr>
<tr>
<td>Accelerations</td>
<td>$\text{Acc}_{\text{left}}$</td>
<td>$\text{Acc}_{\text{right}}$</td>
</tr>
<tr>
<td>Displacements</td>
<td>$\Delta \text{Disp}_{x,\text{left}}$</td>
<td>$\Delta \text{Disp}_{x,\text{left}}$</td>
</tr>
<tr>
<td></td>
<td>$\Delta \text{Disp}_{y,\text{left}}$</td>
<td>$\Delta \text{Disp}_{y,\text{left}}$</td>
</tr>
<tr>
<td></td>
<td>$\Delta \text{Disp}_{z,\text{left}}$</td>
<td>$\Delta \text{Disp}_{z,\text{left}}$</td>
</tr>
<tr>
<td>Indices</td>
<td>face_left</td>
<td>face_right</td>
</tr>
<tr>
<td></td>
<td>row_left</td>
<td>row_right</td>
</tr>
<tr>
<td></td>
<td>column_left</td>
<td>column_right</td>
</tr>
</tbody>
</table>

Figure 6.9: Collected displacements sample (right and left).

The employed motors in the gripper are controlled through varying the amount of supplied current. The collected data is in the form of forces in Newton. Thus an experiment was required to relate the amount of current and a 1 N force. A software interface, shown in Figure 6.10, was developed that controls the amount of current to generate a plot of the current-force relationship.
6.4.2 ANN training

The data is fed into a supervised learning technique, which is ANN in this case. The output of this process is a trained network that can be simulated in run-time to acquire the calculated forces. To ensure an improved coverage of the ANN [144], a feature selection process is done before hand on the data. The feature selection is done using Wakaito Environment for Knowledge Acquisition (WEKA) [112][113] software, as shown in Figure 6.11. This ensures that low impact features are neglected.
The ANN is of a feed-forward back-propagation type. The performance function is the mean squared error (MSE). There are three layers: input, hidden and output. The input layer includes the displacements, indices and accelerations unless some of these features are removed in the feature selection process. The hidden layer is set to contain 40 neurons while the output layer is reserved for the feed forces. The data set is divided into three set: one for training, one for validation while training, and the rest is left for testing after the training is over.

The trained ANN weights are then saved to a file accessible to the system software interface. Through an access layer the ANN is simulated whenever an interaction happens with the deformable object to calculate the desired haptic forces.
6.5 Results

In this section we measure how successful is the system from two points of view. The first is the system accuracy in predicting the haptic forces compared to the actual forces using the MSE measure. The other is the system ability to meet the graphics and haptic loop requirement and run smoothly.

To check the first point, the system calculated feed forces are plotted against the actual forces, as shown in Figure 6.12. This was done for 20 random pair of points and the average MSE was not more than 5% of the force range.

![Figure 6.12: The ANN performance - Calculated force vs. Actual.](image)

The other point is checked by generating the frames rate per second (FPS), as shown in Figure 6.13. The average haptic loop is approximately 750 frames and the average for the graphics loop is approximately 25 frames. Although these rates are less than the recommended rates, different force magnitudes and interaction points...
were tested and the system was stable. The main reason for this is the dependence on the CHAI3D framework visual rendering. A direct improvement is to decouple the haptic loop from the graphics one. Another solution is to implement a better global visual rendering other than the implemented MSS one.

Figure 6.13: virtualGrasp interface with graphics and haptics frames per second (FPS).

6.6 Conclusion

This work provides an accurate multi-point haptic system that enable the users to grasp deformable models. This is done through a novel haptic gripper device and a data-driven haptic rendering algorithm. The system is efficient and stable compared to the collected data. One of the limitations of the system though is that it models only local deformations visually and rely on MSS for global deformations. In the future, it is planned to employ a visual system to capture the global visual deformations. Also, it is useful to perform a user study on the system on simple tasks
like recognition and higher tasks like breast palpation test and quality assurance in bread and cheese factories.
Chapter 7

Conclusion

7.1 Research outcomes

This thesis presents optimization techniques for overcoming the challenges in the

data-driven modeling of haptic interactions with deformable objects. These chal-

lenges can be summarized as follows:

Data collection time: The required volume of data is large in order to cover the

large number of interaction scenarios with variable force vectors and reduce

the amount of error in the learning process. The process takes time that could

be up to days [5] whether the data to be collected physically with sensors or

in a simulation environment. The longer the process, the higher the possi-

bility of having errors and inconsistencies. Planning algorithms that utilize

the geometry of the underlying object and change the complexity from $O(n)$

to approximately $O(\log n)$ where $n$ is the number of object vertices was in-

troduced in chapter 3. This reduced the time required tremendously while

maintained the level of accuracy.

