UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS

A learning-based method for 3D object modeling and deformation prediction using neural networks

Thèse présentée au Département d'informatique et d'ingénierie

Pour l'obtention du grade de

PHILOSOPHIAE DOCTOR (Ph.D)

En sciences et technologies de l'information

PAR

BILAL TAWBE

Gatineau, Québec, Canada, Avril 2021

Jury d’évaluation

Président du Jury :	Dr. Omar Abdul Wahab, Ph.D.
Directeur de recherche :	Dr. Mohand Said Allili, Ph.D.
Codirecteur de recherche :	Dr. Ana-Maria Cretu, Ph.D.
Examinateur interne :	Dr. Omar Abdul Wahab, Ph.D.
Examinateur interne :	Dr. Nadia Baaziz, Ph.D.
Examinateur externe :	Dr. Bob-Antoine Jerry Ménélas, Ph.D.

REMERCIEMENTS

Je tiens à remercier profondément mon directeur de thèse, Professeur Mohand Said Allili et mon co-directeur de thèse, Professeur Ana-Maria Cretu de m'avoir accueilli dans leur équipe et de m'avoir encadré durant toutes les étapes de la recherche. Leur explications judicieuses el leur patience ont été un atout de taille pour compléter ce travail. Je vous remercie de tout mon cœur pour votre aide et soutien tout au long de ces années.

Je tiens également à remercier les professeurs Omar Abdul Wahab, Nadia Baaziz et Bob-Antoine Jerry Ménélas, d'avoir accepté d'assurer la tâche d'examinateurs et d'avoir consacré une partie de leur temps à l'examen de cette thèse.

Je tiens à remercier mes collègues à l'Université et au travail surtout mon ami le professeur Jamal Abd-Ali.

Mes remerciements les plus sincères vont à mon père et ma mère qui m'ont encouragé à faire ce travail.

À ma femme Hanan, à mes enfants et à ma petite Zainab. Je vous aime tous.

SOMMAIRE

La représentation réaliste des déformations d’objets 3D reste un domaine de recherche actif, notamment pour les déformations dont le comportement ne peut pas être décrit simplement en termes de paramètres d'élasticité. Plusieurs méthodes ont été proposées dans la littérature pour modéliser des objets déformables. La plupart de ces méthodes supposent que les paramètres décrivant le comportement de l'objet sont connus à l'avance ou que les valeurs de ces derniers sont choisies manuellement en les ajustant jusqu'à ce que la forme et le comportement de l'objet paraissent plausibles. Ce n'est pas un processus facile et ne peut pas être adopté lorsqu'une certaine précision du modèle est souhaitée. Ces problématiques justifient l’intérêt porté au développement des nouvelles méthodes qui ne tiennent pas compte des matériaux de l’objet, comme notre méthode proposée.

Dans cette thèse nous proposons une nouvelle méthodologie pour la modélisation et à la prédiction de la déformation d’objets souples qui ne font pas d’hypothèses sur les matériaux de l’objet. Nous capitalisons sur les solutions d’intelligence informatique, notamment sur la combinaison de la technique de neural-gas avec les réseaux de neurones à propagation avant (feedforward neural network).

Une nouvelle approche basée sur l'ajustement du neural-gas est proposée pour décrire les particularités d'une déformation sur la surface 3D sélectivement simplifiée de l'objet, sans avoir de connaissance des matériaux de l'objet. Une procédure d'alignement, un regroupement basé sur la distance et l'inspiration de l'échantillonnage stratifié sont des blocs de soutien supplémentaires à cette fin.

Une série de réseaux de neurones à propagation avant (feedforward neural network) est ensuite entraînée pour prévoir la correspondance entre les paramètres de force

caractérisant l'interaction avec l'objet et la modification de la forme de l'objet, tels que capturés par la méthode de gaz neuronal ajusté. Cette méthode permet de prédire la position des nœuds gazeux neuronaux pour des amplitudes de force et des angles non mesurés. Des tests sur des ensembles de données ont permis de valider l'approche et évaluer sa performance par rapport aux travaux de la littérature.

Cette méthode est améliorée et adaptée pour pouvoir prédire la déformation d'un objet en utilisant des forces importantes pouvant atteindre 100 N. Alors que, dans la première version de notre méthode, nous avons prédit les objets en augmentant la force utilisée dans l'entraînement des réseaux de neurones jusqu'à un maximum de 5 N. Dans la première version de notre méthode, nous avons utilisé 3 réseaux par cluster (5 clusters pour représenter chaque objet déformé) et dans la version améliorée, nous avons utilisé un seul réseau pour prédire l'objet entier. De cette façon, nous avons réduit le nombre d'opérations et le temps de calcul est devenu beaucoup plus court.

Nous avons adapté notre méthode de prédiction pour pouvoir transférer l'apprentissage entre plusieurs réseaux de neurones. Par cette méthode, nous pouvons prédire la déformation d'un petit objet en utilisant de grands objets déformés et prédire la déformation sur un grand objet en utilisant de petits objets déformés. Cette méthode fonctionne en utilisant des objets qui ont les mêmes formes et matériaux. Enfin, la méthode proposée est comparée à une méthode de modélisation classique pour la modélisation d'objets déformables, la méthode des éléments finis (FEM).

ABSTRACT

The realistic representation of deformations is still an active area of research, especially for soft, deformable objects whose behavior cannot be simply described in terms of elasticity parameters. Several methods have been proposed in the literature for modeling deformable objects. Most of these methods assume that the parameters describing the object behavior are known in advance or values for these parameters are chosen by manually tuning them until the object shape and behavior seems plausible. This is not an easy process and cannot be adopted when a certain accuracy of the model is desired. All the issues in these methods justify the interest in the development of new methods that do not make assumptions on the material of the object, such as our proposed method.

In this thesis, we propose original solutions to the modeling and prediction of the deformation of soft objects that do not make assumptions about the material of the object and that capture its deformation behavior implicitly. For this purpose, we capitalize on computational intelligence solutions, namely on a combination of neural gas fitting with feedforward neural network prediction.

A novel approach based on neural gas fitting is proposed to describe the particularities of a deformation over the selectively simplified 3D surface of the object, without requiring knowledge of the object material, and obtain a compact representation of the object shape. An alignment procedure, distance-based clustering, and inspiration from stratified sampling are additional supporting blocks for this purpose.

Feedforward neural networks are then trained to predict the mapping between the force parameters characterizing the interaction with the object that creates the deformation and the change in the object shape, as captured by the fitted neural gas network. As such, the proposed method can predict the position of neural gas nodes, and thus the shape of an object for force magnitudes and angles that were not measured. Tests on datasets made it possible to validate the approach and assess its performance against the work of the literature.

This method is improved and adapted to be able to predict the deformation of an object using large forces up to 100 N. While in the first version of our method we predicted the objects by increasing the force used in training neural networks up to a maximum of 5 N. In the first version of our method, we used 3 networks per cluster (5 clusters to represent each deformed object) and in the improved version, we used a single network for predict the whole object. In this way, we have reduced the number of operations and the computation time has become much shorter.

We adapted our prediction method to be able to transfer learning between several neural networks. By this method, we can predict the deformation of a small object using large deformed objects and predict the deformation on a large object using small deformed objects. This method works by using objects that have the same shapes and materials. Finally, the proposed method is compared with a classical modeling method for deformable object modeling, the finite-element method (FEM).

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS, ACRONYMS AND SYMBOLS

RGB-D Red Green Blue – Depth

NURBS Non-Uniform Rational Basis Splines

RANSAC Random Sample Consensus

ICP Iterative Closest Point

FEM Finite Element Method

SOM Self-Organizing Map

ANN Artificial Neural Network

SSIM Structural Similarity Metric

$\boldsymbol{m_i}$, $\boldsymbol{m_j}$ mass of vertices

$\mathbf{K}_{\boldsymbol{ij}}$ stiffness of the spring

$\mathbf{F}_{\boldsymbol{ij}}$ internal force of a spring between two vertices

$\mathbf{F}_{\boldsymbol{external}}$ external force over each vertex

$\mathbf{f}_{\boldsymbol{result}}$ sum of elasticity force, curvature force and constraint force

$\mathbf{x}_{\boldsymbol{i}}$ positions

$\dot{\boldsymbol{x}}_{\boldsymbol{i}}$ velocities

$\ddot{\boldsymbol{x}}_{\boldsymbol{i}}$ accelerations

$\boldsymbol{\gamma_i}$ are the damping factor

$\boldsymbol{F}$ total of external forces

$\boldsymbol{M}$ mass matrix

$\boldsymbol{D}$ matrix of damping

$\boldsymbol{K}$ stiffness matrix

$\boldsymbol{u}$ displacement vector

$\dot{\boldsymbol{u}}$ velocity vector

$\ddot{\boldsymbol{u}}$ acceleration vector

$\boldsymbol{\Gamma_0}$ boundary for displacements

$\boldsymbol{\Gamma_1}$ boundary for tractions

$\mathbf{u}_{\boldsymbol{k}}$ displacement vector

$\mathbf{p}_{\boldsymbol{k}}$ traction vector

$\boldsymbol{M}$ Green's functions matrix

F force magnitude at point $\boldsymbol{P}$ at the moment $\boldsymbol{t}$

$\mathbf{F_{aj}}$ mean of the measured force magnitude

$\mathbf{a_{Pa}}$ mean of the force angle computed over the series of images

$\mathbf{M_s}$ simplified mesh (denser around probing point)

INN *n*-nearest neighborhood of point **I** in mesh

$\mathbf{M_r}$ reference mesh

$\mathbf{M_a}$ aligned mesh

$\mathbf{M_{a\_axis}}$ aligned mesh along *y* axis

$\mathbf{P_r}$ closest point $\in$ $\mathbf{M_r}$

$\mathbf{p}(\mathbf{X})$ probability distribution

$\mathbf{w_i}$ $(i = 1, \ldots, N)$ finite number of feature vectors

k ranking of each feature vector

t time step

$\boldsymbol{\alpha}(\mathbf{t})$ learning rate

$\Lambda(\mathbf{t})$ characteristic decay constant

T training length

$\boldsymbol{\alpha_o}$ initial value for $\alpha(t)$

$\boldsymbol{\lambda_o}$ initial value for $\Lambda(t)$

$\boldsymbol{\alpha_T}$ final values for $\alpha(t)$

$\boldsymbol{\lambda_T}$ final value for $\Lambda(t)$

NNX, NNY, NNZ series of two-layer feedforward networks

$\mathbf{X_T}$, $\mathbf{Y_T}$, $\mathbf{Z_T}$ coordinates of the point of intersection **I**

$\mathbf{S_1}$ surface of first mesh

$\mathbf{S_2}$ surface of second mesh

p given point on the surface $\mathbf{S_1}$

$\mathbf{e}(\boldsymbol{p}, \mathbf{S_2})$ distance between **p** and $\mathbf{S_2}$

$\mathbf{d_h}(\mathbf{S_1}, \mathbf{S_2})$ Hausdorff distance

$\mathbf{d_{mean}}(\mathbf{S_1}, \mathbf{S_2})$ mean distance between the two surfaces

$\mathbf{d_{rms}}(\mathbf{S_1}, \mathbf{S_2})$ root mean square error

$\mu_{\mathbf{im1}}$ estimation of mean intensity of image $\mathbf{im_1}$

$\mu_{\mathbf{im2}}$ estimation of mean intensity of image $\mathbf{im_2}$

$\mathbf{l}(\mathbf{im_1}, \mathbf{im_2})$ luminance comparison function

$\mathbf{C_1}$, $\mathbf{C_2}, \mathbf{C_3}$ constants to avoid instability

L is the rang of the pixel values

$\boldsymbol{\sigma}_{\mathbf{im1}}$, $\boldsymbol{\sigma}_{\mathbf{im2}}$ estimation of the signal contrast

$\mathbf{c}(\mathbf{im_1}, \mathbf{im_2})$ contrast comparison function

$\mathbf{s}(\mathbf{im_1}, \mathbf{im_2})$ structure comparison function

$\mathbf{SSIM}(\mathbf{im_1}, \mathbf{im_2})$ structural similarity (SSIM) index

$\mathbf{cov}_{\mathbf{im1im2}}$ covariance of $\mathbf{im_1}$ and $\mathbf{im_2}$

Max maximal error

Mean mean error

Rms mean square err

NN neural network

Ansys Ansys academic software simulator

Net1 feedforward neural network used for training

Net2 feedforward neural network used for prediction

CHAPTER 1

1. INTRODUCTION

Modeling and prediction of 3D deformed objects is an important problem in machine vision and robotics. As such, researchers have proposed methods and solutions to deal with it for various fields such as computer simulation and animation [1, 2], surgical training and treatment [3, 4, 5], robot manipulation [6, 7, 8], mechanical engineering [9] and in other fields [10, 11, 12]. These methods allow studying the behavior of 3D objects by analyzing, simulating, modeling and predicting their deformations.

Among the methods proposed in the literature on this topic, the most popular are: mass-spring models [13,14,15,16], finite-element (FEM) representations [9,11,12,17], surfels [18,19,20,21], mesh-free deformation [22,23,24,25], Green functions [26] and Non-uniform rational basis spline NURBS [27]. In a mass-spring system, the structure of a mesh consists of mass points and springs: a mass is associated with each vertex and a spring with each edge. The method is well established, and several authors used such systems for modeling 3D objects. The resulting models can be simulated in real time [28], but these systems make different assumptions about the material of the object, such as its isotropy or homogeneity [29]. Moreover, the choice of the values of the parameters in this system is carried out manually for most of methods and the distribution of the mass on the vertices of the objects which must be defined a priori.

In the finite element method (FEM), the object is divided into small finite elements, for example into small triangles. These elements are interconnected by nodal points situated on their boundaries. To characterize the state of the displacement over each element, a set of equilibrium equations is defined. These equilibrium equations are derived over each element to characterize the unified behavior of the object. The FEM models have more precision than the mass-spring models [30], but they are computationally expensive and make assumptions about the material of the object [31].

Surface elements method (surfels) models an object as surfels, sample points without explicit connectivity. However, the main drawback is related to the problem of surfels initialization when dealing with natural scenes such as cloth, printed material, textured surfaces such as wood, etc. [32]. These sets are usually selected to satisfy some equilibrium equations [33]. These use only nodes and therefore do not require maintaining connections between them. However, mesh-free methods are often unstable when treatments are required to implement derived boundary conditions [34]. Finally, in the NURBS (Non-uniform rational B-spline) method, the geometrical object complexity is decoupled to provide a low dimensional space describing realistic deformations [35]. However, this method has not been widely applied for representing a 3D scattered non-ordered data set [35].

Most of these methods assume that values of the physical parameters describing the object behavior (e.g., elasticity parameters) are known a priori. Therefore, they cannot be used in some applications in the real life as in the robotics domain. One solution is to estimate the physical parameters by manual tuning until satisfactory results are obtained and use these approximate models. While this can be used for certain types of applications, it cannot be employed where accuracy is expected, as an improper choice of parameters can lead to undesired behavior.

Recently, learning-based approaches have been investigated for predicting 3D object deformation. Among these methods we can mention deep networks, feedforward neural networks, self-organizing maps and neural gas networks. Deep learning architectures require large datasets for training, accompanied by long computations, rendering them unusable for complicated deformed objects in real-time applications [137, 138, 141, 142]. Additionally, some deep learning solutions are sensitive to the object properties, such as its texture [138, 139], or make use of volumetric models that tend to require expansive computing storage [150]. In terms of methods based on feedforward neural architectures, the existing solutions do not predict the entire object shape; they only predict local deformation as 2.5D profiles [140] or predict if a point belongs or not to the surface of a 3D object [145] and, thus, require additional computational effort to recreate the object shape by testing virtually the entire 3D space. Regarding neural gas networks [140, 146]

and self-organizing maps [148, 149], while interesting for reducing the computational cost, they are only appropriate for representing roughly the shape of the object (for example to avoid collisions for a mobile robot). Therefore, they cannot be used independently when accurate object models are desired.

Moreover, to deal with realistic scenarios and real datasets (e.g., robotic applications), novel methods need to follow a complex sequence of steps, namely: 1) acquisition of 3D data, 2) cleaning and simplification of 3D data to improve computational speed, 3) choice of the methods for modeling and prediction, and 4) reconstruction of the predicted 3D object models based on the selected model. In terms of data acquisition, realistic deformation prediction requires experimental measurements acquired through physical interaction with the object in order to capture its behavior when subjected to various forces. These measurements can be carried out based on instrumented indentation tests. Such tests usually involve the monitoring of the evolution of the force (e.g., its magnitude, direction, and location) using a force-feedback sensor. The active indentation probing needs to be accompanied by a visual capture of the deformed object to collect geometrical data characterizing the local surface deformation as a result of the interaction.

Once 3D shape data is acquired, an object model is usually constructed. Modeling of large and complex deformable objects requires in general more computations and also more memory storage which becomes fast a challenge to their deployment in real-time scenarios where computation and memory resources are limited (e.g., robots, drones, etc.). As such, several methods have been proposed for reducing the complexity of 3D objects by reducing the number of faces and vertices of deformed 3D objects, while preserving as much as possible the similarity to the non-simplified object. However, each of the existing techniques has specific issues. For example, random sampling can discard important features of an object, whereas uniform sampling requires the sampling density to be high uniformly over the entire surface to preserve details in the object shape. In sequential sampling, once the object has been sampled, one must verify whether the sequence of selected samples is significant or not. Stratified sampling is a technique that generates samples by subdividing the sampling domain into non-overlapping partitions

(clusters) and by sampling independently from each partition. The issue with this technique is to find a way to efficiently sample from each cluster. Finally, a few techniques have been proposed for sampling 3D objects using unsupervised techniques, such as neural gas and self-organizing maps. However, these depend on carefully selected parameters and tend to take a long computation time to properly identify the points to be sampled. This justifies the interest in developing novel sampling methods or novel sampling method combinations for 3D objects to improve not only the compactness of their representation, but also the quality of sampled data in terms of preserving the main (salient) objects geometric properties.

1.1. Objectives

As briefly described in the previous section, we have identified have three major problems in the state of the art, that we try to address in this thesis. The first problem is the need for an accurate prediction method that can be applied in a variety of real-world applications (e.g., robotics, manufacturing, etc.) involving contact forces with physical objects for which explicit knowledge of the physical parameters is not available. The second problem is related to ensuring that these methods are computationally efficient so as to facilitate their deployment in real-time scenarios where computation and memory resources are limited (e.g., robots, drones, etc.). The third problem is related to the capacity of the methods to deal with the variability of objects in terms of size and physical properties. To address these issues, we outline the following objectives of the thesis:

1) Develop an automatic and universal method for predicting 3D object deformations based on machine intelligence. This method combines an efficient object representation and supervised learning based on neural networks.

2) Build a simplified representation of 3D objects that reduces the number of vertices while preserving the salient geometric features of the objects. This representation leads to a reduction in computation time and in the memory storage.

3) Develop a transfer learning capability to account for the within class variability (shape and material) of objects. This consists mainly in enabling the method to account for different sizes of the same object as well as slightly varying physical parameters of the objects.

Our ultimate goal is to develop a method for modeling and predicting of the deformation of soft objects that does not make assumptions about the material and size of the object. This method must be able to predict as accurately as possible object deformations, while also operate with the minimum cost possible in terms of computing time and memory storage. As such, it can be deployed in real-world applications such as robots exploring their environment in an attempt to learn object properties, predict their deformations without knowing their physical parameters and, as a consequence, efficiently manipulate them.

1.2. Methodology

We briefly summarize the general methodology in this thesis in the following steps:

1) Data acquisition and preparation: This procedure contains four parts, namely: data acquisition, data synchronization, data cleaning, and data alignment. It is important to mention that this step is required for collecting real data over 3D object. However, beyond collecting real data, we have also made use of modeling software to generate additional data for 3D objects with different material and sizes for training and testing our system in a more comprehensive and rigorous way.

2) Object simplification: This procedure contains three steps: simplification of the mesh, reducing the number of vertices using inspiration from stratified sampling and a fitting procedure based on the neural gas algorithm.

3) Model selection: Several architectures of feedforward neural networks are tested to select an appropriate model for each object class (shape and material). We generated training examples for each object by applying different magnitudes of point forces to the object at rest and measured the corresponding deformation induced by the force.

4) Prediction: After training a selected neural network for each object, we used it to make predictions on new instances of deformations of the same object class by applying different force magnitudes. We showed in several experiments that our method is able to efficiently predict not only the deformation of elastic objects, but also the one of elasto-plastic and rigid objects. We tested several classes of objects and compared the performance of our method with a classical method, the FEM.

4) Transfer learning: This step is aimed at minimizing the training phase when dealing with similar object classes (i.e. objects having the same shape and/or material). We argue that a learning produced by training a model to predict the deformation of a given object class can be exploited to reduce the computational time of model training for similar object classes (e.g., objects with the same material with different size, objects of the same size but with different material). This can be achieved by fine tuning the original model to adapt it to the new object class.

To analyse the performance of our solution, we reconstructed the deformed 3D objects after each step of the proposed method and computed two categories of error measures: Metro errors [36] and perceptual errors [37], in additional to a visual comparison that allowed us to detect possible distortions in the object shape.

1.3. Contributions

The contributions of this thesis revolve around the outlined objectives, and can be summarized in the following three points:

1. We propose an automatic and accurate method for predicting 3D soft object deformations that does not make assumptions about the material of the objects (Objective 1). We capitalize on computational intelligence, namely on selectively-densified simplification, stratified sampling, neural gas and feedforward neural networks. The proposed method avoids recovering elasticity parameters which cannot be precisely and accurately identified for certain materials such as foam or rubber, as those used in the context of this work. Therefore, we eliminate the need to make assumptions on the material, such as its homogeneity or isotropy,

assumptions that are often encountered in the literature. This is achieved through the use of several configurations and architectures of feedforward neural networks that are trained to map the complex relationship between the force and the corresponding position of neural gas nodes over each cluster. Once trained, the networks can be used to predict deformations for forces (magnitude, angle of application and point of application) for which the network has not been trained.

2. We propose a preprocessing method to create compact 3D object representation by innovatively combining selectively-densified simplification, stratified sampling and neural gas fitting (Objective 2). The selectively-densified simplification technique allows us to reduce the computation time by simplifying only the non-deformed parts of the object, while the subsequent stratified sampling technique further reduces the time while preserving as much as possible the similarity with the original non-simplified object. Finally neural gas fitting allows us to capture accurately the fine differences around the deformed zone and over the surface of the object that are missed by the previous sampling steps. This preprocessing reduces the complexity of the deformed objects by creating compact and accurate representation and thus brings savings in computation time for the subsequent object modeling.
3. We improve our prediction method using the transfer of learning between neural networks models. This allows us to use the learning of a pre-trained network on one object to accelerate the training of another object given that the physical properties of the objects are similar except for their size and/or slight geometrical differences (Objective 3). Therefore, we can predict the deformation of a small ball from training on large ball and vise-versa. Transfer of learning also allows us to use less data to train the second neural network, and therefore reduce the training time.
4. We analyze the performance of our method on 3D objects with various shapes, sizes and materials. We demonstrate that, while our method with the automated choice of parameters is much faster to predict deformations than FEM (a desired ability for real-time applications where the decision time is important), the models are slightly less accurate than the ones obtained by using FEM. However, we have

also demonstrated that by modifying the selectively-densified simplification step we can reach the same performance as FEM if that is preferable for a specific practical application, at the price of a slightly higher computation time, but still lower than the one used by FEM.

1.4. Thesis Organisation

This thesis is organized as follows. Chapter 2 discusses the relevant literature on the topics of interest for this thesis. It provides a brief discussion of 3D object deformation methods, of vision-based measurements to collect data and of the reconstruction of 3D objects from collected data. It also provides an overview of methods for modeling 3D deformable objects, clustering and sampling methods. This chapter sheds the light on the several issues and problems that are solved or covered by the existing literature.

Chapter 3 presents our proposed method for modeling and prediction of 3D objects. We discuss the proposed framework and the different steps required to implement it. Among these steps, we will consider: data simplification, data alignment, sampling (clustering) and selectively-densified mesh construction using a formulation of neural gas fitted simplification and neural network prediction of neural gas nodes position. We will also introduce our solution for merging QSlim, a classical, efficient, uniform simplification algorithm with neural gas networks in order to construct more accurate and compact object models based on the inherent capability of neural gas to concentrate on regions of interest over the surface of an object. This hybrid solution allows to create compact models (QSlim), while also maintaining a high resolution and thus a better quality of the model in deformed areas over the surface of the object (neural gas).

We then present the proposed prediction method for the behavior of deformable 3D objects under interaction using feedforward neural networks. The use of these networks is an efficient way to provide real-time estimates while using raw and noisy data, as opposed to using different approximations and assumptions about the object material, which are nowadays found in the literature. A rationale is provided for the various choices of architectures studied. Finally, the issue of transfer learning between

feedforward neural networks is also studied to enable the use of pretrained network to make predictions for other objects.

Chapter 4 presents experimental results obtained by using our proposed method as well as an in-depth study on the impact of various parameters involved (i.e., number of nodes in the neural gas network, learning rate, feedforward network architecture, etc.). It presents the modeling and prediction results obtained as well as a comparison with the FEM method to demonstrate the performance of our method with respect to a classical existing solution. The conclusions along with the future work are presented in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

2. Introduction

The interest areas of this thesis cover three major directions of research, namely methods for data acquisition and reconstruction of 3D deformable objects from collected data (sections 2.1, 2.2 and 2.3), modeling of 3D deformable objects (section 2.4) and the prediction of object deformations (section 2.5). Research related to our work has been carried in a number of fields with varying goals, starting from physics-based deformable models to computer vision, computer graphics, computer simulation and animation, engineering, mechanics, surgical training and treatment, etc. In this thesis, we focus on the acquisition and modeling of soft deformable 3D objects and the subsequent prediction of their deformation when subjected to forces of various magnitudes and angles of application, at various locations over their surface.

Current research in the area of deformable objects is in general focused towards the modeling of inherently elastic objects. Researchers concern themselves with simulations meant to address issues related to the computational cost of increasingly complex models and the requirements for realistic interaction with deformable models, and less with accurate object modeling. As a response to the difficulties encountered in conducting strain-stress relationship measurements for objects made of materials that exhibit nonlinear behavior, or when a model is available for the object's material, but its material properties are not known, the choice for the selection of elastic parameters is left to the user. The latter can choose some values for these parameters according to some a priori knowledge regarding the deformable object model. Usually, an expert is asked to observe an object and provide a feedback on the nature of deformation that the object exhibits. This is a subjective measure and while it can be used for certain types of applications (i.e.

gaming), it cannot be employed where a certain degree of accuracy is expected (i.e. virtual training, robotic manipulation, etc.).

There are four main categories of solutions encountered in the literature for acquiring the deformation of objects: indentation, vibration-based measurements, sound-based measurements, vision-based measurements and combinations of these. The most popular for daily engineering problems are indentation and vision-based measurements and they will be thus used in the context of this thesis. The principle employed is to apply a deformation force (indentation) and make use of vision sensors to capture the deformed object surface as a result of force application.

2.1. 3D Object Deformation by Indentation

The deformation of an object is created using instrumented indentation tests, which are meant to elicit the elastic behavior of objects [47]. These tests usually imply monitoring the evolution of the applied force using a force feedback sensor and a visual sensor to capture the surface deformation resulted as a consequence of the applied force. Several researchers in the literature make use of this approach to deform 3D objects in various fields of research, such as medical simulation, mechanics and engineering, computer graphics and several other fields.

Frank et al. [6] use a physical system consisting of a mobile platform with a 7-degree of freedom (DoF) manipulator that is equipped with a force-torque sensor and a depth camera. The force-torque sensor allows to deform the object and the depth camera to observe the object from different viewpoints and measure the corresponding deformation forces.

In the work of Petit et al. [30], an object is deformed by a single contact force which acts vertically and is applied by a tool mounted on a robotic arm, equipped with a force sensor at the wrist. They assume a pointwise contact on a point of known position on the surface of the object.

Following indentation probing, two main interaction point detection methods are proposed in the literature to represent the contact between a probing tip ad the surface of the object when a force is applied, namely the single point-based method and the ray-

based method. In this first method, the interaction between the tip and the object is considered as a single point [48,49]. In the second method, more than one interaction point is considered between the tip and the object [50,51,52,53].

Most of the modeling techniques developed in the literature use the point-based method in which the probe is simply represented as a single point [54] (for more details see chapter 3, sections 3.2.1 and 3.2.1). Some modeling techniques use the ray-based method that considers that the interaction between the probe and the object occurs at several points [50,51,52,53]. Srinivasan et al. [52] used two interaction points that move and exchange forces along the line that connects them. McNeely et al. [53] modeled the probe as a set of surface points. In their approach, objects are divided into voxels. Multiple collisions are detected between the surface points of the probe and voxels of the object to reflect forces. Then, the force acting on the object is obtained as the sum of all forces from such point-voxel intersections. The disadvantage of this method is that it is difficult to model thin or small objects. To handle the ray-based method in the simplest possible manner, Ho et al. [50] consider the interaction between the probe and the object as being represented by a number of interaction points forming a polygon or a line depending on the movement of the probe. The disadvantage of this method is that it is more complicated and more expensive in terms of computation than the single point-based method.

2.2. Methods for 3D Visual Data Acquisition

To capture the geometry of an object, the indentation procedure needs to be accompanied by the observation of the deformed object surface. Several methods were used to acquire 3D data from deformed objects, including the acquisition of data from images [55,56,57,58], the acquisition of data using laser scanners [59,60,61,62] and using depth cameras, such as the Kinect [55,56,57,63]. These will be described briefly in the next sections.

2.2.1. Methods Used to Estimate 3D Data from Images

The problem of estimating 3D data from images has been the main focus in computer vision and in other fields over the years. It is the process of finding 3D information from

2D data. As images are taken around the target object from various viewpoints, some points on the object surface can be visible only in some of the views which can cause difficulties when reconstructing the object shape.

Rohe et al. [55] introduced a technique to extract 3D objects from a set of images acquired by a single camera using finite element technique. They created synthetic finite element images with arbitrary points of view.

Merras et al. [56] present a technique for collecting 3D data of a human face from a sequence of images. Their method is based on the estimation of the projection matrices of the cameras to determine the 3D point cloud and then reconstruct the 3D mesh of the face.

To collect 3D data from objects, Zhang et al. [57] propose a method based on an un-calibrated image sequence captured orderly around a scene. They rotate a platform with a fixed camera to capture 2D images. After establishing a number of key-points in the images for each two images selected, the points are matched, and a dense 3D model is reconstructed.

Ze-tao et al. [58] extract feature points from images using the Harris operator and then match corresponding points between images. To determine the coordinates of 3D points, they calculate the fundamental matrix with extrinsic and intrinsic parameters of camera.

In their work, Lin et al. [64] capture images from arbitrary viewpoints using a virtual multi-camera system. The images are captured by a virtual camera with the position and orientation synthesized from the locations of the physical cameras and allow for the subsequent 3D model reconstruction.

Ding et al. [65] make use of a digital camera that is turned around an object of interest to acquire a series of images from a number of locations and at various angles. Several matching pairs of images are then employed to reconstruct the corresponding 3D cloud points.

Fritsch et al. [66] use multiple cameras mounted on a rectangular shaped frame in order to collect images from multiple views at once. By using an automated software pipeline,

they compute a 3D point clouds for each image. Then, the 3D point clouds obtained from all the images taken from the multiple views are registered and connected.

Wang [67] read and arrange 2D image slices to be able to obtain 3D data. The process starts with the identification of matching pixels in each two closest images as being the projection of the same point. All the images are projected into a common space prior to analysing the correspondence between the images, and then triangulation is employed to reconstruct the 3D points.

Huang et al. [68] present an approach for automatic 3D reconstruction of objects depicted in web images. They analyze jointly a collection of images of different objects along with a smaller collection of existing 3D models, where the latter guides the reconstruction process. The images are analyzed and reconstructed together to create 3D models. The main problem with this type of solution is that it requires a huge amount of storage space and analysis time to rebuild each object and sometimes it does not provide accurate object models.

2.2.2. Acquisition of 3D Data by Laser Scanners

There are many different 3D scanning technologies each with its own limitations, benefits and costs [69, 70, 71, 72]. 3D laser scanning is another well-known technique used to acquire 3D data.

Lee et al. [59] apply a terrestrial laser scanning technology for collecting 3D data in an indoor environment. Similarly, Cheng et al. [60] discuss a method for modeling buildings using laser-scanning technology in which a building is separated into several stations and each station is scanned separately in order to capture its shape and contents. A software is then used to register the resulting data and a mesh is built on the point cloud to construct the 3D building model.

Bartol et al. [61] provide a comprehensive review of 3D scanning technologies and shape estimation for body measurements. Li et al [73] developed a technique to acquire 3D data for the human body by laser scanning. The data is registered, denoised and simplified before rebuilding the body surface model.

Buonamici et al. [74] present a system for 3D scanning of human arms. Their system is composed by a 3D optical scanner based on stereoscopic depth sensors and by an acquisition software responsible for the processing of the raw data.

Lin et al. [75] propose a method that uses a robotic hand to pick up non-rigid objects resting on a table. The configuration consists of a Barrett hand and a 3D laser scanner. Navarro-Alarcon et al. [76] develop a strategy to control the 3D shape of non-rigid objects by capturing visual characteristics which quantify the deformation. They used two robots that hold the object while deforming it and a stereo vision system built with two sensors.

2.2.3. Acquisition of 3D Data by Kinect Camera

A cheaper alternative to laser scanners for capturing 3D data are depth cameras, such as Kinect cameras. A method for acquiring 3D data by using multiple Kinect cameras is presented by Ruchay et al. [77]. They capture two types of data by each Kinect, the depth information in the form of point clouds (a set of points with 3D coordinates) and the color images. Then, they map the coordinates between the data coming from each Kinect for registering the depth and color images to reconstruct the 3D object.

Gupta et al. [78] developed an indoor mapping system for data collection in a building environment. They used a Kinect Sensor and a Stereo Labs ZED Camera. In their experimentation, the results of indoor mapping obtained by the Kinect sensor and the ZED camera are compared in terms of speed and memory usage. The ZED camera is faster at recording high-resolution depth maps than the Kinect, whereas the Kinect camera requires less amount of memory space than the ZED camera to store the point cloud.

The work in [79] uses depth images acquired by a Kinect sensor from different viewpoints to generate point clouds and makes use of the iterative closest point (ICP) algorithm to register two adjacent point clouds in order to create a full 3D object model.

Xu et al. [80] present a system to build complete 3D models using the Kinect sensor and a rotatable platform. The Kinect is fixed on a tripod. The rotatable platform is controlled by a motor at a constant speed. The authors calibrate the relationship between the

rotatable platform and the Kinect and thus derive the parameters used for registering partial point clouds in real time. Finally, the Poisson reconstruction method is used to construct the mesh surface from collected point clouds.

Delgado et al. [82] used a platform consisting of a robot hand equipped with a Tekscan tactile sensor on the fingertips and a Kinect camera with an eye-in-hand configuration. Their collected visual data from the Kinect sensor are used to detect the deformation.

2.3. Reconstruction of 3D Objects from Collected Data

Several methods were used to reconstruct objects from 3D point clouds. Rodriguez et al. [83] developed an algorithm for accurate real-time reconstruction of 3D deformable objects using a single Kinect sensor. Their reconstruction process consists of the following steps: capture in time of RGB-D information with a Kinect sensor, registration using a modified iterative closest point algorithm, and dynamic construction and refinement of a dense 3D object model.

Chen et al. [84] developed an algorithm to reconstruct a 3D object from cloud points based on the idea of region growing. From their side, Ruchay et al. [85] propose a new method for dense 3D object reconstruction using an RGB-D sensor at high rate. In order to obtain a dense shape of a 3D object, they used the Iterative Closest Point algorithm [181].

Wiemann et al. [86] present an approach to compute polygonal reconstructions from arbitrarily large point clouds. Their approach mainly concentrates on the limitation of the memory that usually occurs when handling large scale point cloud data on standard computers. They serialized the data into partitions on a shared storage into geometrically coherent chunks. These partitions are then sent to the slave nodes, which perform the necessary computations on their assigned parts.

Osmanković et al. [87] used combined information captured by a thermal camera and a laser scanner to reconstruct a 3D model of the environment for a mobile robot. Other authors build deformable object models from pure kinematic motion trajectories [186]. The expectation-minimization algorithm finds the most probable node positions for the model given the measurements. Most of the experimentation is performed in a controlled

environment against a green background, which limits its applicability to real-world conditions.

Wang et al. [88] address the issue of reconstruction and tracking of deformable objects and propose an approach for non-rigid reconstruction with a Kinect camera. A depth scan is performed using a Kinect sensor. The depth sequence is divided into several local segments, and a global optimization method is employed to merge all the segments. Finally, a partial 3D model of the object of interest is generated using Poisson surface reconstruction.

Li et al. [89] present a novel and highly efficient framework to generate dense point clouds for representing the 3D shape of deformed objects. Given a single image of an object of interest, taken from an arbitrary viewpoint, their method can generate multiple partial surfaces of the object. The final 3D object representation is obtained by fusing the point cloud surfaces into a single point cloud.

2.4. Simplification, Clustering and Sampling Methods in the Context of 3D Object Modeling

The simplification of an object involves creating a simpler version of the original object, using fewer elements (faces and vertices). The aim is to obtain a simplified representation in order to improve the processing time, while preserving as much as possible the similarity with the original non-simplified object.

In this context, Garland et al. [90] developed a surface simplification algorithm for polygonal models based on iterative contractions of pairs of vertices.

Guéziec [40] present a technique for simplifying a triangulated surface by approximating the surface with another surface of lower triangle count, while taking into account the approximation error with respect to the original surface.

Clustering and sampling are two important methods often used in 3D object modeling for reducing the complexity of the deformed objects, which in turn leads to a reduction in the computation time.

Clustering is a process in which a group of data are partitioned into a number of clusters, in which, ideally, similar [91,92] data are assigned to the same cluster, and dissimilar [93,94] data are assigned to different clusters. Several algorithms have been proposed in the literature for clustering [63, 95, 96, 97] for applications such as medical image processing [98, 99], biomedical [100], classification [101, 102] and data mining [103, 104].

Sampling is a method used to choose sample points from data with the aim to reducing its complexity. There are several sampling methods in the literature, among these methods we can mention: random sampling [41], sequential sampling [42], uniform sampling [43], systematic sampling [44], and stratified sampling [105].

In random sampling, all points representing an object have the same probability of being chosen at any stage during the sampling process. Although this type of sampling is easy to understand, it is not easy to implement for large datasets and randomly-selected samples can miss important features of an object.

In sequential sampling, a sequence of samples is taken from an object. Once the object has been sampled, it is verified whether the sequence of samples taken is significant or not. When the sequence of samples extracted is not significant, the whole procedure is repeated. In this method, one doesn’t know in advance how many times the process must be repeated to acquire sufficient data to represent an object.

In uniform sampling, samples are spread such that the probability of a surface point being sampled is equal for all surface points. This method has the advantage of being easy to implement and it covers all the surface of the object. However, its cost is high because when a high density is required in some regions on the surface of the object (regions with minute details), the sampling density must be uniformly high over the entire surface.

In systematic sampling, sampled points are chosen from the data collected by selecting random starting (seed) points. Once the number of these samples is decided, the entire population size is divided by this number. Then a number of points from the data collected must be assigned to each member of the sample points. These points assigned for each member are chosen based on the distance to that member. This method is easy to

implement but the disadvantage is that the results achieved depend on the selection of the random starting points.

Sometimes, data are divided into different groups or clusters sharing the same characteristics. This is where it is important to divide data into clusters before sampling. In stratified sampling, samples are collected by subdividing the domain into non-overlapping clusters [105], and then by sampling independently from each group. Using this method, one can ensure that an adequate sampling is applied to all groups in the partition. This method ensures a high degree of representativeness of all the clusters in the domain. However, it requires a prior knowledge about the areas to be sampled at higher resolution, if desired, in the application.

Rodolà et al. [106] used a technique of sampling based on the concept of relevance. They define the relevance for each point over a mesh taking into consideration how similar the point is to the points around it (i.e. similar points, more distinctive points, and less relevant points). Similar points are considered those having a unique normal orientation with respect to their surroundings, most distinctive points correspond to ridges or corners, and the least distinctive ones are those in flat areas.

Turk [107] used a uniform sampling technique that produces samples evenly distributed over the surface of an object. The obtained sample points are re-triangulated to produce a simplified version of the original mesh.

In their work, Ghosh et al. [43] present a method based on uniform sampling. Their method first decomposes the model into approximately convex pieces and then uses uniformly sampling on the surfaces of these pieces.

Nehab et al. [105] use a stratified technique for modeling 3D objects. Their algorithm is divided into three steps. The first step is the voxelization of the model. The next step produces one sample for each voxel. They register all triangles in the interior of each voxel bounding box and on triangle is chosen from each voxel, and from this triangle a new sample is produced. To compute the sample positions, they integrate the probability density function over all area of the model. The final step constrains the minimum distance between samples by removing samples that are too close to each other.

Finally, Shih et al. [108] proposed three novel non-uniform sampling algorithms: adaptive sampling, local adjustment sampling, and finite element centroid sampling. Their adaptive sampling algorithm makes use of a recursive surface subdivision process. In local adjustment sampling, a set of predefined starting points are chosen, and the local optimum position of each nodal point is found. In the finite element sampling, the centroids of the surface triangle mesh produced from the finite element method are considered as representative samples. The authors compared the three algorithms and concluded that adaptive sampling is better for different classes of objects.

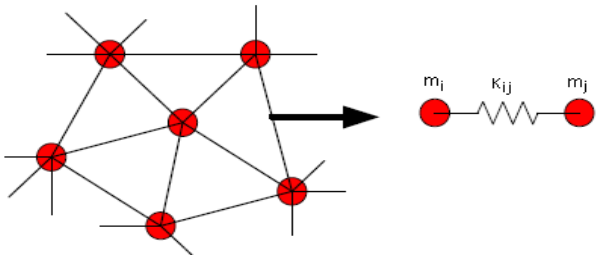
2.5. Modeling of 3D Deformable Objects

Several solutions have been proposed in the literature for modeling deformable objects. These include mass-spring models [13,14,15,16], finite-element (FEM) representations [109,110,111,112], Green functions [26], Non-uniform rational basis spline (NURBS) representations [27], surfels [113], mesh-free deformation [114] and subvoxel triangulations [115]. These techniques are described in more detail in the following sections.

It is important to mention that most of these techniques assume that the parameters describing the object behavior (e.g., elasticity parameters) are known a priori or values for these parameters are chosen by manual tuning until the results seem plausible. This is a subjective process and while it can be used for certain types of applications, it cannot be employed where accuracy is expected, as an improper choice of parameters can lead to undesired behavior. To overcome this problem, several researchers propose estimating the parameters based on a comparison between the real and simulated object subject to interactions. This is the approach that we will be using in this work.

2.5.1. Mass-Spring Systems for Modeling Surface Deformation

A mass-spring system is a model described as a graph with a node at each vertex and a spring along each edge. Generally, all the vertices are connected by springs and a mass is assigned for each vertex. As such, the structure of a mesh consists of mass points and springs (Figure 2.1 a). The edge connecting the two vertices with mass $\boldsymbol{m_i}$ and $\boldsymbol{m_j}$ is represented by a spring with stiffness $\mathbf{K}_{\boldsymbol{ij}}$ (Figure 2.1 b).



(a) **(b)**

Figure 2.1. (a) Structure of a mesh, and (b) spring between two vertices (adapted from [1]).

When external forces are applied on the object, all the vertices in the region where these forces act are displaced which, in turn, creates spring forces in the object. These spring forces propagate inside the object and finally the springs come to rest. In physics, it is very important to make sure that springs do not oscillate forever but come to rest over time. To ensure this behavior and ensure realistic simulations of deformable objects modeled as mass-spring systems, dampers are combined to springs. The internal force of a spring between two vertices $\boldsymbol{V_i}$ and $\boldsymbol{V_j}$ is given by [116]:

$$\mathbf{F}_{ij} = \mathbf{K}_{ij}(|\mathbf{p}_j - \mathbf{p}_i| - \mathbf{l}_{ij})\frac{\mathbf{p}_j - \boldsymbol{p}_i}{|\mathbf{p}_j - \mathbf{p}_i|} \tag{1}$$

where $\mathbf{K}_{ij}$ is the stiffness coefficient of the spring, $|\mathbf{p}_j - \mathbf{p}_i|$ is the magnitude of the displacement of the current state of the spring, $\mathbf{l}_{ij}$ is the rest length of the spring. To simulate the effect of an external force applied on the object, one can make use of the Lagrange's equations [117]:

$$\mathbf{F}_{external} = \mathbf{f}_{result}(\mathbf{x}_i) + \boldsymbol{\gamma}_i\dot{\boldsymbol{x}}_i + \mathbf{m}_i\ddot{\boldsymbol{x}}_i \tag{2}$$

where $\mathbf{F}_{external}$ is the external force applied at each vertex $\boldsymbol{i}$. Three different forces, namely the elasticity force, the curvature force and the constraint force are summed to form $\mathbf{f}_{result}(\mathbf{x}_i)$ [117]. The symbols $\mathbf{x}_i$, $\dot{\boldsymbol{x}}_i$ and $\ddot{\boldsymbol{x}}_i$ designate respectively the positions,

velocities and accelerations of the $\boldsymbol{i}$th vertex . $\mathbf{m}_{\boldsymbol{i}}$ are the nodal masses and $\boldsymbol{\gamma_i}$ are the damping factor. The equations of motion for the whole mass-spring system are given by [2]:

$$\boldsymbol{F = Ku + D\dot{u} + M\ddot{u}} \tag{3}$$

where $\boldsymbol{F}$ is the total of external forces on the nodal points, $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{D}$ is the matrix of damping coefficients, $\boldsymbol{K}$ is the stiffness matrix, and $\boldsymbol{u}$, $\boldsymbol{\dot{u}}$ and $\boldsymbol{\ddot{u}}$ are, respectively, the displacement, velocity and acceleration vectors.

2.5.1.1. Applications of Mass-Spring Models

Mass-spring models were employed in many applications for shape modeling [118], computer simulation and animation [1,2] and in surgical training and treatment [3,4,5]. Closer to the topic of this thesis, several authors make use of mass-spring systems for modeling 3D objects.

Zaidi et al. [13] model 3D deformable objects represented mass-spring systems in an interaction with a robot multi-fingered hand. The springs are considered linear, but the global behavior of the deformable objects is nonlinear due to large displacements and rotations. The contact forces are generated according to the relative positions between the fingertips and the boundary surface of the deformable object mesh.

Arnab et al. [116] study the feasibility of employing a surface mass spring system capable to emulate soft volume behaviour. They introduce volume springs for which stiffness is extracted on the surface mesh based on the node, the neighbouring triangular elements and the object centre.

In [120], the stiffness properties of mass-spring models are estimated by applying force constraints at different locations on the object, recording the resulting displacements and comparing the reference and the estimated model using particle filters. In a similar way, Choi et al. [121] track the global position of moving deformable balls painted in red against a blue background in a video stream. They adjust the elasticity parameters of a mass-spring model of the ball by optimizing the differences between the object model and the real object.

2.5.1.2. Advantages and Disadvantages of Mass-Spring Models

Mass-spring models can be easily implemented because the physics of these models is simple and can be understood easily. Moreover, these models can be simulated in real time [28], but involve assumptions about the material of the object, such as its isotropy or homogeneity [29]. Compared with FEM models described in the next section, mass-spring models are easier to implement and effective in computation, but their accuracy is in general significantly lower [122].

When using mass-spring models, problems can be encountered when choosing the system parameters. Properties like the length and stiffness of strings, the mass distribution on the vertices of the object and the topology of the mesh have to be defined a priori. Choosing values for these parameters is a difficult and lengthy trial-and-error process as the results vary widely for various settings.

2.5.2. Finite Element Method (FEM)

In this method, the object is divided into many small finite elements interconnected by nodal points situated on their boundaries (see Figure 2.2). A set of interpolation functions called equilibrium equations is defined and used to characterize the state of the displacement over each element. After deriving these functions over each element, they are assembled to characterize the unified behavior of the object. The equilibrium equations are given by [123]:

$$\boldsymbol{F} = \boldsymbol{K}\boldsymbol{u} + \boldsymbol{D}\dot{\boldsymbol{u}} + \boldsymbol{M}\ddot{\boldsymbol{u}} \tag{4}$$

where $\boldsymbol{F}$ *is* the composite vector of forces that act on the object, $\boldsymbol{K}$ is the stiffness matrix, $\boldsymbol{D}$ is the damping matrix, $\boldsymbol{M}$ is the mass matrix of the structure, $\boldsymbol{u}$ is the composite vector of displacements, $\dot{\boldsymbol{u}}$ is the vector of the nodal point velocities (the first derivative of $\boldsymbol{u}$) and $\ddot{\boldsymbol{u}}$ is the vector of the nodal point accelerations (the second derivative of $\boldsymbol{u}$). The damping matrix constructed using the mass matrix and stiffness matrix [124] is defined as:

$$\boldsymbol{D} = \boldsymbol{\alpha}\boldsymbol{M} + \boldsymbol{\beta}\boldsymbol{K} \tag{5}$$

where $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are the parameters for damping matrix.

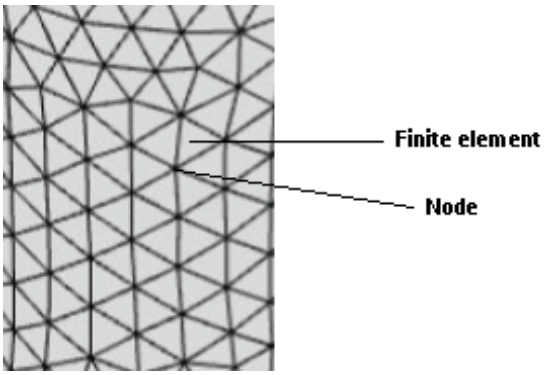


Figure 2.2. Each small triangle represents a finite element.

2.5.2.1. Applications of Finite Element Method

FEM can be used in many applications, such as surgical simulations [125,126], animation [17], mechanical engineering [9] and other fields [10, 11, 12]. Several authors have used the FEM for modeling 3D objects. The authors in [127] present a nonlinear FEM model formulated with Green strain tensor for simulating large deformation and deformations with rotation.

To track the geometry of a deformable object, Wuhre et al. [128] combined a volumetric elastic model with a tracking-based approach to improve the estimation of the unobserved deformable side of the object. When a force is applied to the object, a linear FEM is employed to model physical deformations.

Petit et al. [129] propose to track 3D elastic objects in RGB-D data. A rigid transformation from the point cloud to a linear tetrahedral FEM model representing the object is estimated based on the iterative closest point (ICP) algorithm. Linear elastic forces exerted on vertices are computed from the point cloud to the mesh based on closest point correspondence and the mechanical equations are solved numerically to simulate the deformed mesh.

The authors of [130] present a genetic-based solution to identify stiffness properties of mass-spring-systems by using a linear FEM model as a reference. While the method seems to support isotropic and anisotropic reference models, only simulated results are

presented. The material is isotropic when the elastic behavior is uniform in all orientations. The anisotropy varies systematically depending to the orientation.

In order to determine the elasticity properties of deformable objects, Frank et al. [6] propose to minimize the difference between the real object in interaction with a force-torque sensor and captured as a point cloud and the simulated object in the form of a linear FEM model. However, their method only works on homogeneous and isotropic materials. These methods justify the interest in the development of new methods that do not make assumptions on the material of the object, such as the one proposed in this thesis.

2.5.2.2. Advantages and Disadvantages of FEM Models

In general, FEM models have more precision than the mass-spring models [30]. With the FEM, one can easily model complex geometries and irregular shapes [131]. However, the main limitation of the FEM is the fact that it is computationally expensive [31].

2.5.3. Surface Elements (Surfels)

A surfel is a point sample of an object surface that comprises geometric attributes [18]. Surfels are thus sample points without explicit connectivity. Each comprises several attributes like the depth, texture color, displacement mapping and normal. Different levels of texture colors can be pre-filtered and stored per surfel.

2.5.3.1. Applications of Surfels Method

Several authors used the surfels in the literature. Pfister et al. [18] propose a surfel-based method for rendering objects with rich shapes and textures. They sample the surfaces of complex geometric models to enable conversion of geometric objects and their textures to surfels. During the sampling process, the computation of some attributes as texture, bump, or displacement mapping is performed. Andersen et al. [19] reconstruct the geometry and object surface by representing it as a collection of surfels, associated with each data point. Ruggeri et al. [20] develop a 3D object retrieval system, which comprises a surfelization technique for converting polygonal mesh models into a corresponding surfel-based representation. Carceroni et al. [21] model multi-view scenes captured from

video streams as a set of surfels, each surfel being described by its shape, texture, bump map and motion. They showed that, given camera parameters and knowing the position of light sources, they can recover the surfel parameters.

2.5.3.2. Advantages and Disadvantages of Surfels Method

Surfels are well suited to modeling dynamic geometric objects. Unlike FEM and mass-springs models, there is no need to compute topology information, such as connection between vertices. In addition, every geometric model in form of a polygonal mesh can be converted into surfels [132]. However, problems might be encountered with the initialization of surfels. They also require the scene surface to be sufficiently textured. Unfortunately, this quality is difficult to find in natural scenes (cloth, printed material, textured surfaces such as wood, etc.) [32].

2.5.4. Mesh-Free Methods

Mesh-free methods are commonly used in applications in engineering [23], in electromagnetics [24], and in computer graphics and animation [25]. These methods use sets of points scattered on the boundaries of the object (Figure 2.3). These sets of points represent the problem domain and its boundaries and are carefully selected to satisfy automatically the equilibrium equations [33]. These methods do not require a known relationship between the nodes [22,133]. In terms of disadvantages, they are often unstable when treatments are required to implement derived boundary conditions [34].

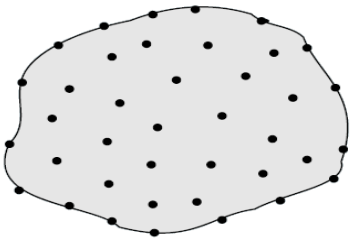


Figure 2.3. Sets of nodes scattered on the boundaries represent the problem domain and its boundaries for mesh-free object representations.

2.5.5. Green Functions

The estimation of deformable models can be based on discrete Green functions. This technique is used to model the global deformation of a solid by a discrete Green function matrix. Supposing that the domain of the problem is a solid $\boldsymbol{T}$, with its boundary $\boldsymbol{\Gamma}$, and supposing that we have two boundaries prescribed $\boldsymbol{\Gamma_0}$ and $\boldsymbol{\Gamma_1}$, the boundary $\boldsymbol{\Gamma_0}$ is for the displacements and the boundary $\boldsymbol{\Gamma_1}$ is for the tractions (see Figure 2.4), Green's functions are employed to relate a field of displacement vectors $\boldsymbol{u}$ to a field of traction vectors $\boldsymbol{p}$ on the boundary of the elastic solid [26].