Data storage: This is again related to the problem of the large volume of data
especially when considering the visual simulation of the object. In this case, the positions of all vertices of the object have to be stored for each time step in the simulation. A parameterized representation of the object in the form of a set of planes is presented. For every plane there are an equation, a dimension of a rectangle, and positions of missing vertices. The amount of the storage required in this case is minimal, which allows the recording of more scenarios and better user experience.

**Data Pre-processing:** The data collected do not all have the same contribution in the learning process. A major problem in supervised learning is the over-fitting of data. This can reduce the coverage of the model. The identification of the priority of the features of the data set and removing the less important is vital for the learning process.

**Learning technique choice:** There are several supervised learning methods available such as artificial neural networks ANN and radial basis functions RBF. In this thesis, an examination of the choice of one of them versus using an ensemble of multiple techniques is provided. Using a single learning method produces better results than an ensemble for this type of data and can be trained faster.

**Multi-point interactions modeling:** Most of the real-life interactions need multiple point interaction or require tools that need more than one finger. The parametric methods such as finite element methods FEM need in most cases a very high processing power to handle multi point interaction scenarios. This is a potential area for data-driven methods as the complexity of the model increases in the offline part with the increased number of interaction points but the online speed is not affected in the same way. A model that can capture
two points of contact is introduced. It can capture the two applied loads effect and their mutual effect as well.

**Generalization to similar objects:** This is a great feature for data-driven modeling to have. The ability to reduce the model creation for similar objects can improve the reliability on the data-driven modeling and make it time efficient.

The standard data-driven modeling of the haptic rendering interaction has many advantages over the rival parametric methods. With the addition of the introduced optimization methods in this thesis the data-driven modeling can be done for any material whether it is homogeneous or heterogeneous or any combination of other properties and there no need to mechanically identify the physical properties (e.g. Poisson ration) as there is no explicit model. The modeling can be done for any object topology on a global level deformation. Also, The number of interaction points can be more than one. All of these features and still a real time speed that meets the haptic loop requirement can be met thanks to the offline stage with the heavy calculations.

The limitations of the data-driven methods are primarily in the data collection phase where noise can affect the quality of the model. Another limitation also exist in the modeling of material with hysteresis effect. The modes of interaction need to be carefully planned to account for the memory feature.

### 7.2 Potential applications

There are several applications that can use the data-driven modeling of the haptic interactions. Here are some examples:
7.2.1 **Haptically enabled medical atlas**

Understanding the human body is one of the important fields of health informatics. Both students and professionals need to have access to a detailed view of different organs to observe how things work. In recent years, a lot of advances have been made to create 3D searchable objects of the human body [145][146], as shown in Figure 7.1 [145]. These models enabled the study of the components of the human body. However, the behavior and material properties are still not associated with these components. Attaching the material properties with each object and simulating its haptic feedback with different interactions can open new areas of research and enhance further the understanding of the human body anatomy.

![Figure 7.1: A cross section 3D view of a human brain details in Google body.](image)

To accomplish the build of such a haptically enabled model, the computational
tomography (CT) scans and conventional 3D reconstruction methods are not sufficient. Empirical data need to be collected for interactions with the objects and a data-driven model can then be trained to simulate the haptic and visual feeds. This application can be enhanced further with shape correspondence techniques to enable the customization of the models with different human bodies to best match certain cases.

Figure 7.2: Haptic simulation of a human liver in a typical medical application.