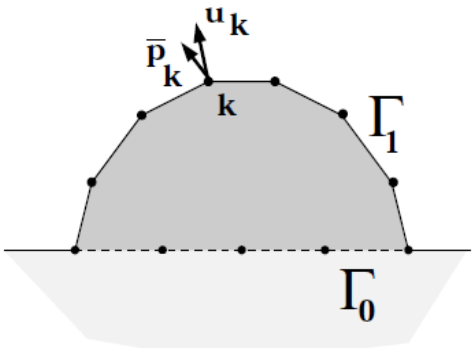


Figure 2.4. Green function representations: on the boundary $\boldsymbol{\Gamma_0}$ displacements are prescribed and, on the boundary, $\boldsymbol{\Gamma_1}$ tractions are prescribed (adapted from [26]).

The displacement and traction vectors at each vertex are denoted by $\boldsymbol{u_k}$ and $\boldsymbol{p_k}$, respectively. These vectors are combined into block vectors $\boldsymbol{u}$ and $\boldsymbol{p}$, respectively. For a given boundary, the block vectors $\boldsymbol{u}$ and $\boldsymbol{p}$ can be rearranged such that all prescribed values are collected in a block vector $\overline{\boldsymbol{v}}$. Entries $\overline{\boldsymbol{v_k}} = \overline{\boldsymbol{u_k}}$ if the vertex $\boldsymbol{k}$ is on the surface $\boldsymbol{\Gamma_0}$, while $\overline{\boldsymbol{v_k}} = \overline{\boldsymbol{p_k}}$ if the vertex $\boldsymbol{k}$ is on the surface $\boldsymbol{\Gamma_1}$.

We introduce the complementary displacement and traction vectors into a block vector $\boldsymbol{v}$. The matrix relating the prescribed values $\overline{\boldsymbol{v}}$ and the block $\boldsymbol{v}$ is the discrete Green's functions matrix, supposing this matrix is $\boldsymbol{M}$. $\boldsymbol{v}$ and $\overline{\boldsymbol{v}}$ are related by:

$$\boldsymbol{v} = \boldsymbol{M}\overline{\boldsymbol{v}} \tag{6}$$

The Green's functions matrix $\boldsymbol{M}$ is determined from displacement and traction boundary conditions. In simulation, a discrete Green's function and the input vector of prescribed

tractions and displacements allows one to calculate the deformed shape of an object as well as the reaction forces [26].

2.5.6. Non-Uniform Rational B-Spline (NURBS)

A non-uniform rational B-spline (NURBS) based deformation function allows to decouple the geometrical object complexity, providing a low dimensional space to describe realistic deformations [35]. NURBS allow control of curvatures defined by control points and weights. Some sections of a defined shape can be shortened or elongated relative to other sections (Figure 2.5).

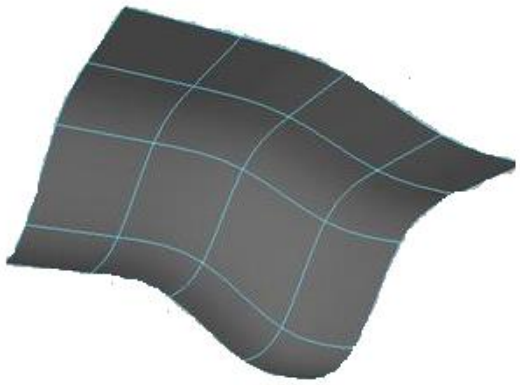


Figure 2.5. Non-uniform rational B-spline (NURBS) (adapted from [134]).

The main idea for modeling 3D objects using NURBS [95] is to define an initial 3D mesh object and deform it subsequently, by matching it with the depth and color data extracted. Unlike mass-spring and FEM methods, the deformation is not described by a displacement for every vertex in the mesh, but by using a deformation function that allows realistic approximation of the object surface [27]. One can create a NURBS surface function to track the deformation, then make an approximation of the object surface with this function. This process involves the minimisation of the distances to the NURBS surface of all mesh vertices. After the NURBS surface is fitted to the 3D object model, vertices are registered to the NURBS surface function.

B-Spline surface modeling possesses many properties such as boundedness, continuity, and invariance to affine transformations that make it very suitable and attractive for surface representation and 3D object modeling. Despite their numerous attractive

properties, however, B-Splines have not been widely applied for representing a 3D scattered non-ordered data set [35].

2.6. Neural Network Predicting and Modeling of 3D Objects

Several methods in the literature used the neural networks to model and predict 3D objects; among these methods we can mention deep networks, particle graph network, convolutional neural networks, feedforward neural networks, self-organizing maps and neural gas networks.

Wang et al. [135] proposed a deep network able to predict body deformations under external forces from a single RGB-D image. The network is trained on a collection of different objects generated by a physical finite element model simulator. To train the network, they used a single RGB-D depth image of the deformable object, the type of material, the size of the force and the location of the force. The network is able to output a predicted 3-D deformation of the object. They applied their network to the problem of safe and fast navigation of mobile robots carrying payloads over different obstacles and floor materials. Their architecture is composed of two main neural networks: a generator neural network G and a discriminator neural network D. The generator G used for training, while the discriminating neural network uses the data produced by the generator and determines whether the predicted outputs are realistic. The performance of their solution is demonstrated for soft materials such as foam.

Valencia et al. [136] discuss the use of Growing Neural Gas (GNG) and Particle Graph Networks (PGN) to capture the deformation of non-rigid objects from real world scenes. Growing Neural Gas (GNG), presented in [137] is a type of unsupervised learning algorithm that demonstrated interesting capabilities to represent complex changing structures in the context of non-rigid object manipulation. Particle Graph Network (PGN) model [138] uses particles to represent complex entities. PGN represents each entity as a graph neural network which has the ability to predict by learning parameters from data. The authors employ PNG in conjunction with GNG to predict the shape of non-rigid deformable objects from real world observations. While interesting, the proposed method requires a lot of processing.

Hu et al. [139] present a method to manipulate and predict the deformation of objects using deep neural networks. Based on 3D point cloud extracted from the 2D images, the authors compute a feature vector using a 135-dimension histogram. In order to learn the deformation function, they used a neural network with five layers, with a 30-16-8-6 architecture. The problem with their algorithm is the long computation, rendering it unusable for complete, complicated deformed objects.

Pumarola et al. [200] proposed a method for predicting the 3D shape of a deformable textured surface of object from a single view RGB image using deep learning methods. and demonstrate its use on a bending paper and a deforming t-shirt. Deep learning techniques require large amount of data to be trained, which is a limitation of such models. Moreover, the approach is designed based on textures, making it hard to use on low-textured surfaces.

Tsoli et al. [141] presented a deep learning method to predict and reconstruct a deformable 3D surface without texture from a single RGB image, under various lighting conditions. No explicit assumption about the shape of the object observed is made. They use a deep neural network to learn how to map image patches to their corresponding geometries. The input image is divided into overlapping patches and a depth map, as well as a normal map, are estimated for each patch using deep learning. They found that different areas of the deformable object have similar types of deformation, for example similar wrinkles appear in different areas of the surface of a cloth. To solve this problem, they proposed learning local models of shape variations from image patches which they combined in a global reconstruction of the object observed. This method cannot be applied on objects with texture and requires a large dataset to be able to learn, so the processing time is long.

Arriola-Rios and Wyatt [142] developed a multimodal learning framework to monitor and predict the deformations exerted by a robot in a single point contact scenario. They incorporated force and visual information into their development. They focused on learning to predict what happens when an elastic or plastic object is pushed by a robot finger. They studied the possibilities to predict the deformations of the object’s shape and the behavior of the object after contact by the robot terminates.

Wang et al. [143] proposed a deep network to predict the deformation on non-rigid 3D objects for tasks such as as robot manipulation, terrain deformation assessment, and in general for predicting in the context of end-to-end learning of physical models of the environment.

Yumer et al. [144] introduced an end-to-end solution to manipulate shape deformation using a volumetric Convolutional Neural Network (CNN) that learns deformation flows in 3D. The network architecture takes the voxelized representation of the shape as input and generates a flow deformation at the output.

Deng et al. [145] introduced a method based on the neural articulated shape approximation that enables efficient representation of articulated deformable objects using neural indicator functions. They propose a neural model of articulated objects to predict and model shapes by networks that encode a piecewise rigid decomposition. The articulated deformation is accomplished by using the linear blend skinning (LBS) algorithm [146] that deforms vertices of a mesh surface.

Li et al. [147] introduce a predictive, model-driven approach for robotic manipulation of 3D deformable objects using a precomputed, simulated database of deformable object models. To validate their approach, they developed a comprehensive pipeline for manipulating clothing, as in a typical laundry task.

Orts-Escolano et al. [148] proposed a method based on neural gas algorithm, an artificial neural network introduced by Thomas Martinetz and Klaus Schulten [149]. Neural gas is used over a raw point cloud to provide a 3D structure which has less information than the original 3D data but preserves the 3D topology.

Garcia-Rodriguez et al. [150] proposed a method based on the neural gas model to reconstruct objects from the point cloud obtained from overlapped multiple views using low cost sensors. To validate their proposal, compared their results with those obtained by a self-organizing map (SOM), another artificial neural network that uses unsupervised learning to reduce the input space of the training samples. SOM is essentially another method for dimensionality reduction. However, the main problem is that it suffers from boundary problems, and the modeled objects generally do not look good [151].

Frey [152] developed a method based on Dynamical Deformation Network (DDN) that is able to capture deformation over time in voxel data. They represent the deformed 3D object as voxels and a DDN, a deep generative model is demonstrated to be able to learn the deformation characteristics of 3D objects and capture the deformation.

2.7. Other Methods for Modeling and Prediction Deformable 3D Objects

Puig and Daniilidis [153] subdivide a deformable surface in smaller patches using visual and depth information. They use thin-plate splines with a minimal number of control points. Object deformations are computed based on tracking the node positions and solving the dynamic equations of Newton's second law. The main disadvantage of this technique is its complexity and computation time.

Aguiar et al. [154] built a system to capture the shape of 3D objects by combining surface- and volume-based shape deformation and a mesh-based analysis-through-synthesis framework. However, the high cost limits the practical application of such a system.

Caccamo et al. [155] proposed a system for modeling the deformation of elastic surfaces using an RGB-D camera, and a robotic arm equipped with a 3 fingered gripper that carries a 3D force sensor. They developed a framework to estimate the deformability parameters using the initial and final point cloud of the object extracted before and after physical interaction.

Elbrechter et al. [156] model a piece of paper in an interaction with a robot hand as a 2D grid of nodes connected by links that specify the bending constraints and a stiffness coefficient. However, the parameters of their model are tuned manually.

Park et al. [157] use a method that can monitor topological changes of the deformable surface using geometric measures containing notions of robust alignment, active region and minimal surface.

Conti et al. [158] have proposed an alternative method to model deformable objects that generates and connects together a surface and a volumetric representation based on the middle axis of a solid. Their algorithm separates local and global deformation and renders

them together with variable levels of resolution. This method is based however on a complex geometric representation.

Li et al. [159] integrate the extended finite element (XFEM) in their system in order to model the deformation of a dissected body. They develop a semi-progressive cutting method based on a local neighborhood search on the tetrahedral mesh.

Sederherg et al. [160] use the free-form deformation technique for deforming solid geometric models. Their method can deform surface primitives of any type or degree: planes, quadrics, parametric surface patches, or implicit surfaces and can be applied locally or globally.

The authors in [161] use extensive pre-computations of deformations. The problem in their approach, is that pre-computing deformations becomes quickly unfeasible when the number of possible user input configurations grows exponentially.

Oliviera et al. [163] propose an approach in virtual medical training, which combines methods and models to simulate elastic deformations and reach equilibrium between visual and haptic realism. They manipulate 3D objects representing human organs and tissues described by parameters such as shape, topology, color, volume, texture, and physical properties such as elasticity and stiffness. Authors of [164,165] present a method for simulating deformation of 3D objects that represent human organs through many tissue layers, in which the following physical parameters are considered: modulus of elasticity, girth and density.

Alambeigi et al. [166] simulate and control the deformation of 3D objects in medical surgery using a robot.

The authors of [167] track deformable objects from a sequence of point clouds by identifying the correspondence between points in the cloud and a model of the object composed of a collection of linked rigid particles, governed by dynamical equations. They apply their method for analyzing the motion of the left ventricle of the heart.

Park et al. [168] use a technique based on a class of physics-based deformable models whose parameters are functions. The latter allow to capture the local shape variation of

complex objects. In [169], common deformation techniques, such as axial deformation, are employed to track objects. Then, parametric features are grouped into systems of primitive constraints based on user specification. Finally, parametric features are reconstructed by minimizing the changes in the deformed model.

Yu et al. [170] use an RGB video as input to capture the deformations of generic shapes and the depth estimation. Bin et al. [171] use a multi-body system consisting of rigid bodies coupled by spring and damper elements to model objects undergoing large deformations.

Mero et al. [172] use a mixed model based on FEM and mesh free methods. They build a 3D deformable model which can be included in a general application of virtual reality in a medical surgery simulator. They construct a multiresolution model that can be used to obtain real-time response when a user interacts with the object.

Hui et al. [173] proposed a robotic system for modeling and classifying deformable objects as elastic, plastic or elasto-plastic, according to the material from which they are made, and to support the recognition of the category of these objects. They used a three-finger hand robot to manipulate the object and a Kinect sensor to acquire visual and depth data. Drimus et al. [174] also propose an object modeling and classification system which comprises a mixture of rigid and non-rigid objects. Jiménez [174] presented an overview of techniques for modeling and manipulating planar and linear deformable objects such as thin sheets and tissue.

A few comprehensive overview papers on the topic of modeling and manipulating deformable objects are published as well in the literature. Sanchez et al. [176] presented a review of recent approaches focusing on applications for handling deformable objects in domestic and industrial environments. In the same direction, Nadon et al. [177] provide a comprehensive overview of recent advances in modeling and manipulation of non-rigid objects. They described several methods and technologies used to measure the shape and estimate the material and physical properties of non-rigid deformable objects.

2.8. Summary and Discussion of the Work in the Literature

In this chapter, we described methods that cover the three main domains of research related to the work in this thesis, namely data acquisition and reconstruction of 3D deformable objects, modeling of 3D deformable objects and prediction of deformable objects.

Several methods were used to acquire 3D data from deformed objects. Among these, we described the acquisition of data from images, using laser scanners and using Kinect cameras. The problem with the acquisition of data from images is that some points can be visible only in some of the views, which causes difficulties when modeling the object. Laser scanners can scan with precision, but are in general slow for real-time applications. In the context of this work, we chose the Kinect camera because the data collection takes less time than using a scanner. Moreover, it is considered that this sensor provides sufficient precision for the kind of research work proposed in this paper.

This section also discussed various methods and solutions to reconstruct 3D objects from collected data. The main problem for object reconstruction is that when the object is complex, it requires a huge amount of storage space and analysis time to rebuild it. Several solutions have been proposed as well in the literature for modeling deformable objects. These include: mass-spring, finite-element (FEM), surfels, mesh-free, Green functions and Non-uniform rational basis spline NURBS. Each solution has its specific advantages, but at the same time its own issues: mass-spring models are easy to implement and effective in computation, but their accuracy in general is significantly lower. FEM models have more precision than the mass-spring models, but the main limitation of the FEM is that it is computationally expensive. In the surfels method, there is no need to compute topology information such as connection between vertices, but they require the scene surface to be sufficiently textured. Mesh-free methods do not use meshes, but they are often unstable when treatments are required to implement derived boundary conditions. B-Spline surface modeling possesses many properties such as boundedness, continuity, and invariance to affine transformations, but it not been widely applied for representing a 3D scattered non-ordered data set [35].

Several solutions have been also proposed in the literature for modeling and predicting deformable objects using neural networks. These include deep networks, particle graph networks, feedforward neural networks, self-organizing map and neural gas networks. While all the methods are interesting and have their specific advantages, most solutions make assumptions on the material of the deformable object. Deep learning solutions require large amounts of data for training and many solutions involve long processing time. Other solutions cannot predict a deformable object with a texture, while yet other solutions require higher precision sensors with real-time acquisition capacity.

The discussion of all these methods led us to a good understanding of the problems encountered in these three areas, highlighting the need to find new and complete solutions to address these problems. Reducing the complexity of deformed objects without losing the similarity to their original versions is an issue often encountered in the literature. Solving this issue helps us achieve two goals, the first is that we can obtain more precision in modeling and prediction method and the second is the reduction of computation time and memory storage. Most of modeling and predicting techniques assume that the parameters describing the object behavior are known in advance or values for these parameters are chosen by manually tuning them until the object shape and behavior seems plausible. This is not an easy process and cannot be adopted when a certain accuracy of the model is desired. Therefore, the interest in the development of novel methods that do not make assumptions on the material of the object, such as the one proposed in this thesis, is well justified. The aim of this work is to model and predict the deformation of objects without any assumption on its material and without explicit computation of elastic parameters. We describe the details of the proposed method in Chapter 3 and present experimental results in Chapter 4.

CHAPTER 3

Proposed Method for 3D Deformable Object Modeling and Prediction

In this chapter, we propose a novel method for the modeling and prediction of 3D object deformations. It is a data driven approach for constructing implicit, compact representations for deformable object shape and behavior (without assumption on the object shape or material). To achieve this, we capitalize on computational intelligence techniques, namely on unsupervised learning algorithms, such as neural gas networks and clustering-based stratified sampling for object representation/modeling, and on supervised learning algorithms, i.e. feedforward neural networks, for predicting 3D deformations.

Our method is based on a full framework for acquiring, modeling and predicting deformable object behavior (Figure 3.1). It begins with data acquisition. A 3D point cloud is collected by a Kinect sensor, while forces are exerted by means of the probing tip of a force-torque sensor. A neural gas fitting is proposed to describe the particularities of a deformation over the selectively simplified 3D surface of the object, without requiring prior knowledge about the object material. An alignment procedure, distance-based clustering, and inspiration from stratified sampling are supporting blocks of this process. A series of feedforward neural networks is then trained to learn the mapping between the force parameters characterizing the interaction with the object and the change in the object shape. To identify the most appropriate network to represent the deformable object shape and behavior, several neural network architectures were compared such as three feedforward networks (i.e. one for each 3D coordinate, x, y, z), as well as a single feedforward network with one hidden layer or with two hidden layers to preserve the coherence of 3D coordinates. Finally, it is demonstrated that learning can be transferred

from one neural network to another for objects with similar materials and shapes. For example, it allows to predict small objects from large objects and large objects from small objects of similar shape and material.

The inherent parallel structure and the potential for hardware implementation of neural networks provide an efficient way to ensure real-time prediction while still using real, raw noisy data, as opposed to the use of different approximations and assumptions on the object material, found nowadays in the literature. Therefore, the proposed method avoids estimating elasticity parameters which cannot be precisely and accurately identified for certain materials such as sponge or rubber, as those used in the context of this work. This means that the proposed method does not need to make assumptions on the material, such as its homogeneity or isotropy, assumptions that are often encountered in the literature. Due to this property, the proposed method can be used as is to characterize any object material deformation, including elastic, elasto-plastic, and even rigid objects; it is applicable as long as the object's material doesn't break or tear when forces with too large magnitudes are exerted on its surface.

Figure 3.1 summarizes the main steps of the proposed method for data acquisition, modeling, and prediction for 3D deformable objects and will be described in detail in the following sections.

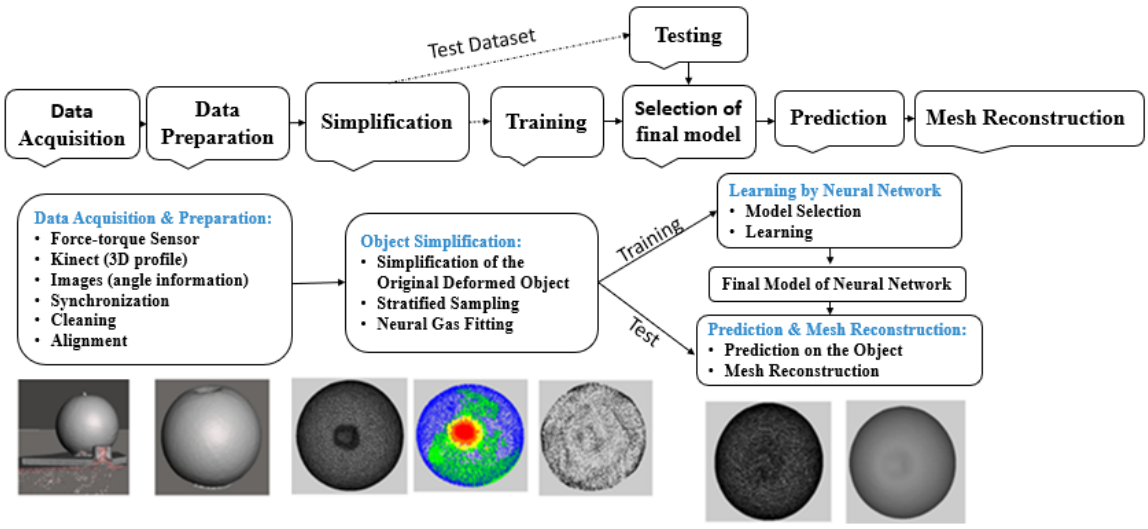


Figure 3.1. Proposed framework for deformation data acquisition and preparation, object simplification, modeling, and prediction.

3.1. Data Acquisition

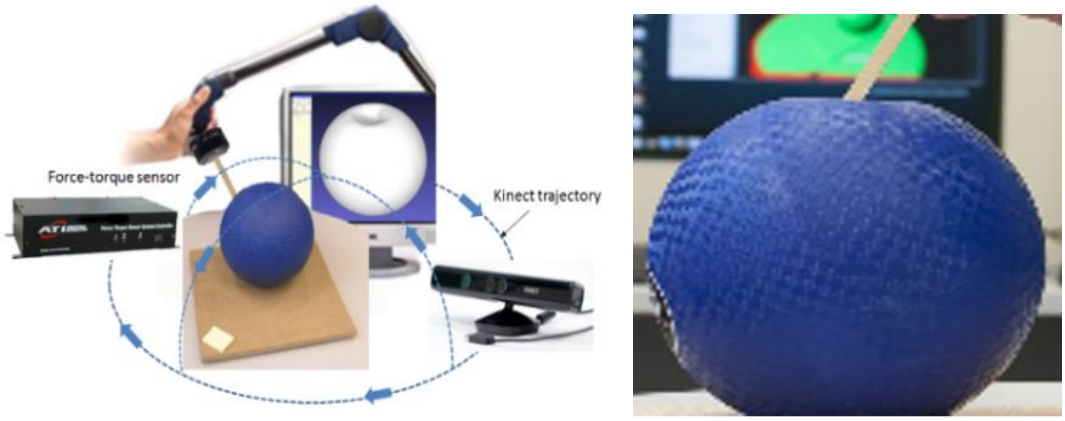
The experimental platform, illustrated in Figure 3.2 a, contains a Kinect sensor and an ATI force-torque sensor equipped with a probing tip. The Kinect is employed to collect 3D point clouds representing deformable objects by turning the sensor around the object of interest following the trajectory marked by blue arrows in Figure 3.2 a, and integrating the partial collected point clouds using a commercially available software Skanect [178]. Meanwhile, a constant force is applied on the object using the force-torque sensor.

Three-dimensional data are collected over the surface of the object with various force magnitudes at various angles of the force-torque sensor with respect to the normal plane on the surface. The interaction parameters with the force-torque sensor are recovered via a serial port. A Tera Term [179] script is used to initiate the data transfer and collect in real-time the measured force magnitude over the *x*, *y*, and *z* axes sent by the sensor. The force magnitude *F* at point *P* at the moment *t* can then be computed as the norm of the three orthogonal force components along the *x*-, *y*-, and *z*-axes of the force/torque sensor's reference frame returned by the sensor:

$$F_P = \sqrt{F_x^2 + F_y^2 + F_z^2} \tag{7}$$

It is important to mention that the raw data collected required to be corrected due to the fact that a probing tip was attached to the sensor and its weight had an influence on the measured force components. As well, it is also important to mention that only the force exerted on the object and captured by the sensor is considered, and not the reaction force of the objects as a result of the interaction with the object; the latter is reflected only in the change of shape that is visually captured by the Kinect. At the same time with the force data acquisition, images of the setup (an example is shown in Figure 3.2 b) are collected using a camera during each measurement step, each for 30 s, in order to recover with the help of image processing software the angle between the probing tip and the object surface. In particular, the probing tip is extracted from each image based on color information and morphological operations are applied to eliminate noise. The resulting probing tip is thinned (i.e., the image is simplified into a topologically equivalent image in which the skeleton of the probing tip is obtained by mathematical morphological

operations) to enable the computation of the angle with respect to the normal plane on the object surface. The angle is computed between the obtained skeleton representing the probing tip and the direction parallel to the top-bottom side of the image.



(a) **(b)**

Figure 3.2 (a) Experimental platform for collecting data on deformable object deformation behavior and (b) example of an image for angle data calculation.

3.2. Data Preparation

As previously mentioned, the data preparation procedure contains four steps, namely: data synchronization, data cleaning, data alignment, and data simplification. A synchronization process is required to associate the surface deformation with the corresponding angle and force magnitude measurements as they are collected at different sampling rates. The 3D models initially collected contain undesired elements present in the visual scene. These are eliminated in part manually and in part automatically using the random sample consensus (RANSAC) algorithm [180]. To ensure a simple interpolation between the various states of deformation and enable similar processing on the data collected over the surface of the objects, we first align the objects such that the deformed zone is situated on the top of the object. In a second step, we perform a fine alignment between each deformed object and the reference model using the iterative closest point (ICP) algorithm [181].

3.2.1. Data Synchronization

Because of the different sampling rates of the force-torque sensor, the angle information collection, and the 3D data acquisition, a synchronization process is required in order to associate the correct surface deformation with the corresponding angle and force magnitude measurements. This is achieved by calculating the mean of all measurements collected over the time it takes for the 3D model to be acquired (e.g., in the interval $t_1 - t_2$), namely: (1) the mean of the measured force magnitude values returned by the force-torque sensor for each component of the force along the *x*, *y*, *z* axes, respectively:

$$F_{aj} = \sum_{t=t_1}^{t_2} F_j/n, \tag{8}$$

where $j = \{x, y, z\}$ and n is the number of readings returned by the sensors in the interval $t_1 - t_2$ and (2) the mean of the force angle computed over the series of images:

$$a_{Pa} = \sum_{t=t_1}^{t_2} a_P/m, \tag{9}$$

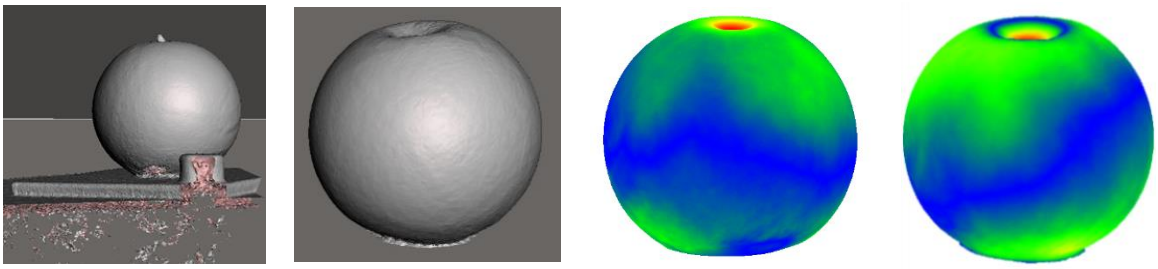
where m is the number of images captured (similar to the ones in Figure 3.2 b) in the interval $t_1 - t_2$. The deformed object model is therefore considered to be the result of the application of a force with a magnitude equal to the computed average magnitude along the *x*, *y*, *z* axes respectively: F_{ax}, F_{ay}, and F_{az} (Equation (8)), and $F_{Pa} = \sqrt{F_{ax}^2 + F_{ay}^2 + F_{az}^2}$, which is applied at an angle equal to the calculated average angle value, a_{Pa} (Equation (9)).

3.2.2. Data Cleaning

The 3D data collected using the Kinect sensor contains undesired elements, such as the table on which the object is placed, the fixed landmarks required by the software to merge data from multiple viewpoints, and the probing tip, as can be noticed in Figure 3.3 a. These are eliminated in part automatically (e.g., table and landmarks) and in part manually (e.g., the probing tip that is acquired as part of the object). Since the object is placed on a table during experimentation (Figure 3.2), an efficient way to remove most of the background is to locate the planar surface representing the table and extract it from

the RGB-D data. The flat surface identification is viewed as a plane-fitting problem, which is resolved with the random sample consensus (RANSAC) algorithm [182].

Once the best plane model is obtained by the algorithm, the flat surface and the background information, represented by the inliers of this model, are removed from the point cloud. To automatically remove the landmark, only the largest component remaining in the scene, which is the object, is considered for further processing. This is a reasonable assumption, because the landmark used during experimentation is relatively small with respect to the size of the object (shown in the bottom right corner of the table, in Figure 3.3 a). The holes in the object around the probing tip resulting from its removal are filled using Meshmixer [183], a mesh processing software, in order to obtain clean representations of the deformable object. Figure 3.3 b shows an instance of cleaned 3D data, while Figure 3.3 c and 3.3 d encode in color the regions that are displaced during an interaction with a soft and a strong force, respectively.



(**a**) (**b**) (**c**) (**d**)

Figure 3.3. (a) Raw data collected; (b) preprocessed data; and (c, d) deformation distributions with respect to the non-deformed object when applying a light and a strong force, respectively; regions in blue are not deformed, and the deformation is getting stronger from green to red.

By comparing the initial non-deformed object with a deformed object using CloudCompare [184], one can notice in blue the regions that are not deformed (least error), while for the areas going from green to yellow, orange, and red, the deformation is getting stronger (larger errors). Figures 3.3 c and 3.3 d show that, in fact, not only the area around the probing tip is affected by the interaction, but a much larger area of the

object. Such behavior is extremely difficult, if not impossible, to capture by existing 3D modeling techniques, such as mass-spring system or FEM.

3.2.3. Data Simplification

Large and complex object models make the interaction with the object more difficult and require more calculations and more memory for storage. The goal is thus to create simplified objects of the original object with fewer elements (i.e. number of faces) while maintaining the main characteristics of the original model.

In order to better characterize the deformation around the probing point, instead of using the entire collected pointcloud **M**, we first construct a selectively-densified mesh, $\mathbf{M}_s$ (of which an example is shown Figure 3.5 a), in which the area around the point of interaction between the probing tip and the object surface is preserved at higher resolution, while the other areas are simplified. This ensures that the small deformed area around the probing tip has a higher density of points with respect to the rest of the object's surface. To achieve this goal, similar to the approach proposed in [90], the QSlim [185] algorithm was adapted to only simplify points that are not the interaction point with the probing tip and its *n*-degree immediate neighbors.

Algorithm 1 summarizes the proposed object simplification process. For experimentation, we chose 12-degree neighbors for the ball and 8-degree neighbors for the cube and the sponge, respectively. These were selected in such a way that the entire deformation area is covered in all deformed instances of the object within this neighborhood. Details on the selection of 12 and 8 neighbors, respectively, are provided in the Chapter 4 (section 4.3) that presents the experimental results of the thesis. This process allows us to define an equal number of faces for all the instances of a deformed object, as the number of faces is a parameter to be provided in the simplification process. It therefore ensures a uniform representation of all deformed instances of an object.

Algorithm 1. Object simplification using QSlim adapted.

Input: $\mathbf{M}$ = cleaned acquired mesh

Output: $\mathbf{M_s}$ = simplified mesh (denser around probing point)

Step 1:

Identify the point of intersection $\mathbf{I}$ of the probing tip with the object mesh*:* $\mathbf{I} = [X, Y, Z]$, $\mathbf{I} \in \mathbf{M}$

Step 2:

// simplification of mesh $\mathbf{M}$ to obtain selectively-densified mesh $\mathbf{M_s}$

Calculate **INN** = the *n*-nearest neighborhood of point $\mathbf{I}$ in $\mathbf{M}$

// identify only edges that do not belong to **INN**

Initialize $\mathbf{M_s} = \{e(u,v) \mid e(u,v) \in \mathbf{M} - \mathbf{INN}\}$

// apply QSlim

Compute quadric error at each vertex of $\mathbf{M_s}$

Determine contraction cost at each edge $e(u,v)$ in $\mathbf{M_s}$

Create ordered list of edges based on cost

Step 3:

Remove the edge $e(u,v)$ with lowest cost

Use quadric error to choose the optimal contraction target

Contract u and v and recalculate costs for the adjacent edges in $\mathbf{M_s}$

While desired number of faces not reached, repeat Step 3

3.2.3.1. QSlim for Surface Simplification

QSlim is a surface simplification algorithm which can rapidly produce high quality approximations of polygonal models. The algorithm uses iterative contractions of vertex pairs to simplify models. It is assumed that the model consists of triangles only [185]. This implies no loss of generality, since every polygon in the original model can be triangulated as part of a preprocessing phase. When two faces are intersecting at the same point, this point is shared between the two faces rather than using two distinct points (i.e. one point for each face). A geometric error approximation is maintained by this algorithm at each vertex of the object model. This error is represented using quadric matrices.

In the classical QSlim [185], the algorithm contracts the edge between two vertices at a single point when an edge connects them, so one or more faces can be removed.

Unconnected sections of the model are joined by moving the nearest points in each section to the same position. Let $\mathbf{v_1}$ and $\mathbf{v_2}$ two vertices non-connected, let $\mathbf{v_1}$ belongs to the section 1 and $\mathbf{v_2}$ belongs to the section 2, we move $\mathbf{v_1}$ and $\mathbf{v_2}$ to a new position **v** and we connect the two sections on **v**. A sequence of pair contractions is applied until the desired number of faces is reached.

In our adapted algorithm (Algorithm 1), we first identify **I**, the point of intersection of the probing tip with the mesh of the object and compute **INN**, the set of *n*-neighbors closest to this point. We then compute **ns,** the set of vertices do not belong to **INN**, as we aim to simplify all the other areas of the object. Based on the identified edges belonging to **ns**, we calculate the quadric error at each vertex, determine the cost of contraction at each edge, then create an ordered list of edges based on cost. Finally, we remove the edge with the lowest cost, and repeat this process until we reach the desired number of faces. Finally, after all the iterations, we recompute all edges and re-establish connections between vertices.

Among the advantages of our adapted version of the QSlim algorithm, we mention the following:

1) It allows us to reduce the computation time and simplify just the non-deformed part, close the holes and re-establish the connection between the deformed and the simplified non-deformed parts of the model. Several other algorithms [38, 39, 40] can simplify the entire model by iteratively contracting edges. These methods cannot join unconnected regions.

2) The simplified models obtained by our adapted algorithm preserve better the primary features of the original model, because we do not simplify in the model the deformed area around the probing tip. The algorithm can thus simplify complex models even more rapidly and brings thus additional savings in computation time.

3.2.4. Data Alignment

In order to ensure a simpler interpretation of the deformation (e.g., comparison for various angles and forces) and a similar treatment of the deformation over the surface of an object, the objects are realigned such that the deformed zone is situated on the top of

the object. This operation is especially useful for symmetrical objects, such as the ball, but improves the interpretation of the deformed zone for all types of objects. The alignment procedure is executed in two stages: In the first stage, the object is rotated such that the contact point with the force-torque tip is aligned along the *y* axis. The *z* axis points in this case towards the user, *y* is the up-down direction and *x* is the left-right direction, as shown from two viewpoints in Figures 3.4 a and b, while Figure 3.4 c shows an example of the aligned model, $\mathbf{M}_{\mathbf{a\_axis}}$, when compared to a reference model, $\mathbf{M}_{\mathbf{r}}$. The latter is selected to be the one in which the strongest force (with respect to the capabilities of the sensor used) is applied in the normal direction on the top of the object, but other deformed models could be used as well. The reason to choose a reference model that is already deformed instead of its non-deformed initial state is to ensure a better alignment of the two models around the deformation zone.

In a second stage, a fine alignment is performed between the reference model, $\mathbf{M}_{\mathbf{r}}$ and each simplified and aligned model, $\mathbf{M}_{\mathbf{a\_axis}}$, using the iterative closest point (ICP) algorithm [181]. In order to ensure good results, the latter requires the two models to be roughly aligned, from which stems the necessity of the initial axis alignment in the first stage. An example of color-encoded differences obtained using CloudCompare [184] between the non-deformed object model and a model in which a light force (4 N) is applied on the top at an angle of 75 degrees with respect to the *y* axis is shown after the axis alignment, but prior to ICP alignment in Figure 3.4 c and after ICP alignment in Figure 3.4 d. One can notice that after the ICP alignment, the differences focus mainly on the deformed zone and not over the entire surface of the ball (e.g., there are more blue areas around the ball surface, showing a better fitting). The combined transformation (translation and rotation) matrix returned by ICP is saved in order to allow the repositioning of the deformed zone at its correct place over the surface of the object.

This step is summarized in Algorithm 2.

Algorithm 2. Object alignment.

Input: $\mathbf{M_s}$ = selectively densified mesh

$\mathbf{M_r}$ = reference mesh

Output: $\mathbf{M_a}$ = aligned mesh with reference object

Step 1:

// Rough axis alignment

Align $\mathbf{M_s}$ along *y* axis to obtain $\mathbf{M_{a\_axis}}$

Step 2:

// ICP refinement

For each point $\mathbf{P} \in \mathbf{M_{a\_axis}}$ find the closest point $\mathbf{P_r} \in \mathbf{M_r}$

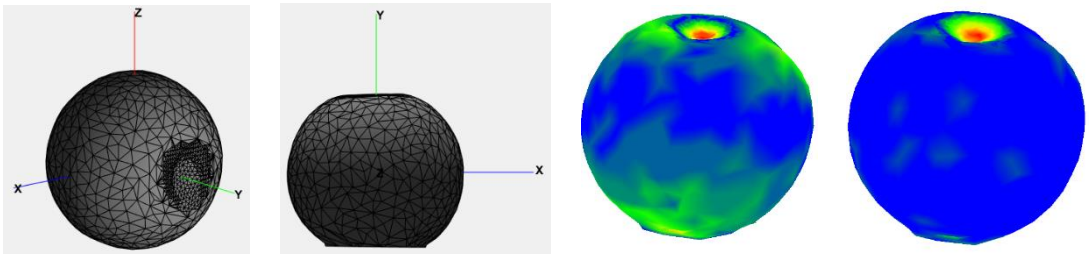
Calculate mean-squared error between **P** and $\mathbf{P_r}$

Estimate the combination of translation and rotation to ensure best alignment between **P** and $\mathbf{P_r}$ based on the computed error

Transform the source points using the transformations to obtain $\mathbf{M_a}$

Store the transformation matrix.

While error larger than threshold repeat Step 2



(**a**) (**b**) (**c**) (**d**)

Figure 3.4. (a) Aligned model with *x*, *y*, *z* axes, (b) *x*, *y* view; and difference between a model when a light force is applied on the top at an angle of 75° with respect to the y axis and the reference model (c) before using the ICP (iterative closest point) and (d) after using the ICP (iterative closest point) alignment.

3.3. Object Deformation Characterization

The object deformation characterization procedure comprises three steps, namely: data sampling, neural gas network fitting, and neural network prediction of neural gas nodes.

Once the object is aligned using the ICP algorithm, it is clustered and sampled to reduce the complexity of each instance of deformed objects and to better characterize the deformation area. After identifying the number of clusters, we use stratified sampling to randomly select points with varying proportions from each cluster.

A neural gas network is fitted over the sampled points and takes the form of the object while preserving more details in regions with local geometry changes [130]. This property allows to ensure that fine differences around the deformed zone and over the surface of the object can be captured accurately in the model.

After obtaining the neural gas fitting for each instance of a deformable object, a feedforward neural-network architecture learns a mapping between the measured parameters characterizing the interaction with the object (force magnitude, angle, etc.) and the change in the object shape (captured implicitly by neural gas nodes) due to the interaction. To identify the most appropriate network to represent deformable object shape and behavior, several neural network architectures were compared such as three feedforward networks (i.e. one for each 3D coordinate), as well as a single feedforward network with one hidden layer or two hidden layers to preserve the shape coherence of the object by considering the existing link between 3D coordinates (instead of considering them separately).

3.3.1. Data Sampling

To enable the characterization of the object deformation, it is necessary to reduce its complexity.

In this thesis, this issue is addressed in part by using a lower number of points from the selectively-densified mesh, using inspiration from stratified sampling and in part by the neural gas fitting procedure described in the next section.

Stratified sampling is a technique that generates samples by subdividing the sampling domain into non-overlapping partitions (clusters) and by sampling independently from each partition. In particular, the ICP aligned mesh, $\mathbf{M_a}$, is clustered according to the distance to the initial non-deformed mesh. We computed the Hausdorff distance between each instance of the deformed object and the non-deformed object to detect the number of clusters. After obtaining the interval of distances between each instance of the deformed object and the non-deformed object, we normalize them in order to ensure a uniform processing over the entire surface of the object. The normalized interval between the deformed mesh and each instance of the object under study is gradually split in an increasing number of equal intervals (number of clusters to be used) and the points in the deformed area (probing tip and its immediate neighbors) are compared with the cluster situated at the largest distance. This cluster normally corresponds to the deformed area. It is desired that the largest possible number of points in the deformed area are situated in this cluster. Figure 3.5 shows our method for choosing the number of clusters. In step 1, we split the interval of distances into two clusters, then we compared the deformed area around the probing tip with the cluster C2 (cluster with largest distance). We stop the process when we find the largest possible number of points in the deformed area situated in this cluster. Subsequently, in step 2, we split the interval of distances into three clusters and we repeat the same process done in step 1. We stop the process when we find the largest possible number of points of the deformed area situated in the cluster with largest distance. Then, at each step i (i between 4 and N-1), we split the interval of distances into $i + 1$ clusters and we repeat the same process done in step 1. At the end of the process, we retain the step that has the maximum number of clusters and which has the largest possible number of points in the deformed area situated in the cluster with largest distance. Once the number of clusters is identified, points are sampled randomly but in various proportions from each cluster to identify the adequate amount of data to be used. The proportions were varied during experimentation, by taking into consideration the fact that a good representation is desired specifically in the deformed area and therefore more samples are desired for regions in which the deformation is larger. In the context of this work, we gradually decreased the percentage of points sampled from the farthest cluster to the closest cluster representing the deformed zone. We chose the percentage of sampling

points from each cluster by monitoring the evolution of errors between the sampled model and the original model (results are presented in Chapter 4, section 4.1.3).

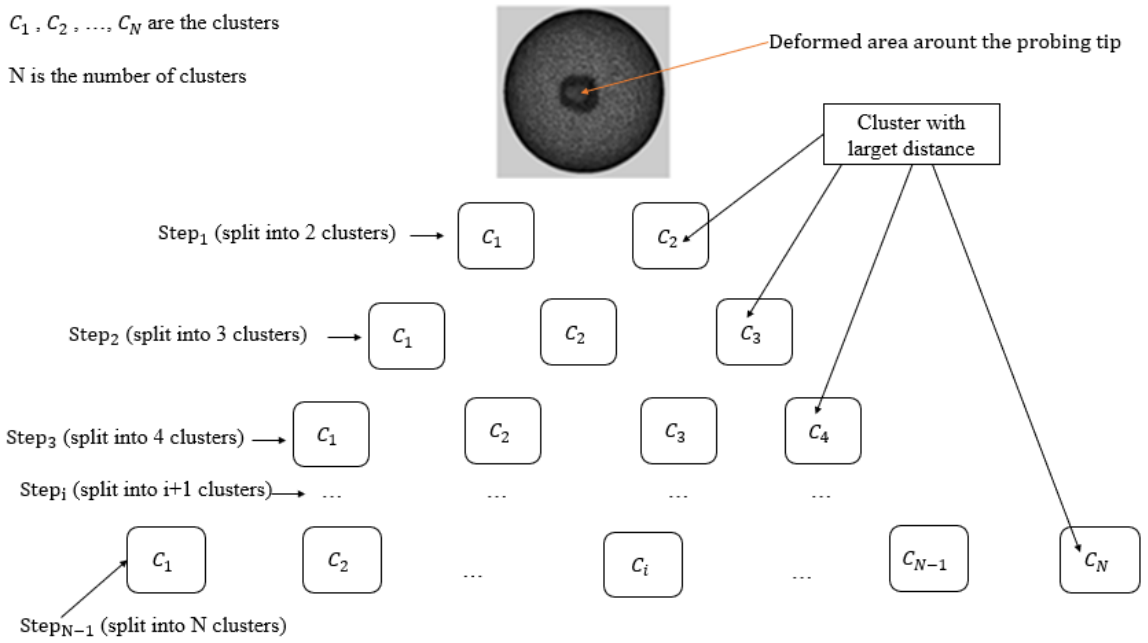
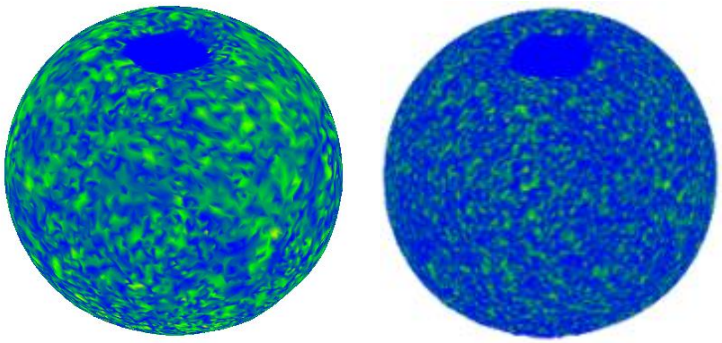


Figure 3.5. Clustering method based on the distance between the mesh of each deformed instance and the initial non-deformed mesh.

3.3.2. Neural Gas Networks

Neural gas networks are subsequently used in an attempt to improve the quality of the object model. After the application of stratified sampling, a comparison was performed between the reconstructed object and the non-simplified reference object using CloudCompare. Figure 3.6 a shows the color-coded comparison between the initial full resolution ball and the reconstructed ball after stratified sampling. Figure 3.6 b shows the color-coded comparison between the initial ball at full resolution and the ball reconstructed after neural gas execution. One can notice that a very good match is associated in both cases with the deformed zone (represented in blue), but that the errors are lower after neural gas tuning (Figure 3.6 b) due to a better distribution of faces over the surface of the object.



(**a**) (**b**)

Figure 3.6. Color-coded results for the ball with respect to the initial full-resolution model: (a) selectively-densified mesh around the probing point for a ball; and (b) final mesh for a ball after using neural gas.

We have therefore chosen to use the neural gas network to capture these fine differences; such differences are very difficult to capture using other classical 3D modeling techniques such as mass-spring models and FEM, for example. The choice of this network is also justified by the fact that it converges quickly, reaches a lower distortion error, and better captures finer details, as opposed to other architectures that obtain larger distortions and tend to smooth fine details, such as self-organizing architectures [45, 46] that we presented in the literature review. The neural gas network is a suitable model because of its flexibility and excellent quality of representation of different real-world objects [150]. The proposed algorithm (Algorithm 3) takes unorganized three-dimensional data points as input and generates a 3D (topology-preserving) pointcloud that better represents the 3D object.

To understand the functionality on the neural gas method, let $\mathbf{p}(\mathbf{X})$ denote the probability distribution of the sampled points obtained by the stratified sampling method, and let $\mathbf{w_i}$ $(i = 1, \dots, N)$ denote a finite number of feature vectors which will cover and represent all the sampled points. The number of neurons represented by the feature vectors $\mathbf{w_i}$ is less than or equal to the number of sampled points. In the context of this work, the number of neurons is considered equal to the number of sample points in order to preserve the degree of accuracy of the model. During learning, in each time a data point $\mathbf{x}$ is randomly chosen from each $\mathbf{p}(\mathbf{X})$. Then the distance of the feature vectors to the given

data vector $\mathbf{x}$ is determined in order to create an ordered list of feature vectors wih respect to this distance. Let $\mathbf{i_0}$ denote the index of the closest feature vector distant to $\mathbf{x}$, $\mathbf{i_1}$ the index of the second closest feature vector distant to $\mathbf{x}$, and $\mathbf{i_{N-1}}$ the index of the feature vector most distant to $\mathbf{x}$. And let $\mathbf{k}$ denote a variable representing the ranking of each feature vector with respect to the input $\mathbf{x}$. $\mathbf{k = 0}$ for the first closet feature vector to the input $\mathbf{x}$, $\mathbf{k = 1}$ for the second feature vector, $\mathbf{k = 2}$ for the third closet feature vector and so on. The algorithm first determines the ranking of the feature vectors $\mathbf{w_{i_0}}$, $\mathbf{w_{i_1}}, \dots,$ $\mathbf{w_{i_{N-1}}}$ $(\mathbf{k = 0}, \dots, \mathbf{N - 1})$, $\mathbf{N}$ is the number of feature vectors. After ranking, each feature vector is adjusted by using the formula [189]:

$$\mathbf{w_{i_k}^{t+1}} = \mathbf{w_{i_k}^{t}} + \boldsymbol{\alpha}(\boldsymbol{t})\, \boldsymbol{e}^{-\boldsymbol{k}/\Lambda(\boldsymbol{t})}\left(\mathbf{x} - \mathbf{w_{i_k}^{t}}\right) \qquad (10)$$

where $\boldsymbol{t}$ is the time step, α is the learning rate and Λ is the characteristic decay constant, and $\alpha(t)$ and $\Lambda(t)$ are slowly decreased with increasing $\boldsymbol{t}$ to ensure the convergence of the learning algorithm. $\boldsymbol{\alpha}(\mathbf{t})$ and $\Lambda(\mathbf{t})$ are formulated, as in [46], as:

$$\boldsymbol{\alpha}(\mathbf{t}) = \boldsymbol{\alpha_o}(\boldsymbol{\alpha_T}/\boldsymbol{\alpha_o})^{\mathbf{t/T}}, \qquad \Lambda(\mathbf{t}) = \boldsymbol{\lambda_0}(\boldsymbol{\lambda_T}/\boldsymbol{\lambda_0})^{\mathbf{t/T}}$$

$\mathbf{T}$ represents the training length. The initial and final values for $\alpha(t)$ and $\Lambda(t)$ must be chosen at the beginning of the algorithm [46, 189]. α_o and λ_o are the initial values for $\alpha(t)$ and $\Lambda(t)$. α_T and λ_T are the final values for $\alpha(t)$ and $\Lambda(t)$. During experimentation, we tried different values for α_o, α_T, λ_o, λ_T, and we noticed that the best results obtained are by using the same values suggested in [149, 189], namely $\alpha_o = 0.5, \alpha_T = 0.05$, $\lambda_o = N/2$ (N is the number of sampled points) and $\lambda_T = 0.001$. The sampled data resulting from the process described in the last section (section 3.3.1), is fitted with a neural gas network. The input point cloud for the neural gas contains in this case the X, Y, and Z coordinates of the points sampled from the clusters. While the fact that the neural gas map size has to be chosen a priori is in general a drawback, it is an advantage in the context of the current thesis, as it maintains a constant size for the representation of a deformed object regardless of the initial size of the object collected by the Kinect. A complete description of neural gas is available in [149], and of its adapted version for 3D pointclouds in [46]. The neural gas fitting process used in this thesis is summarized in Algorithm 3.

Algorithm 3. Neural gas fitting.

Input: $\mathbf{P_{si}}$ = stratified sampled points from clusters C_1 to C_5

Output: **NG** = fitted neural gas network

Initialize neural gas network **NG** and set parameters:

$O = \{P_1, \ldots P_i, \ldots, P_{NS}\}$, $P_i = [X_i, Y_i, Z_i] \in \mathbf{P_{si}}$

$\alpha(t) = \alpha_o(\alpha_T/\alpha_o)^{t/T}$; $\lambda(t) = \lambda_o(\lambda_T/\lambda_o)^{t/T}$;

Initialize the weights w_i with small random values;

// apply neural gas adaptation over **O** to obtain the neural gas map: $\mathbf{NG_i} = \{P_1, \ldots, P_j, \ldots, P_N\} | P_j = [X_j, Y_j, Z_j], j = 1, \ldots, N$;

while $t < T$

Select a point $\mathbf{P_o}$ at random from **O**

for j = 1:1:N

Sort weights according to the distance to $\mathbf{P_o}$

Update weights $w_j(t+1) = w_j(t) + \alpha(t) h_\lambda\big(k_j(x, w_j)\big)\big[x(t) - w_j(t)\big]$;

where $h_\lambda\left(k_j(x, w_j)\right) = exp\big(-k_j(x, w_j)/\lambda(t)\big)$

Reduce learning rate α

end

end

3.3.3. Feedforward Neural Networks

Once a neural gas representation is built for each instance of a deformable object, a network architecture is trained to map the complex relationship between the force and angle data and the corresponding position of neural gas nodes over each cluster, and thus to capture implicitly the deformation behavior of an object. Feedforward neural network architectures are used for this purpose. Such fully-connected networks consist of a number of neurons, organized in layers, with each neuron in a layer being connected with all the neurons in the next layer [190]. Each connection may have a different weight. The weights on the connections encode the knowledge of a network. Data is entered at the inputs and passes through the network, layer by layer, until it arrives at the outputs. With

this kind of network, there is no feedback between layers. Therefore, they are called feedforward neural networks. To identify the appropriate architecture to use, several types of feedforward neural networks were compared. We started by using a series of two-layer feedforward networks (i.e. a network with one hidden layer), three per each cluster of an object, corresponding to the *X*, *Y*, and *Z* coordinates, respectively. After, we have tested three feedforward networks per each object. In an attempt to preserve the link between the 3D coordinates of the object shape, we also tested a single feedforward network with a single hidden layer or with two hidden layers. The following sections describe these various configurations.

3.3.4. Prediction of Neural Gas Nodes Position Using Three Neural Network for Each Cluster

In a first configuration, a series of two-layer feedforward networks, three per each cluster of an object (section 3.3.1), denoted *NNX*, *NNY*, and *NNZ* in Figure 3.7, and corresponding to the *X*, *Y*, and *Z* coordinates, respectively, is trained to map the complex relationship between the force and angle data and the corresponding position of neural gas nodes over each cluster.

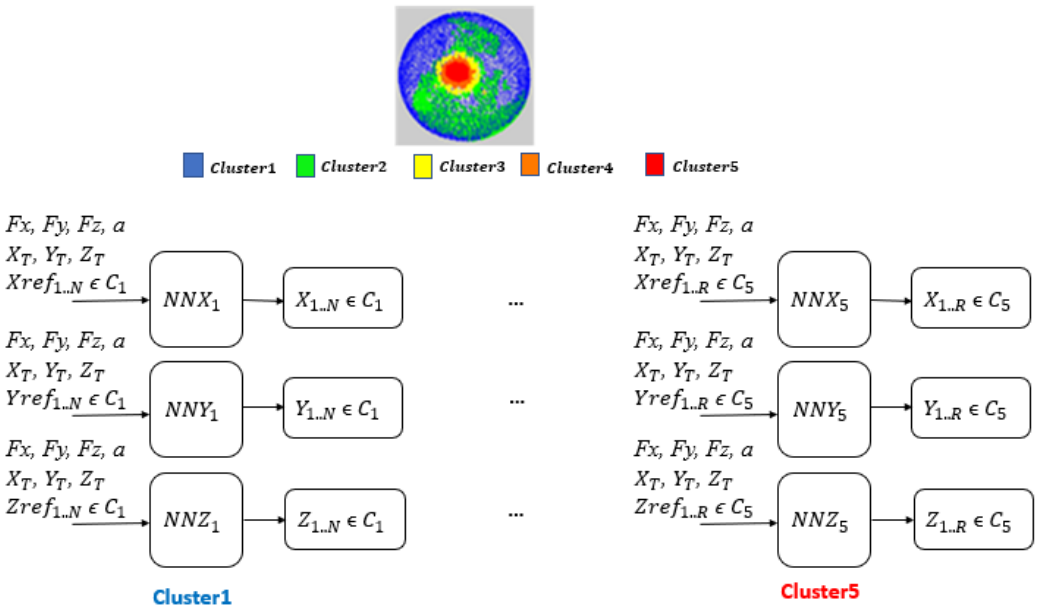


Figure 3.7. Feedforward architecture for neural gas node prediction using three neural networks per each cluster.