7.2.2 Low-cost low-processing power applications

The data-driven model is built offline and can be associated with deformable objects. With parameterized storage of the data and feature reduction algorithms the data size can be small enough to be sent over network and be used in mobile devices. This allows a whole new dimension of haptic applications. The applications can be run with very limited resources and support personal uses. An immersive
environment can be easily created with low budget equipment for home use. Example applications can be in the shopping domain where a customer can touch a remote product and feel its material. Other examples can be in entertainment where users can play games for example with all fingers and have the same feeling as the correspondent physical action.

### 7.2.3 Hybrid systems

The data-driven modeling technique also can be integrated with rival parametric methods. Certain phases in FEM for instance can still use empirical data and the rest can continue perimetrically. An example is the time integration phase where velocity \( v \) and acceleration \( v' \) needs to be calculated:

\[
x' = v \quad (7.2.1)
\]

\[
v = F(v, x, t) \quad (7.2.2)
\]

Where \( x \) is the displacement and \( t \) is the time step. There are two different techniques: explicit and implicit [147]. The explicit method calculates these variables as follow:

\[
x(t + \Delta t) = x(t) + \Delta tv(t) \quad (7.2.3)
\]

\[
v(t + \Delta t) = v(t) + \Delta F(v(t), x(t), t) \quad (7.2.4)
\]

While the implicit method uses quantities at the next time step \( t + \Delta t \) on both sides:

\[
x(t + \Delta t) = x(t) + \Delta tv(t + \Delta t) \quad (7.2.5)
\]

\[
v(t + \Delta t) = v(t) + \Delta F(v(t + \Delta t), x(t + \Delta t), t) \quad (7.2.6)
\]

The explicit method is easy to implement but unstable for large values of \( \Delta t \) [13] while the implicit is stable but requires the solution of a system of equations. This
tradeoff can be relaxed with the usage of data-driven methods in this step to calculate velocity and acceleration values.

7.3 Future research directions

There are still some points for future research directions to investigate. The model performance can be improved or altered to serve other application demands. Here is a set of potential areas:

The stability of the data-driven model: This area has not been studied in full coverage. The continuity of the generated values may need a post processing of the model.

Model extension to higher interaction level: The generalization of the model to more than two points of contact and the handling of concave objects. The friction of the material can also be modeled empirically to support real time tactile feed. This can benefit from the advances made in the haptic hardware.

Modeling materials with hysteresis effects: This can be done through the usage of time series forecasting methods. The data collection needs to be modified as well to account for the memory feature.

Applying different classes of learning models: The introduction of stochastic models can improve the adaptiveness of the model and open opportunities for fluid simulation with variable topology and hybrid materials where plastic and elastic deformations coexist.

Hybrid data-driven and parametric model: Another room of improvement is combining parametric methods with data driven ones. A multi resolution system
can be built where different parts of the modeled objects use different rendering techniques according to priorities and user requirements.

**Improve object description and representation:** This includes the process of identifying other parameters to describe the material directly or indirectly to avoid ambiguity and improve learning variance. Extra parameters such as torques could be used for certain interaction scenarios.

**The sole use of visual aids to collect data:** With the recent advances in computer vision and availability of devices at homes, visual parameter estimation method using a camera or more can be used. This will result in a magnificent cost reduction as no expensive sensors are involved. This can serve the needs of end users who want to be able to differentiate between different materials in a virtual environment but with tolerable level of accuracy.
References


[105] R. Hover, M. Di Luca, G. Szekely, and M. Harders, “Computationally ef-
ficient techniques for data-driven haptic rendering,” in Proceedings of the
World Haptics 2009 - Third Joint EuroHaptics conference and Symposium on
Haptic Interfaces for Virtual Environment and Teleoperator Systems, (Wash-


140, 1996.


[109] V. Vapnik, “Pattern recognition using generalized portrait method,” Automa-

[110] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual infor-
mation: criteria of max-dependency, max-relevance, and min-redundancy,” 
IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, 
no. 8, pp. 1226 – 1238, 2005.


[112] I. Witten, E. Frank, L. Trigg, M. Hall, G. Holmes, and S. Cunningham, 
“Weka: Practical machine learning tools and techniques with java imple-