Each network takes as inputs the force components F_x, F_y, and F_z, the angle a, the coordinates X_T, Y_T, and Z_T of the point of intersection **I** of the object with the probing tip, and a series of X, Y, and Z coordinates. In an attempt to simplify the complex learning scenario, the network learns to map between the neural gas nodes representing the reference model $\mathbf{M_r}$ of an object and any deformed stage of the object. Therefore, the X, Y, and Z coordinates in the input will correspond to neural gas nodes representing the reference model $\mathbf{M_r}$ and are denoted X_{ref}, Y_{ref}, Z_{ref}, respectively, in Figure 3.6. Each network contains 20 sigmoid hidden neurons, linear output neurons and uses the scaled conjugate gradient backpropagation for training.

3.3.5. Prediction Using Three Neural Networks for Each Object

In the second configuration, three neural networks per each object, denoted *NNX*, *NNY*, and *NNZ* (Figure 3.8) are trained to map the complex relationship between the force data and the corresponding position of neural gas nodes over each object. Instead of using three neural networks per each cluster (as in the previous section), we train only three neural networks on the neural gas nodes that represent the object in an attempt to reduce the number of operations and computing time and to identify the type of network that can provide a better prediction performance. Each network takes as inputs the force components F_x, F_y, and F_z, the coordinates X_T, Y_T, and Z_T of the point of intersection **I** of the object with the probing tip, and a series of X, Y, and Z coordinates (denoted X_{ref}, Y_{ref}, Z_{ref}, respectively) that correspond to neural gas nodes representing the reference model $\mathbf{M_r}$. The first network takes the X as inputs, the second network takes the Y as inputs and the third network takes the Z as inputs.

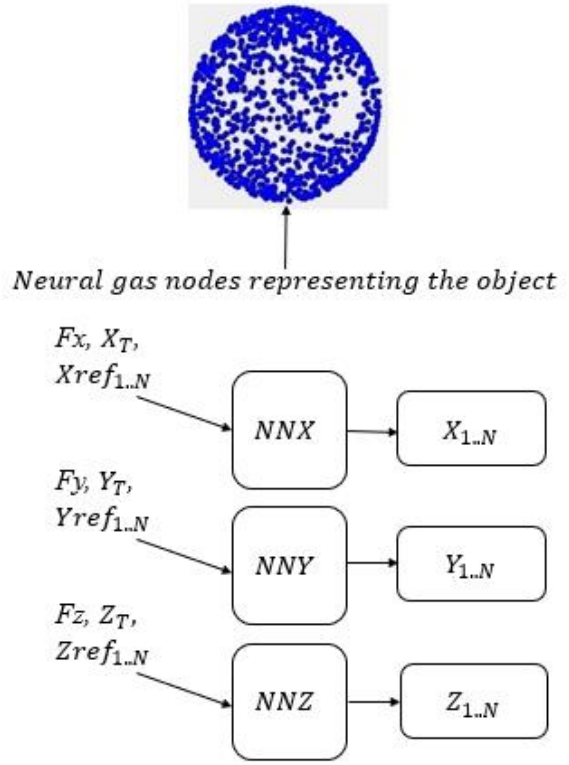


Figure 3.8. Feedforward architecture for neural gas node prediction using three neural networks per each object.

3.3.6. Prediction Using a Single Neural Network with One Hidden Layer for Each Object

In a further attempt to improve the performance (i.e. reduce the number of operations and the computing time) while also considering the link between the 3D coordinates of an object (i.e. its shape), we performed additional tests by using one single neural network with one hidden layer and after with two hidden layers (next section).

Initially, a single neural network with one hidden layer, denoted *NN* (Figure 3.9) is trained, as in the previous two configurations, to capture implicitly the complex relationship between the force data and the corresponding position of neural gas nodes over each object. This network takes as inputs the force components F_x, F_y, and F_z, the coordinates of the point of intersection **I (**X_T, Y_T, and Z_T**)** of the object with the probing tip, and the same coordinates used in the previous section, representing the reference model (X_{ref}, Y_{ref}, Z_{ref},).

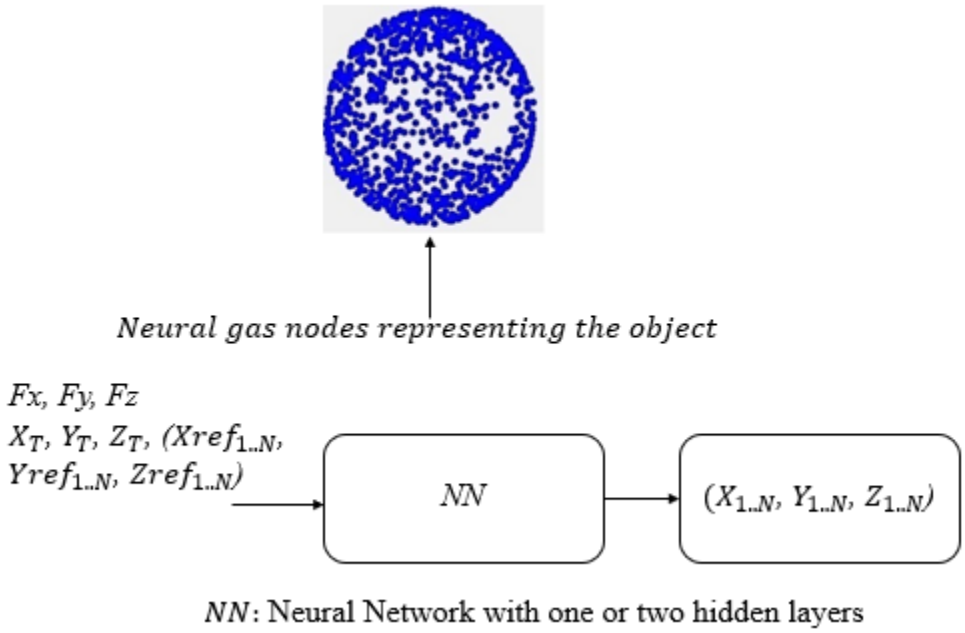


Figure 3.9. Feedforward architecture for neural gas node prediction using one neural network with one or two hidden layers.

3.3.7. Prediction Using a Single Neural Network with Two Hidden Layers for Each Object

In a last attempt, we trained a single neural network with two hidden layers for each object (Figure 3.9). This neural network takes the same inputs as the network used in the previous section (section 3.3.6). As it will be demonstrated in the experimental results in Chapter 4, we have identified this configuration as being the best in terms of performance.

3.3.8. Transfer of Learning Using Neural Networks with Two Hidden Layers

As described in Chapter 1, this thesis also aims to develop a transfer learning capability to account for the within class variability (shape and material) of objects. This consists mainly in enabling the method to account for different sizes of the same object as well as slightly varying physical parameters of the objects by using the learning of a pre-trained network on one object to accelerate the training of another object given that the physical properties of the objects are similar except for their size and/or slight geometrical differences. As training a model from scratch represents a lot of work and requires a lot

of data, the transfer of learning allows to use less data and thus to reduce the training time. For this purpose, we used two neural networks. The first neural network is used to capture implicitly the object model and the second one is fine-tuned and used for prediction (Figure 3.10). For example, we train the first neural network on data of small deformable ball. We transfer its learning to the second neural network, then we fine-tuned the second neural network on data representing a large ball and we use it to predict deformations for the large ball.

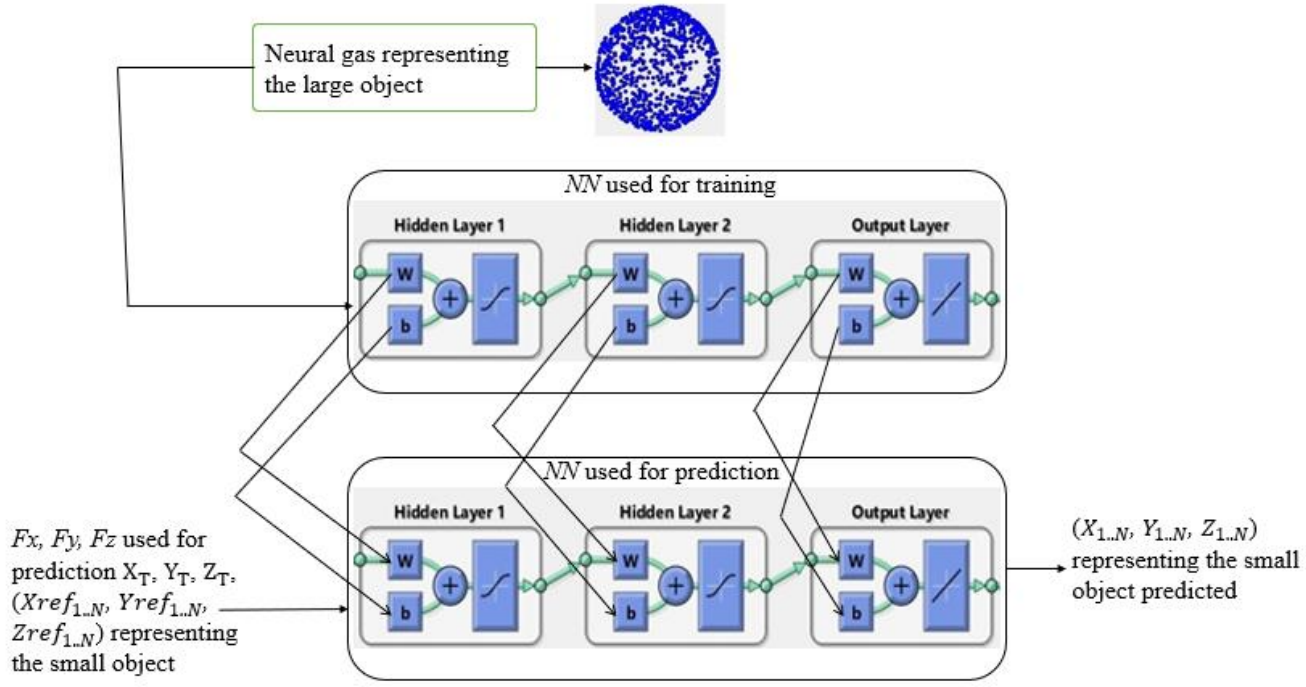


Figure 3.10. Transfer of learning between two neural networks for predicting a small object using a network trained on the large object.

In order to transfer the learning between the two neural networks, it is necessary that they have the same number of layers and the same number of neurons on each layer. After training the first neural network, we transfer its learning to the second neural network with the same architecture by transferring the weights matrices between the input and the first hidden layer, between the first hidden layer and the second hidden layer and between the second hidden layer and the output layer. Then, we fine-tune and use the second neural network for prediction. All the related details are presented in Chapter 4, section 4.6.

3.3.9. Deformed Object Reconstruction

Once the series of feedforward neural networks is trained, they can be used conjunctly to predict the position of neural gas nodes for force magnitudes and angles that were not measured, and therefore for data with which the network was not trained. The *X*, *Y*, *Z* coordinates of points predicted for each cluster by the series of the neural networks are assembled together to provide an estimate of the entire object shape. In order to reconstruct the object, one solution would be to construct a mesh directly over the predicted points, using for example a Poisson surface reconstruction [191]. However, this is not an optimal solution in this case, due to the fact that, during the simplification process we concentrated the effort in the deformed area and sampled much less points for the rest of the object. If the neural gas nodes are used as predicted, the quality of the model will be significantly deteriorated over most of the surface of the resulting mesh.

A solution to deal with this problem is to use the adapted data simplification algorithm that we presented in section 3.2.3, but to constrain the number of triangles in the mesh to be equal to the average number of faces in the reference mesh model, $\mathbf{M}_s$, representing the various deformation instances of an object. In this case, instead of performing a simplification operation, the adapted algorithm will force the redistribution of faces in those areas in which changes occurred as a result of the deformation. This is the solution chosen in this thesis.

3.4. Model Validation

To evaluate the results of the object representation and prediction using the proposed approach, as well as allow the selection of parameters, the similarity of the reconstructed model obtained in section 3.3.9 is compared to the cleaned object model (collected by the Kinect) qualitatively (visually) and quantitatively (error calculation). The difference between the acquired data belonging to a real object and its simplified version is visualized using CloudCompare [184]. Using the obtained, simplified representation as one model and the original full-resolution mesh of the object as another model, the difference between them will highlight the areas most affected by errors. For a quantitative measure of errors, the Metro tool [36] allows, in a similar manner, to compare two meshes (e.g., the original, full-resolution mesh, and its simplified version)

based on the computation of a point-surface distance and returns the maximum and mean distance, as well as the variance (RMS). The lower this error is, the better the quality of the simplified object is.

Since our interest is in improving the perceptual quality of the models, another measure, the perceptual error [186], is also employed. It is based on the structural similarity metric (SSIM). The latter was proposed based on the observation that human visual perception is highly adapted to extract structural information in a scene [186]. The inverse of this metric can be employed as an error measure, because a lower similarity between the simplified mesh and the initial mesh implies a higher error. Since this measure is meant to be applied on images, therefore, to compute it, images are captured over the simplified models of objects from various viewpoints and are compared with the images of the initial, non-simplified object from the same viewpoints. In the current case, the object is turned around the *y* axis for 360°, 15° at a time, and images are collected over the resulting 25 viewpoints. This measure is reported for each deformed instance of an object as an average over these viewpoints. These errors are briefly described in the following sections.

3.4.1. Quantitative Error Computation Using Metro

The Metro tool [36] is used to compare the triangulated mesh for each un-simplified (original) object with its simplified representations. As a triangular mesh is represented by a set of points (vertices) and by a set of triangles (faces) describing how the vertices are linked together, the approximation error between two meshes can be found using the distance between the surfaces of the two meshes. Supposing that $\mathbf{S_1}$ is the surface of the first mesh, $\mathbf{S_2}$ is the surface for the second mesh and $\boldsymbol{p}$ is a given point on the surface $\mathbf{S_1}$. The distance between $\boldsymbol{p}$ and $\mathbf{S_2}$ is defined by [192]:

$$\mathbf{e}(\boldsymbol{p}, \mathbf{S_2}) = \mathbf{min}\ \mathbf{d}(\mathbf{p}, \mathbf{p}'), \quad \boldsymbol{p}' \in \boldsymbol{S_2} \tag{11}$$

where $\mathbf{d}$ is the Euclidean distance between the two points $\mathbf{p}$ and $\mathbf{p}'$ (Figure 3.11).

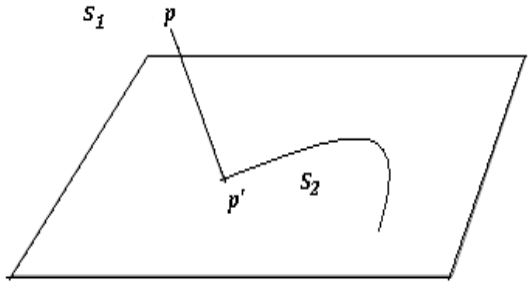


Figure 3.11. The distance between $\mathbf{p}$ and $\mathbf{S_2}$ is the Euclidian minimal distance between $\mathbf{p}$ and $\mathbf{p'}$, where $\mathbf{p'}$ belongs to $\mathbf{S_2}$ (adapted from [192]).

The distance between the two surfaces $\mathbf{S_1}$ and $\mathbf{S_2}$ is defined by [192]:

$$\mathbf{E}(\mathbf{S_1}, \mathbf{S_2}) = \mathbf{max}\, \mathbf{e}(\mathbf{p}, \mathbf{S_2}) \quad \mathbf{p} \in \mathbf{S_1} \tag{12}$$

It is important to note that the definition of the distance between two surfaces is not symmetric, as it can be observed in Figure 3.12.

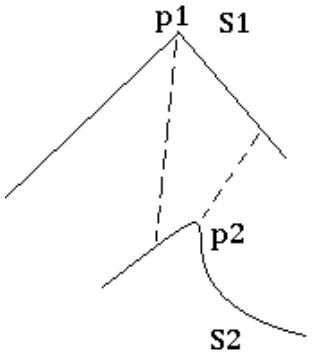


Figure 3.12. In this example, $\mathbf{d(S_2, S_1)}$ is smaller than $\mathbf{d(S_1, S_2)}$, since here $\mathbf{d(p_2, S_1)} < \mathbf{d(p_1, S_2)}$ (adapted from [192]).

$$\mathbf{E}(\mathbf{S_1}, \mathbf{S_2}) \neq \mathbf{E}(\mathbf{S_2}, \mathbf{S_1}) \tag{13}$$

The Hausdorff distance is obtained by taking the maximum of $\mathbf{E}(\mathbf{S_1}, \mathbf{S_2})$ and $\mathbf{E}(\mathbf{S_2}, \mathbf{S_1})$, and this distance is defined as:

$$\mathbf{d_h}(\mathbf{S_1}, \mathbf{S_2}) = \mathbf{max}\, (\mathbf{E}(\mathbf{S_1}, \mathbf{S_2}), \mathbf{E}(\mathbf{S_2}, \mathbf{S_1})) \tag{14}$$

The mean distance between the two surfaces $\mathbf{S_1}$ and $\mathbf{S_2}$ is defined as the surface integral of the distance divided by the area of $\mathbf{S_1}$ as [36]:

$$\mathbf{d_{mean}(S_1, S_2)} = \frac{1}{|\mathbf{S_1}|} \int \mathbf{e(p, S_2)ds} \tag{15}$$

The root mean square error is defined by [192]:

$$\mathbf{d_{rms}(S_1, S_2)} = \sqrt{\frac{1}{|\mathbf{S_1}|} \int \mathbf{e(p, S_2)^2 ds}} \tag{16}$$

To compute numerically the values of this error measure, we used the Metro tool [36]. It is a tool that allows to compare the difference between a triangulated mesh and its simplified representation without requiring knowledge on the simplification approach used to build the meshes. It evaluates the difference between two meshes based on the approximate distances defined in this section and returns the numerical results of the surface area and the mesh volume of the two objects compared. It also provides the number of connected components of each mesh, the total volume of the difference between the two meshes, and the maximum, the mean and the mean square root error. This tool can also display information on the number of triangles and number of vertices for each mesh. In our case, we only make use of the computed maximal distance, the mean and the mean square errors computed between the two meshes.

3.4.2. Perceptual Errors

The structural similarity index measure (SSIM) is a technique to quantify the visible errors between a reference image and a deformed image using properties of the human visual system [37]. As this measure can only be applied to images, in the context of this thesis, images are captured over the simplified 3D models of objects from various viewpoints. These are compared with the images of the initial, non-simplified 3D object from the same viewpoints, as discussed at the end of this section.

To compare a reference and a deformed image, the SSIM-based method uses three images: a luminance image, a contrast image and a structure image. The luminance is the phenomenon where the visibility is lower in the bright regions in the deformed image. The contrast is the phenomenon where the visibility is lower when we have texture

regions in the deformed image. The structural information in an image is defined as those attributes that represent the objects in the scene, independently of the luminance and contrast. First, the luminance images are compared, then the contrast images are compared, and finally the structural images are compared. Then the three comparisons are combined to compute the similarity measure. Figure 3.13 shows a diagram illustrating the quality assessment system, adapted from [37]. In the figure, $\mathbf{im_1}$ and $\mathbf{im_2}$ represent the two images that we want to compare. All the following mathematical formulas are extracted from [37].

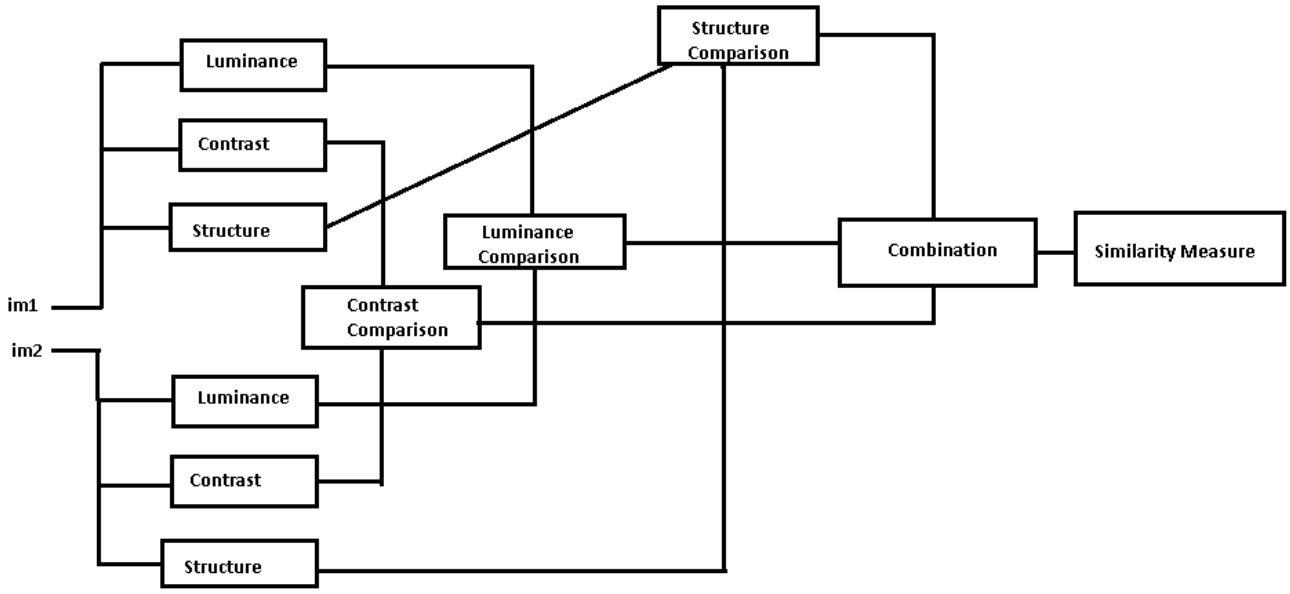


Figure 3.13. Diagram of the quality assessment system adapted from [37].

Assuming that $\mu_{\mathbf{im1}}$ is the estimation of the mean intensity of the image $\mathbf{im_1}$ and $\mu_{\mathbf{im2}}$ is the estimation of the mean intensity of the image $\mathbf{im_1}$, where:

$$\mu_{\mathbf{im1}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x_i} \tag{17}$$

$$\mu_{\mathbf{im2}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{y_i} \tag{18}$$

The luminance comparison function $\mathbf{l}(\mathbf{im_1}, \mathbf{im_2})$ is a function of $\mu_{\mathbf{im1}}$ and $\mu_{\mathbf{im2}}$, and it is given by:

$$\mathbf{l(im_1, im_2)} = \frac{\mathbf{2\mu_{im1}\mu_{im2}+C_1}}{\mathbf{\mu_{im1}^2+\mu_{im2}^2+C_1}} \quad (19)$$

where $\mathbf{C_1}$ is a constant to avoid instability when $\mathbf{\mu_{im1}^2 + \mu_{im2}^2}$ is very close to zero.

$$\mathbf{C_1} = (\mathbf{K_1\, L})^2 \quad (20)$$

$\mathbf{K_1}$ is a small constant and L is the rang of the pixel values (L=255 for 8-bit grayscale image).

The estimation of the signal contrast is given by the square root of variance:

$$\mathbf{\sigma_{im1}} = \left(\frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{x_i} - \mu_{\mathbf{im1}})^2\right)^{1/2} \quad (21)$$

$$\mathbf{\sigma_{im2}} = \left(\frac{1}{N-1}\sum_{i=1}^{N}(\mathbf{y_i} - \mu_{\mathbf{im2}})^2\right)^{1/2} \quad (22)$$

The contrast comparison function $\mathbf{c(im_1, im_2)}$ is defined by:

$$\mathbf{c(im_1, im_2)} = \frac{\mathbf{2\sigma_{im1}\sigma_{im2}+C_2}}{\mathbf{\sigma_{im1}^2+\sigma_{im2}^2+C_2}} \quad (23)$$

In this formula, $\mathbf{C_2}$ is a constant to avoid instability when $\mathbf{\mu_{im1}^2 + \mu_{im2}^2}$ is very close to zero and calculated as:

$$\mathbf{C_2} = (\mathbf{K_2\, L})^2 \quad (24)$$

where $\mathbf{K_2}$ is a small constant and $\mathbf{L}$ is the rang of the pixel values (L=255 for 8-bit grayscale image). The structure comparison function $\mathbf{s(im_1, im_2)}$ is defined as:

$$\mathbf{s(im_1, im_2)} = \frac{\mathbf{2cov_{im1im2}+C_3}}{\mathbf{\sigma_{im1}\sigma_{im2}+C_3}} \quad (25)$$

where $\mathbf{C_3}$ is a constant to avoid instability when $\mathbf{\sigma_{im1}\sigma_{im2}}$ is very close to zero, and computed as:

$$\mathbf{C_3 = C_2/2} \quad (26)$$

and $\mathbf{cov_{im1im2}}$ is the covariance of $\mathbf{im_1}$ and $\mathbf{im_2}$ given by:

$$\mathbf{cov_{im1im2}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x_i} - \mu_{\mathbf{im1}})(\mathbf{y_i} - \mu_{\mathbf{im2}}) \quad (27)$$

By combining the equations (13), (17) and (19), the structural similarity (SSIM) index between $\mathbf{im_1}$ and $\mathbf{im_2}$ [193] :

$$\mathbf{SSIM(im_1, im_2)} = \frac{(2\mu_{\mathbf{im1}}\mu_{\mathbf{im2}} + \mathbf{C1})(2\mathbf{cov_{im1im2}} + \mathbf{C_2})}{(\mu_{\mathbf{im1}}^2 + \mu_{\mathbf{im2}}^2 + \mathbf{C1})(\boldsymbol{\sigma}_{\mathbf{im1}}^2 + \boldsymbol{\sigma}_{\mathbf{im2}}^2 + \mathbf{C_2})} \quad (28)$$

The perceptual error is given by:

$$\mathbf{Perceptual\ error} = \mathbf{1 - SSIM(im_1, im_2)} \quad (29)$$

Because this measure is meant to be applied to images, we apply it to captured images over the simplified models of 3D objects from various viewpoints and we compare with the images of the initial, non-simplified object (or with the FEM model when performing the comparison with this classical method) from the same viewpoints. To achieve this, the object is turned around the *Y* axis for 360°, 15° at a time, and images are collected over the resulting 25 viewpoints. This measure is then reported for each deformed instance of an object as an average over these 25 viewpoints.

CHAPTER 4

4. EXPERIMENTAL RESULTS

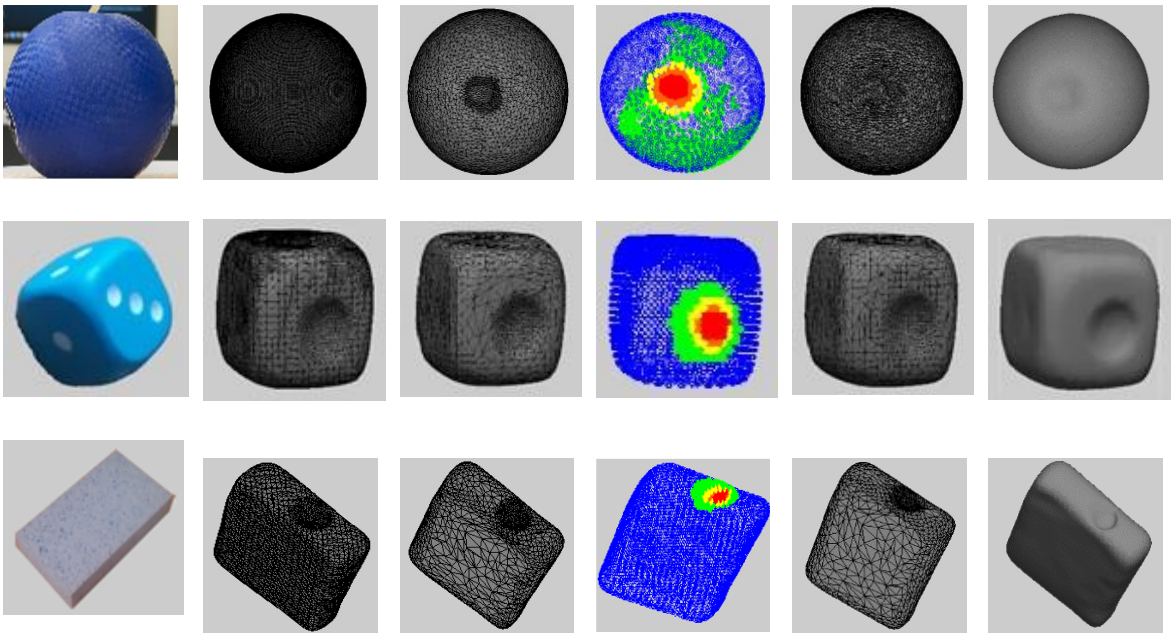
This chapter shows results obtained by our proposed method and discusses the implementation issues and the choice of various parameters involved. To implement our method, all the computations were performed on a Pentium III, 2Ghz CPU, 64-bit operating system, 4Ghz memory machine using MATLAB.

4.1. Preprocessing

This section presents experimental results for parameter tuning and validation of the preprocessing steps of our method. It is aligned with objective 2 (chapter 1, section 1.1) and contribution 2 (chapter 1, section 1.3).

The proposed method (chapter 3, sections 3.2.3, 3.2.4, 3.3.1 and 3.3.2) for modeling and predicting deformable object shapes is validated by using a soft rubber ball, a rubber-coated foam cube, and a foam sponge (see Figure 4.1 a). Starting from the cleaned mesh, instead of using the entire collected point clouds (Figure 4.1 b), an alignment procedure is performed to ensure a simpler interpretation of the deformation and a similar treatment of the deformation over the surface of an object (Chapter 3, section 2.3.4), then a selectively-densified mesh (Figure 4.1 c) is first constructed, in which the area around the point of interaction between the probing tip and the object surface is preserved at higher resolution, while the other areas are simplified. This is achieved by adapting the QSlim algorithm to only simplify points that are not the interaction point with the probing tip and its immediate neighbors.

Figure 4.1 d shows the stratified sampled data chosen for neural-gas mapping. Figures 4.1 e and 4.1 f show the neural-gas-tuned simplification obtained for each object and the neural-gas-tuned simplified objects.



(a) **(b)** **(c)** **(d)** **(e)** **(f)**

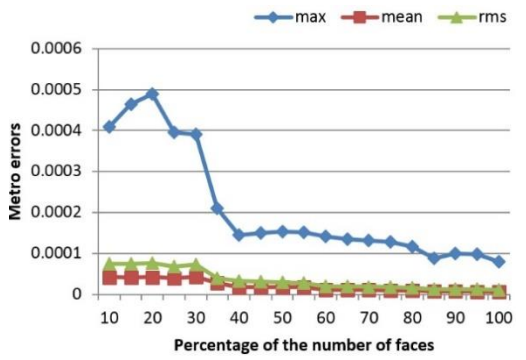
Figure 4.1. (a) Object model; (b) initial object mesh; (c) mesh with higher density in the deformed area; (d) stratified sampled data for neural-gas mapping; (e) neural-gas-tuned simplification; and (f) neural-gas-tuned simplified object.

4.1.1. Choice of Simplification Parameters and Obtained Simplification Results

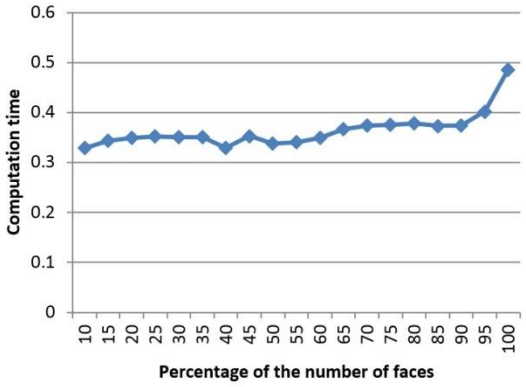
The two important parameters used in the simplification process are the percentage of the number of faces chosen to simplify the object and the number of neighbors chosen around the probing tip.

The proposed simplification procedure ensures a uniform representation of the object by defining an equal number of faces, representing 40% of the faces in the initial model for all the instances of a deformed object. This value (i.e. 40%) is chosen by monitoring the evolution of the errors and the computation time for an increasing percentage (from 10% to 100%) and finding the best compromise between the two, as illustrated in Figure 4.2. This figure shows the evolution of the error measures detailed in sections 3.5.1 and 3.5.2, namely of the Metro errors (Figure 4.2 a), the perceptual error (Figure 4.2 b), and of the computation time (Figure 4.2 c), error measures calculated as an average over the entire surface of the three objects under study, with the percentage of faces used for simplification. One can notice that, as expected, the error decreases with an increase in

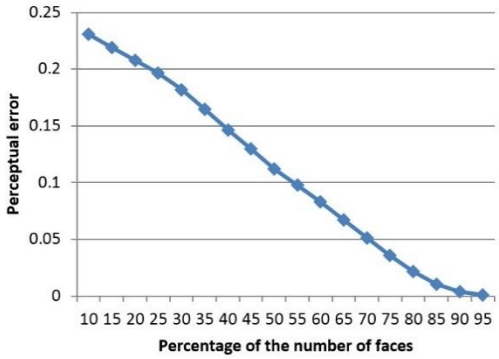
the number of faces. At the same time, the computation time increases slightly with a larger number of faces, and therefore the best compromise is found between these two opposing criteria. The abrupt descent in Metro error and the relatively low computation time around 40% of the number of faces guided our choice for the selection of this parameter.



(a)



(b)



(c)

Figure 4.2. Evolution of: (a) Metro error; (b) perceptual error; and (c) computation time with the number of faces used for data simplification.

A second parameter that affects the simplification algorithm is the number of immediate neighbors that are preserved around the probing tip. Initially (in our publication [188]), we used 12 immediate neighbors for all the objects. In the current work, 12-degree immediate neighbors are preserved for the rubber ball and 8-degree immediate neighbors for the foam sponge and the cube. Only values within this range (8 to 12) are tested, because they allow us to correctly capture the entire deformed area over the 3D point clouds collected. Within this range, the number of neighbors is chosen by selecting the model that results in the lowest perceptual error (or the highest perceptual similarity) with respect to the original, non-simplified mesh. Table 4.1 shows the evolution of the perceptual errors with the size of the neighborhood (***n***) around the probing tip and justifies the choice of the neighborhood size based on the highest similarity.

Table 4.1. Perceptual similarity measures according to neighborhood size, ***n***.

	Ball	**Cube**	**Sponge**
	n = 8/n = 10/n = 12	***n = 8/n = 10/n = 12***	***n = 8/n = 10/n = 12***
Overall perceptual similarity (%)	71.38/71.38/71.38	79.3/78.87/78.5	83.75/83.37/82.81
Perceptual similarity (%) in the deformed zone	84.65/84.11/**85.00**	**93.67**/93.48/93.49	**91.24**/90.82/90.93

Table 4.2. Error measures and computing time for objects after simplification.

Object Name	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object**
	Max/Mean/Rms		**Max/Mean/Rms**		
Ball	17.672/2.948/3.713	0.2862 (71.38%)	46.277/4.496/7.548	0.1407 (85.93%)	0.139 s
Cube	13.128/0.600/2.265	0.2070 (79.30%)	42.755/3.321/6.765	0.0633 (93.67%)	0.033 s
Sponge	4.088/0.113/0.540	0.1625 (83.75%)	42.471/3.159/6.388	0.0876 (91.24%)	0.058 s
Average	**11.629/1.220/2.173**	**0.2186 (78.15%)**	**41.834/3.659/6.900**	**0.0972 (90.28%)**	**0.077 s**

Table 4.2 (for more details see Annex A, Tables 4.2.1, 4.2.2 and 4.2.3) presents the Metro and perceptual errors computed over the entire surface of the object and over the deformed area for all the objects. The overall perceptual similarity obtained is 71.38% for the ball, 79.30% for the cube, 83.75% for the sponge, with an average of 78.14% for the three objects together. Over the deformed area of the object, the average perceptual similarity is 85.93% for the ball, 93.67% for the cube, 91.24% for the sponge and 90.28% for the three objects together. The results for the selective simplification around the probing tip with the above detected values for the parameters are shown for the three test objects in Figure 4.1c.

4.1.2. Clustering and Sampling Results

After obtaining the selectively-densified mesh as described in the previous section, a stratified sampling technique, as detailed in section 3.3.1, is employed to only retain a subset of data for neural-gas fitting. This clustering and sampling process is applied to reduce the complexity of each instance of deformed objects and to better characterize the deformation area. In particular, the mesh is initially clustered according to the Hausdorff distance with respect to the initial non-deformed mesh. It is desired that the highest possible number of points from the deformed zone is situated in the farthest cluster.

The two parameters to be tuned in the clustering and sampling process are the number of clusters and the percentage of points (3D coordinates) sampled from each cluster.

To identify the number of clusters to be employed, in an initial attempt (in our publication [188]), during testing, the number of points of the deformed zone in the farthest cluster was monitored, and the process was stopped once the largest number of points in the deformed zone was found in this cluster. A number of five clusters was thus identified to ensure the best results. Subsequently, in this section, a more thorough analysis led to a better definition of the area around the deformation zone by increasing the number of clusters and regrouping the ones at the largest distance to cover the entire deformation area. In this case, an equal number of five clusters were used, but by identifying seven clusters and then regrouping the three farthest together.

Figure 4.3 shows the difference, for each of the test objects, between the use of five and seven clusters. One can notice that in all three cases, the use of seven clusters and the combination of clusters at the farthest distance more accurately represents the deformation area. For example, for the case of the ball, the combination of the red, fuchsia, and orange clusters in Figure 4.3 b results in a more accurate representation of the deformation area than the black cluster in Figure 4.3 a. The same conclusion can be reached by comparing Figure 4.3 c and d to Figure 4.3 e and f, respectively.

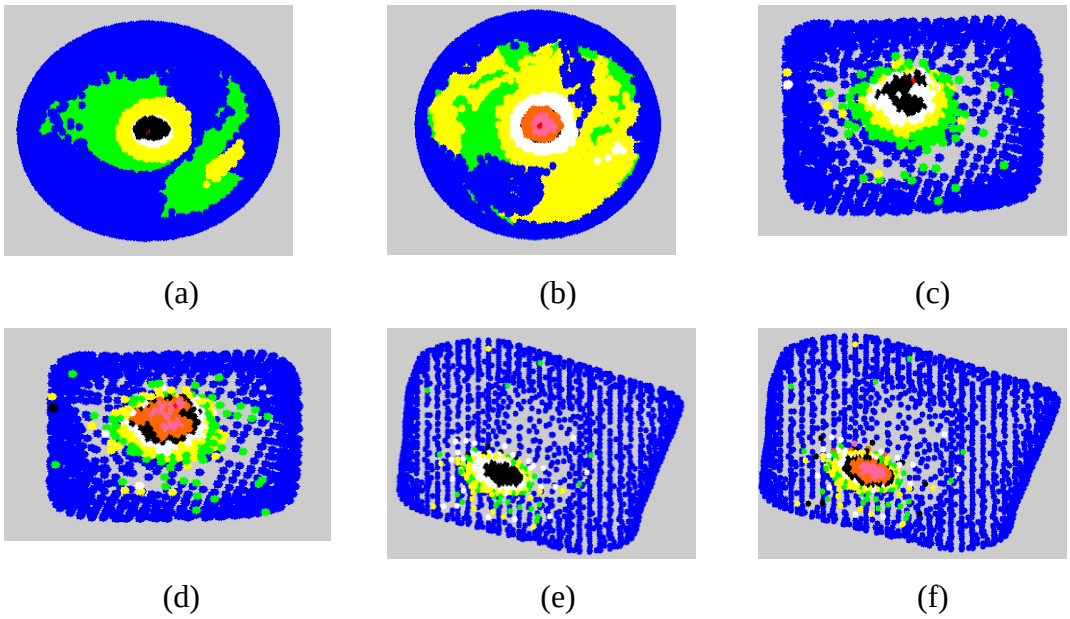


Figure 4.3. Clusters for ball: (a) 5 clusters; (b) 7 clusters; for cube: (c) 5 clusters and (d) 7 clusters; and for sponge: (e) 5 clusters and (f) 7 clusters.

Figure 4.1 d shows the clusters, with red points belonging to the farthest cluster from the initial object (deformed zone) and orange, yellow, green, and blue points being increasingly closer to the initial mesh (blue = perfect match), for the combination of 90% from the farthest (red) cluster, 80%, 70%, and 60%, respectively, from the 2nd, 3rd, and 4th cluster, and 50% from the closest distanced cluster points (blue). While a higher percentage might achieve better results, it will also result in a higher computation time. Table 4.3 (for more details see Annex B, tables 4.3.1, 4.3.2 and 4.3.3) shows the Metro and perceptual errors computed over the entire surface of the object and over the deformed area. The perceptual similarity obtained is 79.76% for the ball, 73.06% for the

cube, 76.54% for the sponge, with an average of 76.46% for the three objects together. Over the deformed area of the object, the average of perceptual similarity is 93.135% for the ball, 94.53% for the cube, 90.64% for the sponge resulting an average of 92.77% for the three objects together. By comparing the similarity between the objects after simplification and after clustering and sampling, one can notice that the average of similarity for the three objects increases in the deformed zone (increase of 2.49%) and decreases slightly over the whole surface of the object (decrease of 1.68%).

Table 4.3. Error measures and computing time for objects after clustering and sampling.

Object Name	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object**
	Max/Mean/Rms		**Max/Mean/Rms**		**Clustering/Sampling**
Ball	17.647/3.000/3.824	0.2023 (79.76%)	25.73/0.83/2.31	0.0687 (93.14%)	0.083 s/0.037 s
Cube	13.111/0.353/1.725	0.2693 (73.06%)	39.37/2.63/6.27	0.0547 (94.53%)	0.025 s/0.028 s
Sponge	4.075/0.066/0.409	0.2346 (76.54%)	42.078/4.9/6.89	0.0936 (90.64%)	0.039 s/0.031 s
Average	**14.104/1.140/1.986**	**0.2350 (76.46%)**	**35.73/2.79/5.16**	**0.0723 (92.77%)**	**0.049 s/ 0.032 s**

4.1.3. Neural Gas Results

The stratified sampling process detailed in the previous section is not sufficient to adequately capture the object shape, as the fine differences around the deformed zone and over the object surface (see Figure 3.3 c-d in Chapter 3) might not be appropriately represented, which is the reason why a tuning of the selectively densified mesh is also executed using a neural gas network. A neural gas network is adapted, as explained in sections 3.3.2 and 3.3.3, and tuned over the resulting sampled data in section 4.1.2 for 25 iterations. It uses a number of neurons equal to the number of points sampled from the different clusters and a number of faces equal to the average of the number of faces over the selectively densified meshes. The object model is then constructed using the method proposed in section 3.3.9. Figure 4.4 presents the color-coded comparison with the initial full resolution object obtained using CloudCompare for each of the three objects after the construction of the selectively-densified mesh and after the neural gas tuning. One can

notice that a very good match is associated in both cases with the deformed area (shown in blue), but that the errors are lower in each case after neural gas tuning due to a better distribution of faces over the surface of the object.

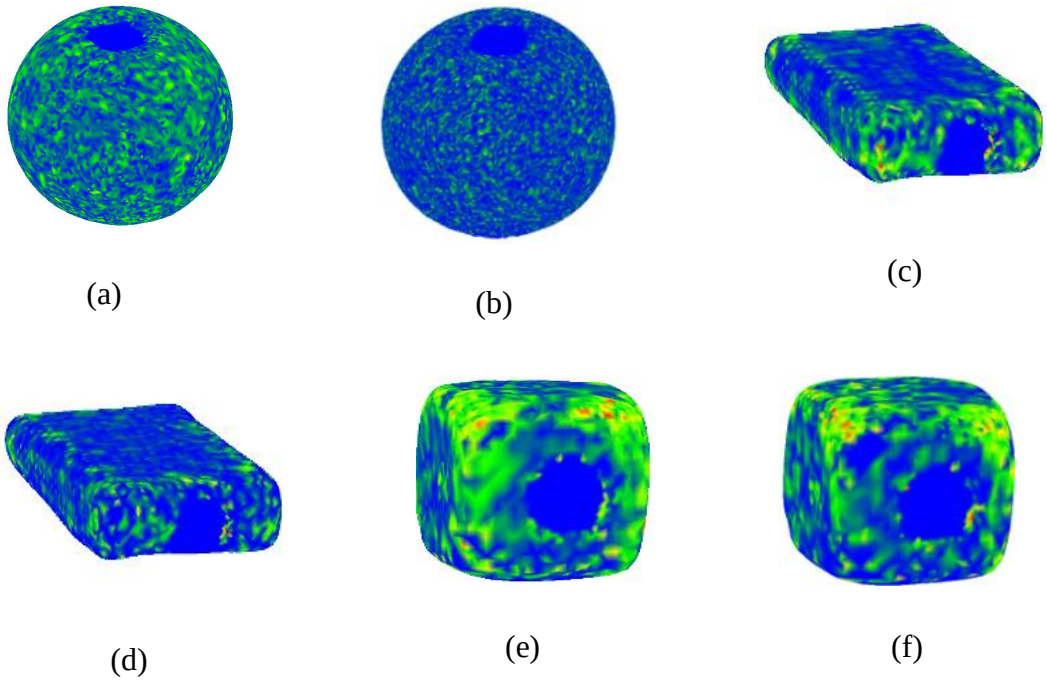
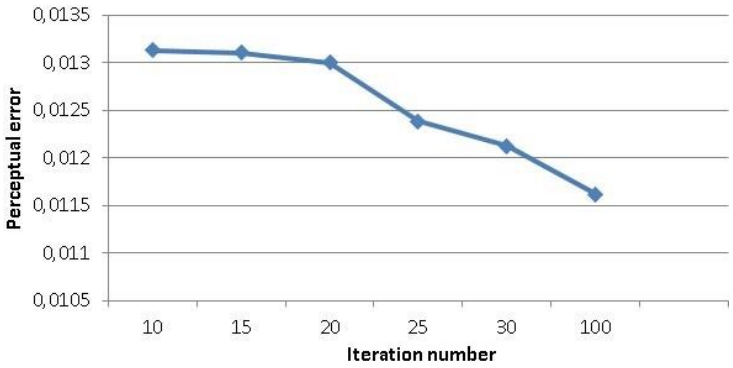
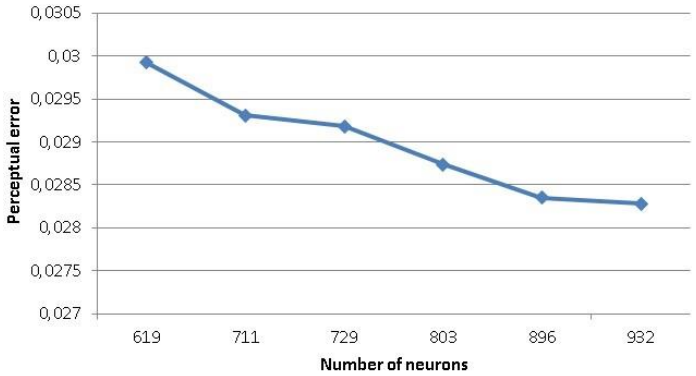


Figure 4.4. Color-coded results for the ball, sponge, and cube with respect to the initial full-resolution model: (a) selectively-densified mesh around the probing point for a ball for F_{Pa} = 4.5 N, a_{Pa} = 10°; (b) final mesh for a ball for F_{Pa} = 4.5 N, a_{Pa} = 10°; (c) selectively-densified mesh around the probing point for a sponge for F_{Pa} = 3.7 N, a_{Pa} = 49°; (d) final mesh for a sponge for F_{Pa} = 3.7 N, a_{Pa} = 49°; (e) selectively-densified mesh around the probing point for a cube for F_{Pa} = 5 N, a_{Pa} = 85°; and (f) final mesh for a cube for F_{Pa} = 5 N, a_{Pa} = 85°.

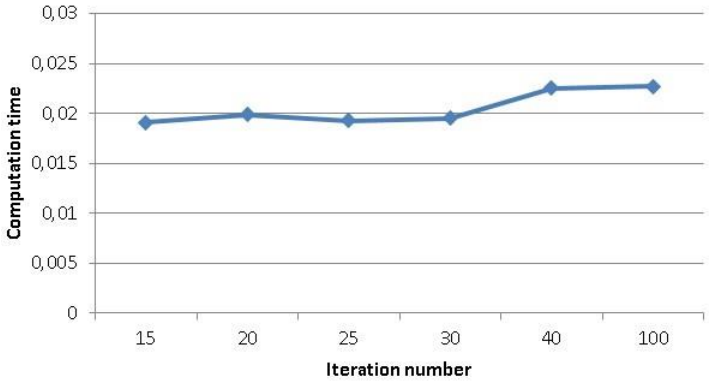
One of the parameters to be tuned is the number of iterations for the neural gas. It was obtained by monitoring the evolution of the errors and of the computation time for an increasing number of neurons and number of iterations (10, 15, 20, 25, 30, 35, 40, 45, 50, 75 and 100).



(a)



(b)



(c)

Figure 4.5. Evolution of perceptual error: (a) in relation with the number of iterations; (b) in relation with the number of neurons; and (c) evolution of computation time with the number of iterations.

Figure 4.5 a, shows the evolution of the perceptual errors in the deformed area in relation with the increase in the number of iterations, Figure 4.5 b shows the evolution of the perceptual errors in the deformed area in relation with an increasing number of neurons in the neural gas map and Figure 4.5 c shows the evolution of the computation time. It can be noticed that an increase in the number of iterations leads to a decrease in the perceptual error. At the same time, the computation time increases with a larger number of iterations, and therefore the best number of iterations is found as a compromise between these two opposing criteria. As a consequence, the number of iterations chosen is 25.

The Metro and perceptual errors computed over the entire surface of the object and over the deformed area are presented in Table 4.4 (for more details, see Annex C, Tables 4.4.1, 4.4.2 and 4.4.3). An average perceptual similarity of 91.66% is obtained for the ball, 81.65% for the cube, 80.02% for the sponge and an overall 84.45% over the three objects. Over the deformed area of the object, the average perceptual similarity is of 98.474% for the ball, 97.68% for the cube, 91.63% for the sponge, resulting in an overall average of 95.93%.

Table 4.4. Error measures and computing time for objects after neural gas fitting.

Object Name	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object /Iteration**
	Max/Mean/Rms		**Max/Mean/Rms**		**Neural Gas**
Ball	54.2/1.42/2.37	0.0830 (91.66%)	11.37/0.16/0.78	0.0155 (98.47%)	25.966 s
Cube	23.33/2.05/3.76	0.1835 (81.65%)	26.16/0.30/1.51	0.0139 (97.68%)	5.047 s
Sponge	36.46/2.65/4.9	0.1991 (80.02%)	40.67/1.01/3.38	0.0872 (91.63%)	7.056 s
Average	**37.99/2.04/11.03**	**0.1552 (84.45%)**	**20.06/0.49/1.89**	**0.0388 (95.93%)**	**12.689 s**

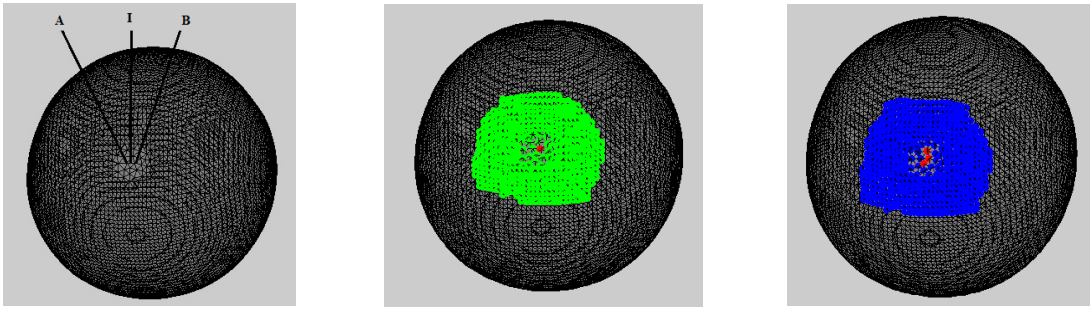
When comparing the similarity between the objects before and after neural gas using, one can notice that the average perceptual similarity for the three objects increases in the deformed zone (increase of 5.65%) and also on the surface of the whole object (increase of 6.30%). For example, for the ball, the overall perceptual similarity obtained for the ball

is 91.66% (an improvement of 20.28% compared to the simplification) and 98.474% (an improvement of 12.54%) over the deformed area.

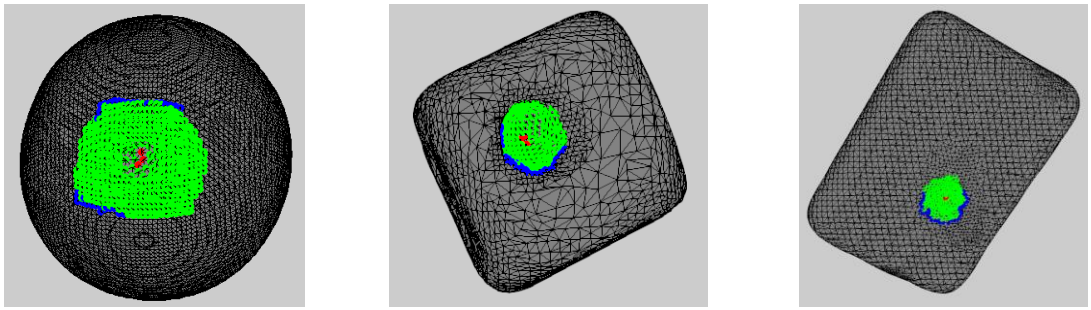
4.1.4. Choice of Number of Interaction Points

In our method, we used one interaction point to represent the point of contact between the probing tip and the object. To justify this choice and evaluate impact of using more than one interaction point, we tested as well the ray-based points interaction method [50-54]. This method considers the interaction with the object as a line or polygon [54]. For example, given the point of interaction of the probe with the object (denoted **I**), the two points with the shortest distance from point **I** are identified (Figure 4.6) based on the Euclidean distance between **I** and all the other points on the surface. Additional points could be considered as well, but the computation time will be longer which is not desired in the context of this work. Figures 4.6 a, b and c show respectively the non-simplified deformed ball with the three interaction points **I, A** and **B**, the deformed zone area around the probing tip when one interaction point is used and when three interaction points (**I, A** and **B**) are used. Figures 4.6 d, e and f illustrate the difference between the deformed zone area when using one interaction point (green color) and when using three interaction points (blue color), for the three test objects respectively. One can notice that the difference between the blue and green colored areas occurs far from the point of interaction of the probing tip.

To study the impact of the two interaction methods (one point interaction and ray-based) on the preprocessing steps of our method, we compared the quality of the object models obtained using these two methods. For experimentation we used 10 deformed instances for the ball, 8 instances for the cube and 8 instances for the sponge and also employed 12 neighbors around the interaction points for the ball and, 8 neighbors for the cube and the sponge (as detailed in section 4.1.1). This leads clearly to a larger number of points when using 3 interaction points than the number of points obtained one interaction point is used. It is thus expected that a slightly better accuracy will be obtained when using 3 interaction points.



(a) (b) (c)



(d) (e) (f)

Figure 4.6. (a) Original deformed ball; (b) deformed zone around the probing tip with one interaction point; (c) deformed zone around the probing tip with three interaction points; difference between the two deformed zones for (d) the Ball; (e) for the cube; (f) for the sponge.

To confirm this, we computed the Metro and perceptual error between objects obtained based on one interaction point and based on three interaction points. Using one interaction point, the overall perceptual similarity achieved is on average 81.52% over the entire surface of the object and roughly 95.47% over the deformed area; while using three interaction points, results in an average 82.06% over the entire surface and 96.10% over the deformed area. It can be noticed that, using three interaction points, we indeed obtained a slight, but insignificant improvement both for the entire object (0.54%) and in the deformed area (0.63%). This is the reason why we chose to perform the subsequent experimentation for one interaction point only.

4.2. Evaluation of the Impact of Preprocessing Steps

As noticed in the first three steps of our method, there are gains and losses resulting from the choice of simplification, clustering, sampling and neural gas tuning, from the similarity point of view, both on entire object surface and in the deformed area. Table 4.5

summarizes the average Metro and perceptual errors after each preprocessing step, and Table 4.6 presents the gains, losses, and the impact of each step on our method.

Table 4.5. Average Metro and perceptual errors for all preprocessing steps.

	Simplification	**Clustering & Sampling**	**Neural Gas**
Overall Metro Error (e^{-3})	Max/Mean/Rms 11.629/1.220/2.173	Max/Mean/Rms 14.104/1.140/1.986	Max/Mean/Rms 37.99/2.04/11.03
Overall Perceptual Error (Similarity %)	0.2186 (78.16%)	0.235 (76.47%)	0.1552 (84.46%)
Metro Error in Deformed Area (e^{-5})	Max/Mean/Rms 41.834/3.659/6.900	Max/Mean/Rms 35.73/2.79/5.16	Max/Mean/Rms 20.06/0.49/1.89
Perceptual Error (Similarity%) in Deformed Area	0.0972 (90.28%)	0.0723(92.77%)	0.0388 (95.93%)
Computation Time/Object	0.077 s	Clustering/Sampling 0.049 s/0.032 s	12.68 s/iteration

Table 4.6. Impact of each preprocessing step.

	Simplification	**Clustering & Sampling**	**Neural Gas**
Gains	Low computation time of 0.077 seconds.	2.77% more similarity in the deformed area. Low computation time of 0.049 seconds for clustering and 0.032 seconds for sampling.	7.99% more similarity overall the surface of the object. 3.17% more similarity in the deformed area.
Losses	Lost similarity of 21.84% over the surface of the object and 9.72% in the deformed area due to the simplification of 40% (but results in gains in computation time for the modeling, as it will be further shown in the thesis).	Lost 1.69% of similarity overall the surface of the object during sampling.	No losses but slightly higher computation time (12.68 seconds per iteration).
Complexity	Order of complexity: O(n), n is the number of points represent the object.	Order of complexity: O(nk), n is the number of points represent the object and k is the number of clusters.	Order of complexity: $O(n^2)$, n is the number of points represent the object.

The simplification, clustering, and stratified sampling are executed in an attempt to reduce the computation in the subsequent steps especially in neural gas and further on for the object shape deformation modeling.

4.3. Model Selection and Learning by Neural Network

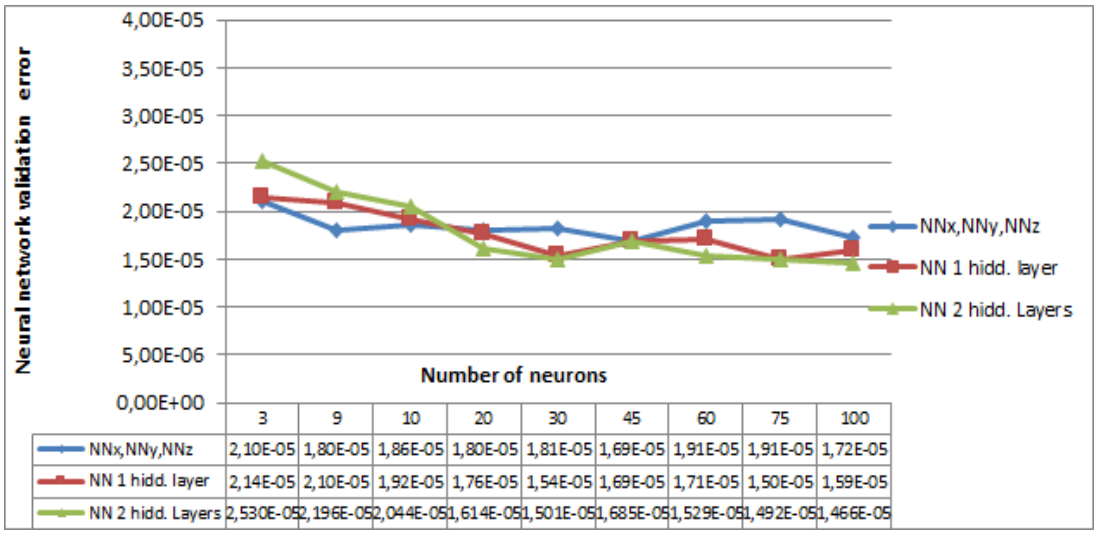
As described in section 3.3.4, we proposed to use a series of feedforward neural networks to capture implicitly the behavior in terms of surface deformations represented as neural gas nodes and the interaction characteristics (i.e. force magnitude, angle of application, point of application).

Several neural network architectures were studied and tested as described in Chapter 3, sections 3.3.4 - 3.3.6. To train each type of architecture, experiments were performed using different forces (magnitude, angle, point of application) as the ground truth and the corresponding deformed instances for each object. For example, in the case of the ball each original deformed ball used for training had 5670 vertices and 11336 faces. Following the preprocessing steps (i.e. simplification at 40%, clustering, sampling and neural gas), the number of vertices is reduced for each deformed instance of ball to 1681 vertices. As such, for the 10 deformed instances, a total number of 16810 vertices is obtained, from which 50% (8405 vertices) is used for the training, 5% (841 vertices) for validation and the remaining 45% for testing (7564 vertices).

In order to identify the best type of neural network architecture to use for object modeling and shape deformation prediction method, we performed experiments during which we increased the number of neurons in the hidden layer between 3 and 100 (i.e. 3, 9, 10, 20, 30, 45, 60, 75 and 100) for the case when using three distinct networks, one for each coordinate, and in the case when using a single network with one hidden layer. We then tested, as described in Chapter 3 (section 3.3.7) the case when using one neural network with three layers (two hidden layers and one output layer). In this case, we also tested for an increasing number of neurons in the two hidden layers between 3 and 100, while maintaining always a lower number of neurons in the second layer to preserve the pyramid structure of the multi-layer feedforward architecture, as suggested in the literature (i.e. we tested 2-1, 6-3, 7-3, 14-6, 22-8, 35-10, 45- 15, 55-20 and 82-18). For each test, we computed the validation and testing errors, as well as the training time.

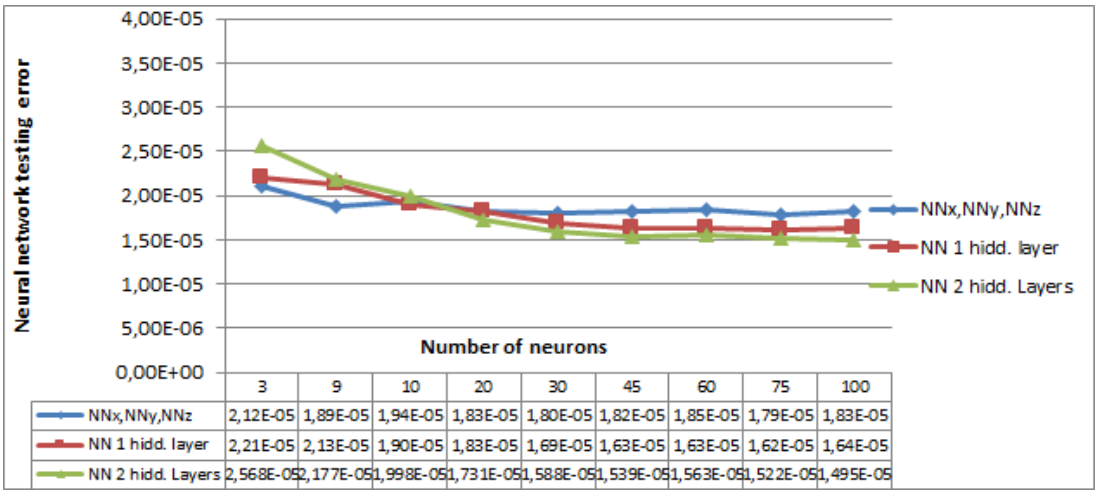
Figure 4.7 a shows the evolution of the validation error obtained using the different types of network architectures studied. Note that the minimal average of validation error using three networks together (with 45 neurons in the hidden layer) is $1.69e^{-5}$ (Figure 4.7 a, graph in blue). The minimum mean validation error using a single network with a hidden layer with 75 neurons is $1.50e^{-5}$ (Figure 4.7 a, graph in red). The minimum average of validation error for a single network with two hidden layers with 100 neurons is $1.46e^{-5}$ (Figure 4.7 a, graph in green; this case corresponds to an architecture with 82 neurons in the first hidden layer and 18 neurons in the second hidden layer). Figure 4.7 b shows the evolution of testing error for the different types of network architectures. The minimal testing error is $1.79e^{-5}$ when using three neural networks together, $1.62e^{-5}$ when using a single neural network with one hidden layer and $1.49e^{-5}$ when using a single neural network with two hidden layers. From the tested architectures, the smallest testing error is $1.46e^{-5}$ using a single network with two hidden layers using a total of 100 neurons (i.e. with 82 neurons in the first layer and 18 neurons in the second hidden layer). This was therefore identified as the best architecture and used in further experiments.

Figure 4.7 c shows the evolution of training time. It can be noticed that the training time increases with an increase in the number of neurons for all the network architectures tested. There is therefore a trade-off to be considered between the number of neurons to be trained and the computation time.



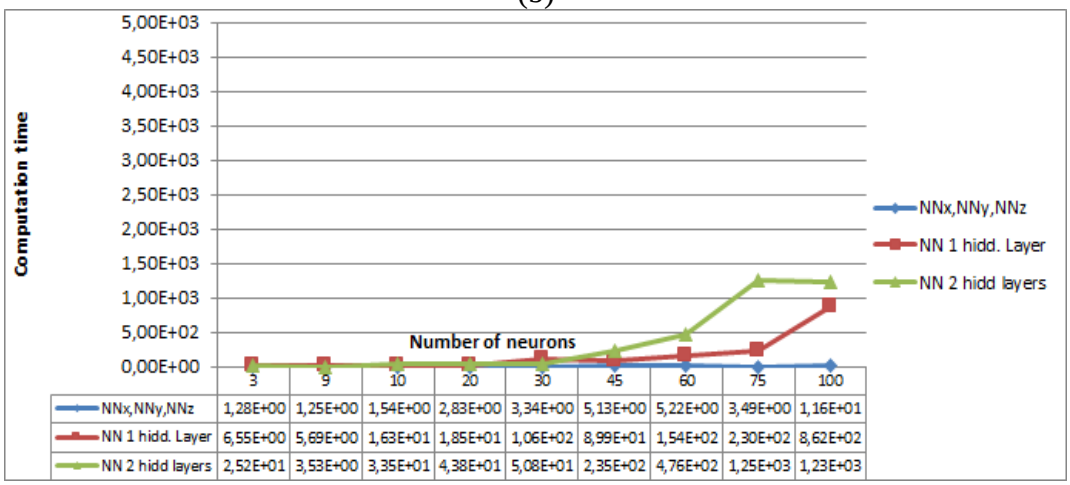
	3	9	10	20	30	45	60	75	100
NNx,NNy,NNz	2,10E-05	1,80E-05	1,86E-05	1,80E-05	1,81E-05	1,69E-05	1,91E-05	1,91E-05	1,72E-05
NN 1 hidd. layer	2,14E-05	2,10E-05	1,92E-05	1,76E-05	1,54E-05	1,69E-05	1,71E-05	1,50E-05	1,59E-05
NN 2 hidd. Layers	2,530E-05	2,196E-05	2,044E-05	1,614E-05	1,501E-05	1,685E-05	1,529E-05	1,492E-05	1,466E-05

(a)



	3	9	10	20	30	45	60	75	100
NNx,NNy,NNz	2,12E-05	1,89E-05	1,94E-05	1,83E-05	1,80E-05	1,82E-05	1,85E-05	1,79E-05	1,83E-05
NN 1 hidd. layer	2,21E-05	2,13E-05	1,90E-05	1,83E-05	1,69E-05	1,63E-05	1,63E-05	1,62E-05	1,64E-05
NN 2 hidd. Layers	2,568E-05	2,177E-05	1,998E-05	1,731E-05	1,588E-05	1,539E-05	1,563E-05	1,522E-05	1,495E-05

(b)



	3	9	10	20	30	45	60	75	100
NNx,NNy,NNz	1,28E+00	1,25E+00	1,54E+00	2,83E+00	3,34E+00	5,13E+00	5,22E+00	3,49E+00	1,16E+01
NN 1 hidd. Layer	6,55E+00	5,69E+00	1,63E+01	1,85E+01	1,06E+02	8,99E+01	1,54E+02	2,30E+02	8,62E+02
NN 2 hidd layers	2,52E+01	3,53E+00	3,35E+01	4,38E+01	5,08E+01	2,35E+02	4,76E+02	1,25E+03	1,23E+03

(c)

Figure 4.7. (a) Validation errors obtained using three kinds of neural networks; (b) testing errors obtained using three kinds of neural networks; and (c) evolution of training time.

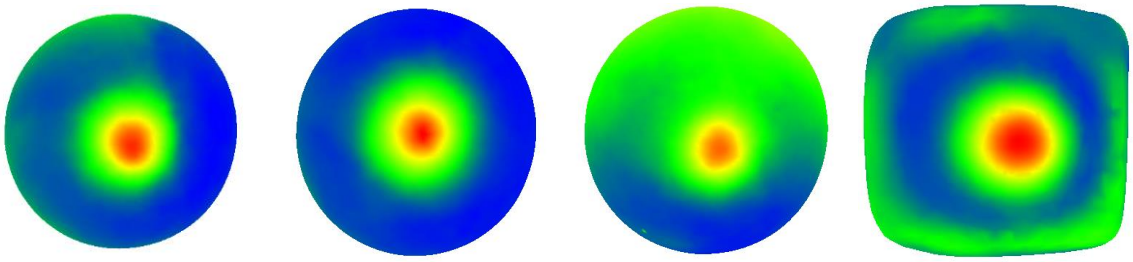
4.4. Prediction Results

This section presents experimental results for the proposed method for predicting deformations on 3D objects using different neural network architectures. It is aligned with objective 1 (Chapter 1, section 1.1) and contribution 1 (Chapter 1, section 1.3).

4.4.1. Prediction Using Three Neural Network per Each Cluster

Figure 4.8 shows a few examples of predicted deformed instances of objects obtained with the proposed method based on three neural networks, one for each cluster (as detailed in section 3.3.4, Chapter 3). The objects are shown such that the deformation is situated along the *y* axis, and the *x* axis points towards the left side of the ball, while the *z*

axis points towards the top of the image (as in Figure 3.4 a). Studying Figure 4.8 a that shows, for example, the difference between the ball model for $F_x = 3$, $F_y = 17$, and $F_z = -226$ and the predicted ball model for $F_x = 2$, $F_y = 14$, and $F_z = -226$, one can notice that there is a difference of force of 1 N along the x axis. Again, CloudCompare is used to encode in color the differences—blue representing a perfect match and the error increases from green, to yellow, orange, and red. The difference along the x axis is visible in the figure, in green, on the right side of the model. As one expects, the network was able to predict this movement of the object's surface along the *x* axis.



(a) (b) (c) (d)

Figure 4.8. The color-coded difference between: (a) the ball model for $F_x = 3$, $F_y = 17$, $F_z = -226$ and the predicted ball model for $F_x = 2$, $F_y = 14$, $F_z = -226$, (b) the ball model for $F_x = -35$, $F_y = 13$, $F_z = -252$ and the predicted ball model for: $F_x = -35$, $F_y = 11$, $F_z = -252$, (c) the ball model for $F_x = -10$, $F_y = 6$, $F_z = -204$ and the predicted ball model: $F_x = -10$, $F_y = 5$, $F_z = -200$; and (d) the cube model and the predicted cube model.

Additionally, there is a difference between the forces applied on the *y* axis that is larger than the one over the *x* axis (i.e., 3 N). This difference is visible in green, red, and yellow around the deformation zone, as expected. For the case in Figure 4.8 b, the force difference is mainly along the *y* axis and is reflected by differences in the deformation zone, as one might expect. A certain error appears around the sides of the object, as reflected by the green-bluish patches. The last example for the ball is for a force that affects the *z* and *y* directions, and it is again correctly predicted. Finally, an example is shown for the estimation of the cube for a force varying with 4 N the along *y* axis. The difference shown in red, yellow, and green is mainly concentrated around this axis as well. Table 4.7 summarizes the difference between some of the forces used for training

and those used for prediction. It is important to state that the force magnitudes shown in the table are the raw forces measured using the force-torque sensor, where the weight of the probing impacted the measurements.

Table 4.7. Forces used for training and prediction.

Real Measurements Forces used for Training $F_x/ F_y/ F_z$	**Forces used for Prediction** $F_x/ F_y/ F_z$	**Difference over *x*, *y* and *z* axes**
1/1/1	1/3/1	2 N over y axis
1/-1/1	1/-1/2	1 N over z axis
1/1/-1	2/1/-1	1 N over x axis
3/17/44	4/17/46	1 N over x axis, 2 N over z axis
2/17/-48	3/17/-49	1 N over x axis, 1 N over z axis
3/17/-226	2/14/-226	1 N over x axis, 3 N over y axis

To quantify the errors obtained for neural network training and prediction, we computed the Metro [36] and the perceptual [37] errors over the object surface and in the deformed area. Table 4.8 illustrates the values of the prediction errors when the networks are tested over previously unseen data, for 26 test cases (10 for the ball, 8 for the cube and 8 for the sponge), including those illustrated in Figure 4.8.

The overall perceptual similarity obtained is on average 76.17% for the ball, 92.02% for the cube, 93.54% for the sponge and 87.24% for the three objects together. Over the deformed area of the object, we obtained an average of perceptual similarity of 90.64% for the ball, 99.21% for the cube, 99.52% for the sponge and 96.46% for the three objects together.

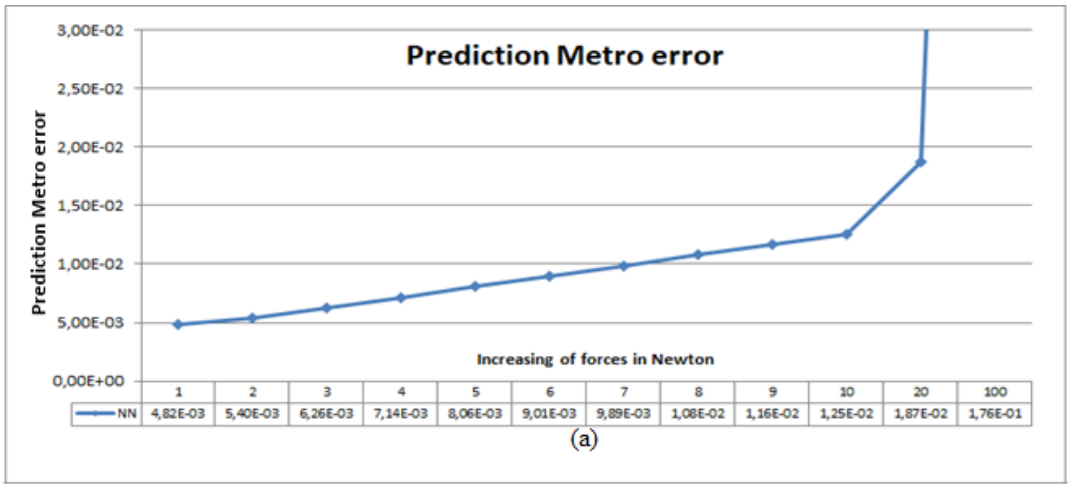
Table 4.8. Error measures for objects predicted.

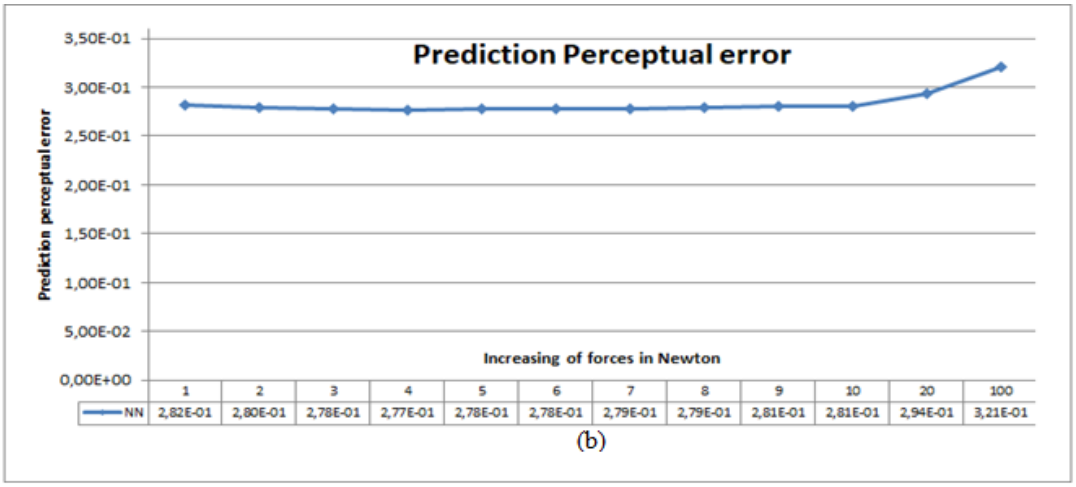
Object Name	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**
	Max/Mean/Rms		**Max/Mean/Rms**	
Ball	20.08/3.02/4.11	0.2383 (76.17%)	67.31/0.84/1.6	0.0936 (90.64%)
Cube	5.48/0.18/0.71	0.0798 (92.02%)	42.89/0.23/1.89	0.0070 (99.21%)
Sponge	2.65/0.65/0.83	0.0646 (93.54%)	2.71/0.0015/0.092	0.0053 (99.52%)
Average	**9.40/1.28/1.88**	**0.1276 (87.24%)**	**37.63/0.36/1.19**	**0.0353 (96.46%)**

It is important to notice that the differences in the force magnitude are not significant in the test scenarios with respect to the real measurements. The networks are only able to accurately predict within a limited region around the measured areas and for forces that do not differ significantly in magnitude and angle (i.e., about 4–5 N in force magnitude and 2–3° in angle), as it will be also shown in the next section for the case of one single network. This is an important aspect that needs to be taken into consideration when designing solutions based on the training of neural networks for predicting deformable object behavior, and any data-driven solution in general. A large number of real data measurements are required to ensure a more accurate prediction.

4.4.2. Prediction Using Neural Network with Two Hidden Layers

To test the prediction method, as described in section 4.3, we also used a neural network architecture with two hidden layers (i.e. the one with 82 neurons in the first layer and 18 in the second layer, being identified as the best choice). In this section, we present results for the ball model, after training it on the dataset described in section 4.3. After training the neural network, several deformed instances of the ball were predicted by increasing the force magnitude that was used in training. As such, we considered normal forces applied at a fixed location on the top of the ball (as the instances are aligned) and with increased force magnitudes between 1 N and 100 N.





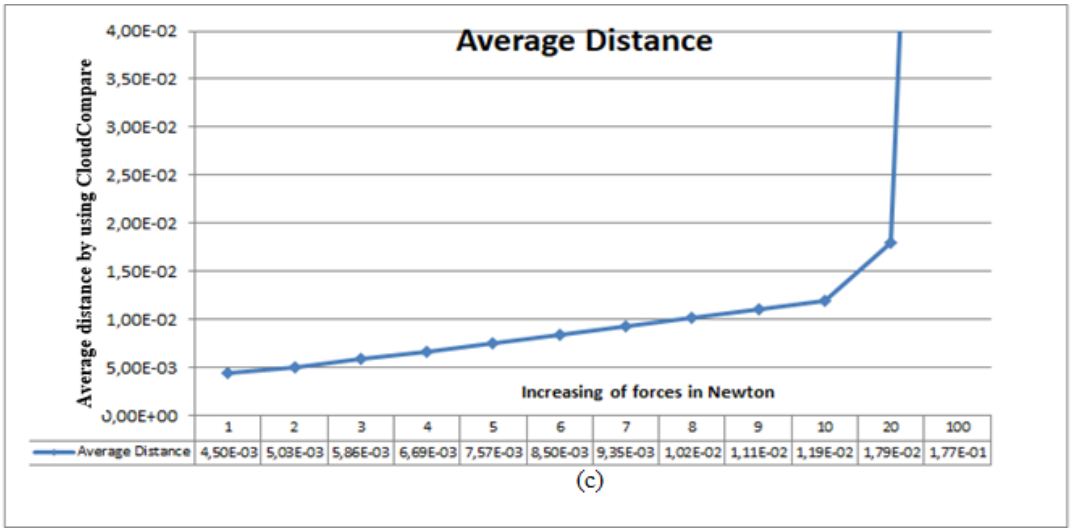


Figure 4.9. (a) Prediction Metro error for the ball; (b) perceptual error for the ball; and (c) average distance obtained by using CloudCompare.

Figures 4.9 a and b present the average of Metro and perceptual errors, while Figure 4.9 c illustrates the average distance computed using CloudCompare achieved during experimentation. Figures 4.10 a to f present the points clouds and the corresponding meshes of the ball predicted for various force magnitudes (1 N, 5 N and 100 N).

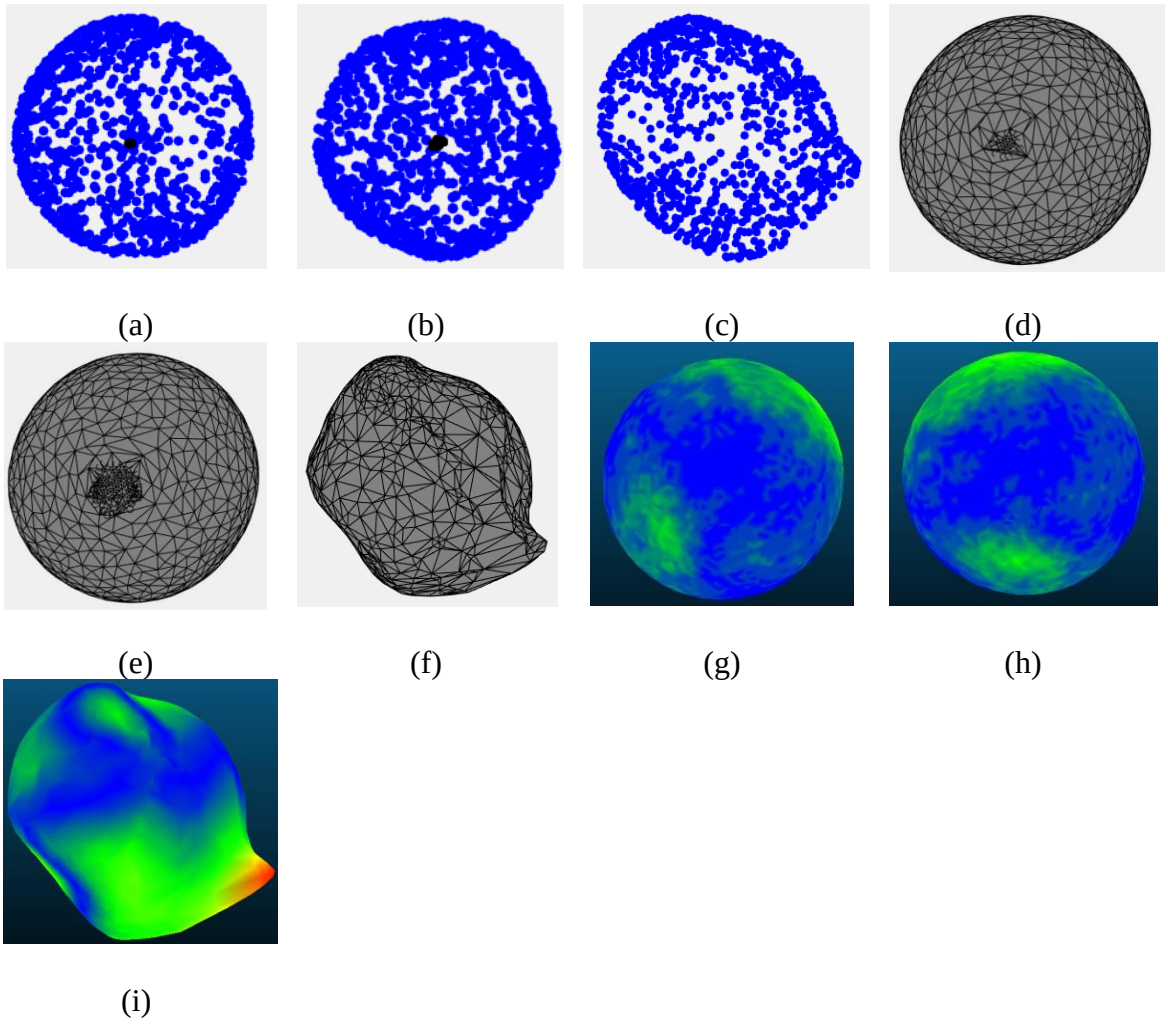


Figure 4.10. Ball point clouds predicted with a force of: (a) 1 N; (b) 5 N; (c) 100 N; Ball mesh predicted with a force of: (d) 1 N; (e) 5 N; (f) 100 N; Ball color-coded by CloudCompare: (g) 1 N; (h) 5 N; (i) 100 N.

Figures 4.10 g, h and i show the color-coded difference between the predicted ball and the one deformed with the same force (and that was not used for training). CloudCompare encodes in color the differences between two object meshes, with blue representing a perfect match and the error increases from green, to yellow and red. One can notice that the Metro error and the average distance obtained increase with an increase in the

magnitude force and that the shape of the predicted deformed ball moves away from the overall shape of the original ball (becomes more and more deformed). This is obvious when we observe the cloud points and the color-coded representation of the predicted balls for forces greater than 5 N. This problem is due to two factors: (1) the forces used for training the network are not in the same range as the ground truth measurements using the force-torque sensor (in which case the weight of the probing tip impacted the magnitude of the force applied); and (2) the number of instances used to train the neural network is not sufficient for training (only 10 deformed instances) and thus to achieve better results than those obtained. This aligns with the observations in the previous section, regarding the fact that the proposed method is accurate only within 4-5N, and that additional data is required to improve the prediction capability. The next section presents results when additional training samples are used.

4.4.3. Generation of Additional Data for Neural Network Training

In an attempt to increase the performance of the prediction capabilities, we increased the number of deformed objects used to train the neural network. In particular, we increased the data available for training 5 times by using 50 deformed instances of the objects with force magnitudes between 1 N and 50 N, applied in the normal direction (i.e. perpendicular to the local surface) at the top of the object (as instances are aligned). Due to our inability to perform additional real measurements in the context of the pandemic, the solution chosen was to make use of a software solution, namely the Ansys academic software simulator [197, 198] (for more details see Annexes D, E, F and G). Ansys is a 3D finite element mesh generator providing a complete modeling method with parametric inputs and advanced visualization capabilities. It is divided into four modules: materials choice, geometry, structure and modeling, method and results. Using this software, one can design and deform 3D objects, and exert different forces on the object, such as nodal forces, nodal pressure, body forces or body pressure. To create and deform 3D objects, one starts with the choice of materials (see Annex E), then creating the 3D object (Annex F), meshing the object, then exerting forces on the object, followed by building and exporting the deformed mesh (Annex G).

To be able to create similar objects to the real ones we used, several steps were required. For example, to create a ball similar to the real one, in a first step, the geometry of the non-deformed ball was exported to the Ansys software, and several trials were performed to identify a material (i.e. rubber) that performed similarly to the ball used in the ground truth. As explained in Annex B, each material has specific parameters and the matrices of stiffness, damping and mass are defined by default for each material. It is important to state that our method does not imply the estimation of material parameters, and this estimation is only required to create additional data, in an attempt to demonstrate that the lack of data is the main reason of the limitation of our method in the previous sections. After choosing the material, we linked it with the geometry of the ball, and then meshed the geometry to obtain a deformed mesh. In a second step, the type and values of the forces to be applied on the object are to be defined to obtain a deformation similar to the one obtained using the force torque sensor on the real ball. As such, we had to identify an appropriate choice of forces to correspond to the real force-torque measurements using the Ansys software. To achieve this, we deformed the mesh representing the ball by Ansys while applying different magnitudes of forces and angles as nodal forces. Among the various options available in Ansys to apply forces such as nodal force, body force on the face, body force on the edge, body force on the vertices and pressure force on the geometry of the object, the nodal force was the one that led to the best results (i.e. more similarity with the real deformed ball). Forces were applied at the same interaction points used in the ground truth to deform each instance of the ball. Therefore, using Ansys, we deformed the ball mesh using the same force measured by the force torque sensor and compared the deformed instances of the ball, we compared them with the real instances of the ball deformed in the ground truth.

To evaluate the performance of our prediction method for various materials, we also created three additional objects (Figure 4.11) of different materials, and for each of them, we created 50 deformation instances. These objects are a flexible polyethylene foam cylinder, a flexible polyurethane foam cube and a flexible rubber and foam toy table (20% rubber and 80% foam). Table 4.9 presents the different force magnitudes used during experimentation. All forces exerted are between 1 N and 50 N. These forces were exerted in the normal direction on the top of each object. Each deformed instance of the

rubber ball has 5670 vertices and 11336 faces. From each one, as it was stated in the previous sections, we used 1681 vertices (after simplification at 40%, clustering, sampling and neural gas fitting). This led to a dataset of 50 deformed instances, and a total of 84050 vertices. From these, we used the same percentages as in the previous section, namely 50% (42025 vertices) for training the neural network, 5% (4203 vertices) for validation and 45% for testing (37822 vertices).

Table 4.9. Forces exerted over x, y and z axis in the additional dataset.

$\mathbf{F_x}$	$\mathbf{F_y}$	$\mathbf{F_z}$	$\mathbf{F_x}$	$\mathbf{F_y}$	$\mathbf{F_z}$	$\mathbf{F_x}$	$\mathbf{F_y}$	$\mathbf{F_z}$
1	18.1	45.7	16.6	49	44.1	34.3	26.1	34.7
2.10	23.5	20.9	17.1	28.8	44.8	36	29.7	35.7
2.30	42.5	16.3	17.4	40	33.7	36.1	43.6	44.2
2.40	36.4	14.2	17.7	35.8	40.5	36.8	1.30	41.3
2.70	46.9	30.5	18.1	15.4	12.3	39.7	22.3	19.5
3.50	10.3	33.7	20	27.5	35.8	40	3.60	21.9
4.20	39.4	27.9	20.2	1.10	22.5	40.6	23.8	44.1
4.70	44.2	20.4	22.5	41.9	24.2	43.9	49.2	46.7
6.10	28	10.3	24.5	27.2	33.3	43.9	46.7	36
6.10	44.8	33.2	25.2	14.3	20.3	44.2	29.9	6.10
7.60	14.3	14.2	27.2	32.3	24.9	44.7	18.6	41.7
9.50	13.5	49.4	29.1	33.4	1	46.1	32.7	21.9
9.80	18.6	22.8	29.3	25.2	19.4	46.5	47.2	13.8
11.6	6.70	45	31	6.90	39.2	48.2	21.7	37.8
12.9	47	38.1	32.6	33.4	7.40	48.6	28	48.6
13.4	15	45	33.4	37.5	43.2	48.7	39	29.9
15.2	41.5	28.7	33.8	32.3	22.7			

The same percentages of points for training, validation, and testing (50%, 5% and 45%) were used for all objects. For the cylinder, the total number of points is 47350 vertices (23675 vertices for training, 2368 vertices for validation and 21307 vertices for testing);

for the cube, 58400 vertices (29200 vertices for training, 2920 vertices for validation and 26280 vertices for testing) and for the table 26450 vertices are used (13225 vertices for training, 1323 vertices for validation and 11902 vertices for testing). Before proceeding with the prediction, we studied the various types of neural network architectures (i.e. three networks (one for each coordinate), a single network with a single hidden layer and a single network with two hidden layers) and repeated the same tests as in section 4.3 but using the newly generated data.

For each type of network, we increased the number of neurons in the hidden layers (between 3 and 100) to identify an appropriate size. Then, we monitored the validation and testing errors obtained by the networks. After performing these tests, we reached to the conclusion that the best type of neural network (the neural network with the smallest test error) is the one with two hidden layers with 82 neurons in the first hidden layer and 28 neurons in the second layer. The following sections present prediction results for each of the objects using the newly created data and the chosen architecture.

4.4.4. Prediction of the Ball Using Additional Data

To test the prediction capabilities of the network for the ball, we trained the network with two hidden layers using 50 deformed ball instances. After training, we predicted various deformed instances of the ball by increasing the force magnitude between 1 N and 100 N. Figures 4.12 a, b and c show respectively the evolution of the Metro, perceptual errors and average distance error for the predicted balls obtained.

When comparing the average of errors computed (Figure 4.12) with those obtained in Section 3.4.2 (using 10 deformed instances), one can notice that the errors here are much smaller. Example with a force increased by 20 N the overall Metro, perceptual errors and the distance average are respectively $2.56e^{-3}$, $2.62e^{-1}$ and $2.03e^{-3}$, while using 10 deformed instances these errors are respectively $1.87e^{-2}$, $2.94e^{-1}$ and $1.19e^{-2}$.

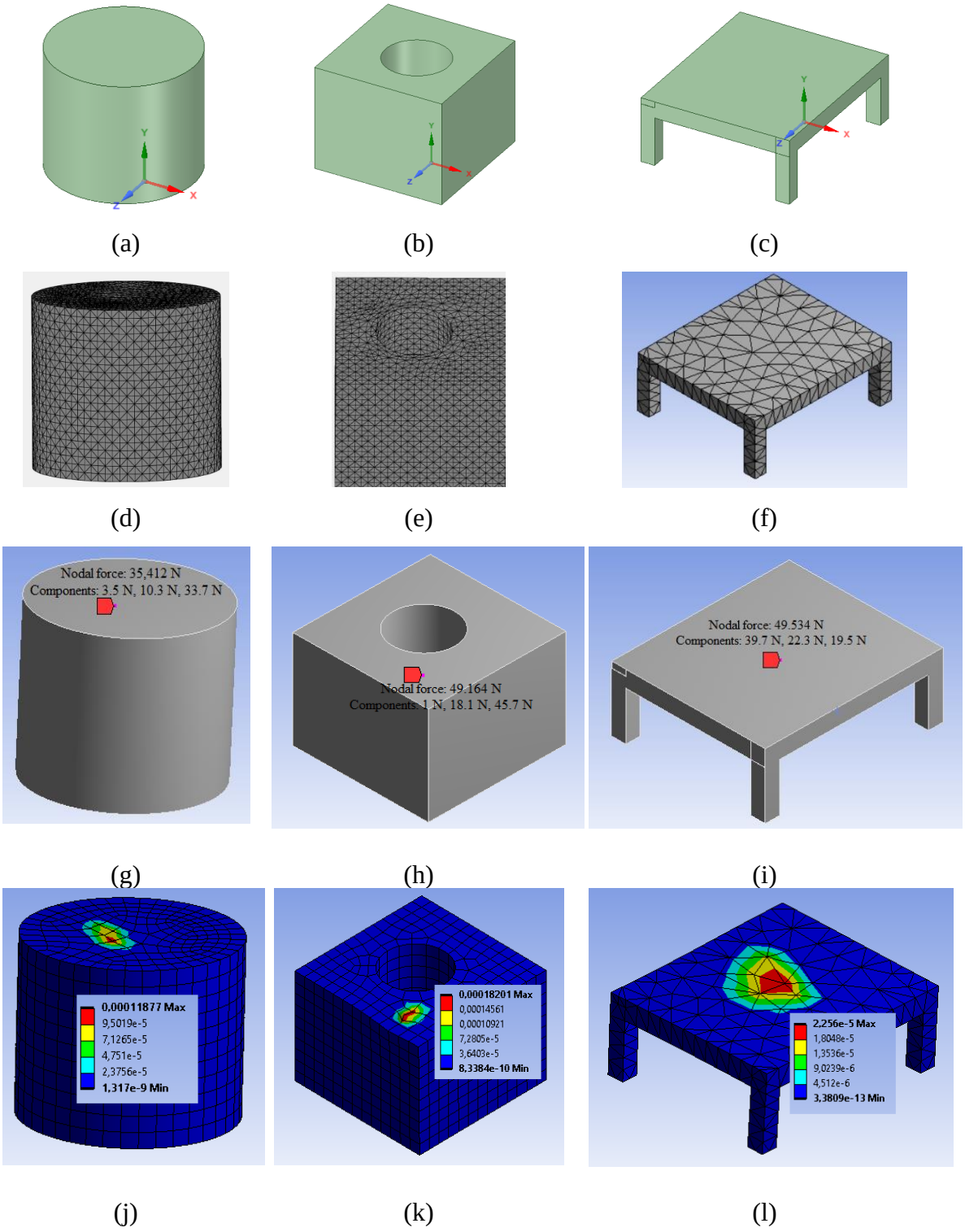


Figure 4.11. (a) Polyethylene foam cylinder; (b) polyurethane foam cube; (c) rubber and foam table; the associated mesh for (d) cylinder; (e) cube; and (f) table; sample forces exerted on: (g) cylinder; (h) cube and (i) table; and associated mesh deformation for: (j) cylinder; (k) cube and (l) table.

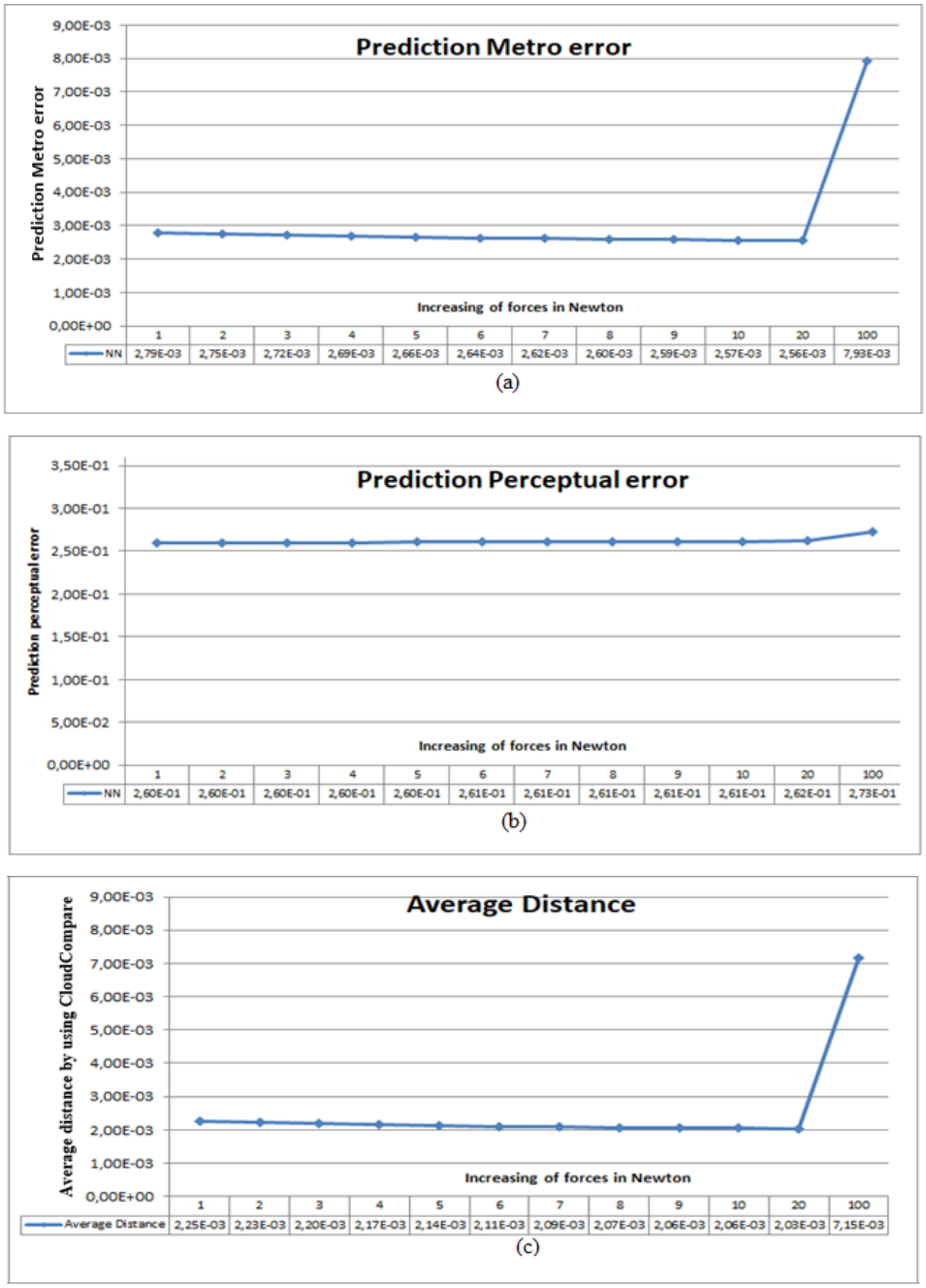


Figure 4.12. (a) Prediction Metro error for the ball; (b) prediction perceptual error; and (c) average distance obtained by using CloudCompare.

The level of similarity is increased and remains relatively stable up to 60 N (Figures 4.12 and 4.13) while by using 10 deformed instances the level of similarity is increased and remains relatively stable up to 20 N.

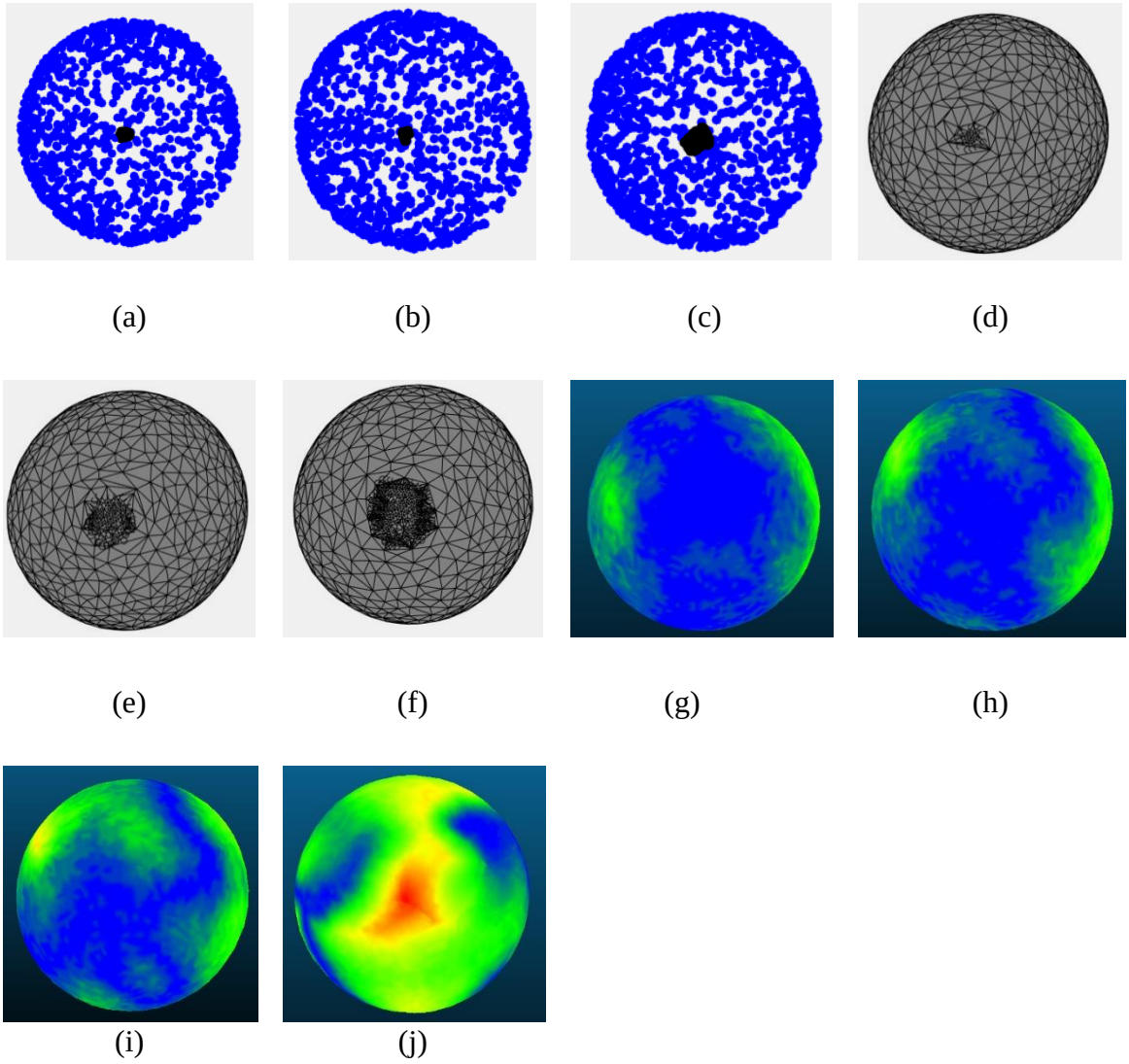


Figure 4.13. Ball point clouds predicted with a force of: (a) 1 N; (b) 5 N; (c) 60 N; Ball mesh predicted with a force of: (d) 1 N; (e) 5 N; (f) 60 N; Ball color-coded by CloudCompare for: (g) 1 N; (h) 5 N; (i) 60 N; Ball color-coded by CloudCompare displayed on the other side for: (j) 60 N.

In spite of the loss of similarity for higher magnitude forces, these results demonstrate that the addition of data helps, as expected, to improve the quality of the neural network model and thus its ability to generate better predictions.

4.4.5. Prediction of the Cylinder

Similar to the model of the ball, we trained and tested the same neural network architecture, using the same parameters (Table 4.9) and the same amount of training data, i.e. 50 deformed instances, for the cylinder.

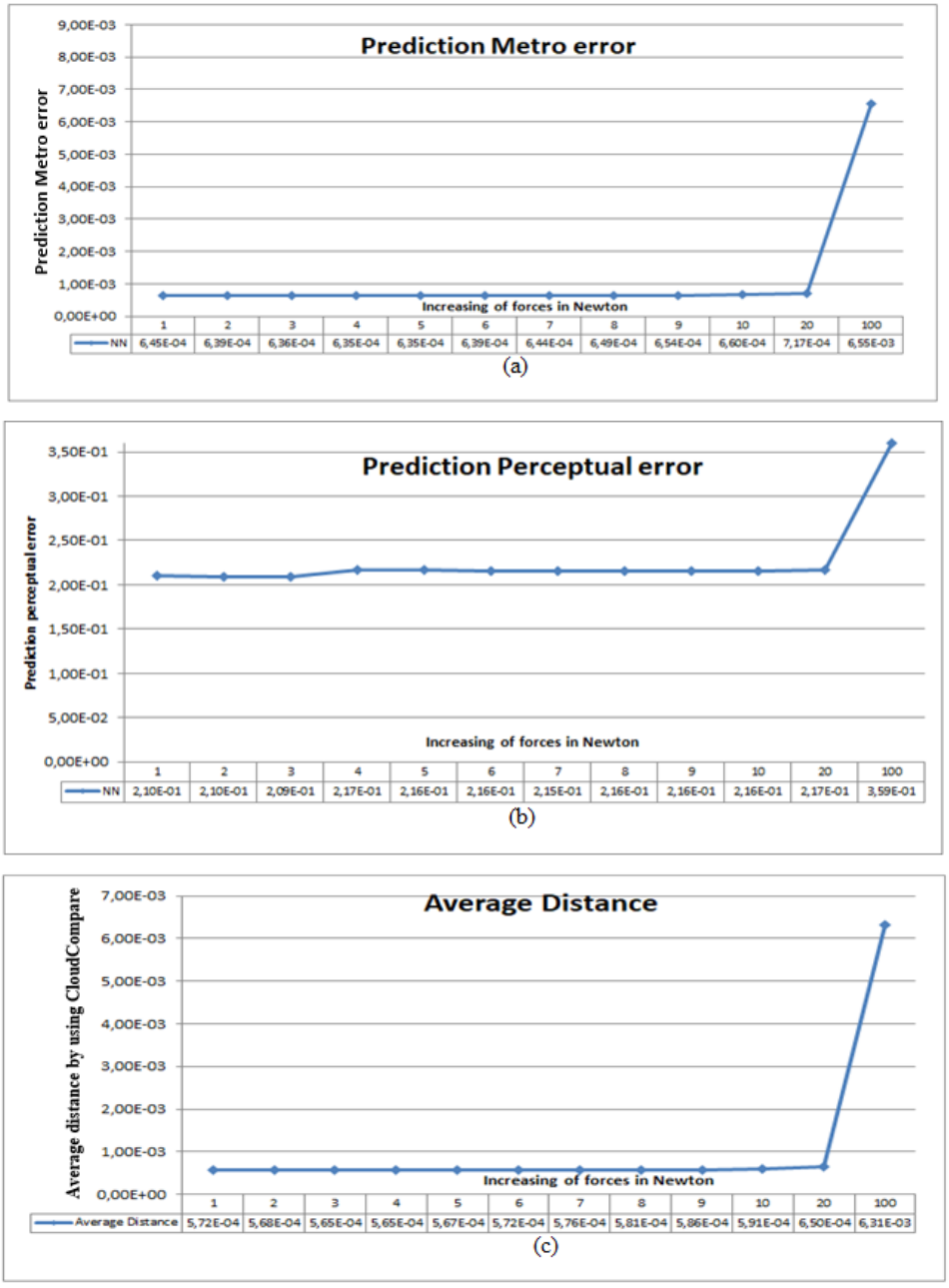


Figure 4.14. (a) Prediction Metro error for the cylinder; (b) prediction perceptual error for the cylinder; and (c) average distance obtained by using CloudCompare.

To predict various deformed instances of this object, we increased the force magnitude, as in the case of the ball (between 1 N and 100 N). The evolution of average of Metro errors, perceptual errors and the average distance for the predicted cylinders is shown in Figures 4.14 a, b and c respectively. Figures 4.15 a to e present the cylinder points cloud predicted for forces increased between 1 N and 30 N, while Figures 4.15 f and j show the predicted mesh for forces increased by 1 N and 30 N, respectively. Finally Figures 4.15 h and i show the color-coded differences between the predicted cylinder and the deformed cylinder, from two viewpoints, for the same force using CloudCompare.

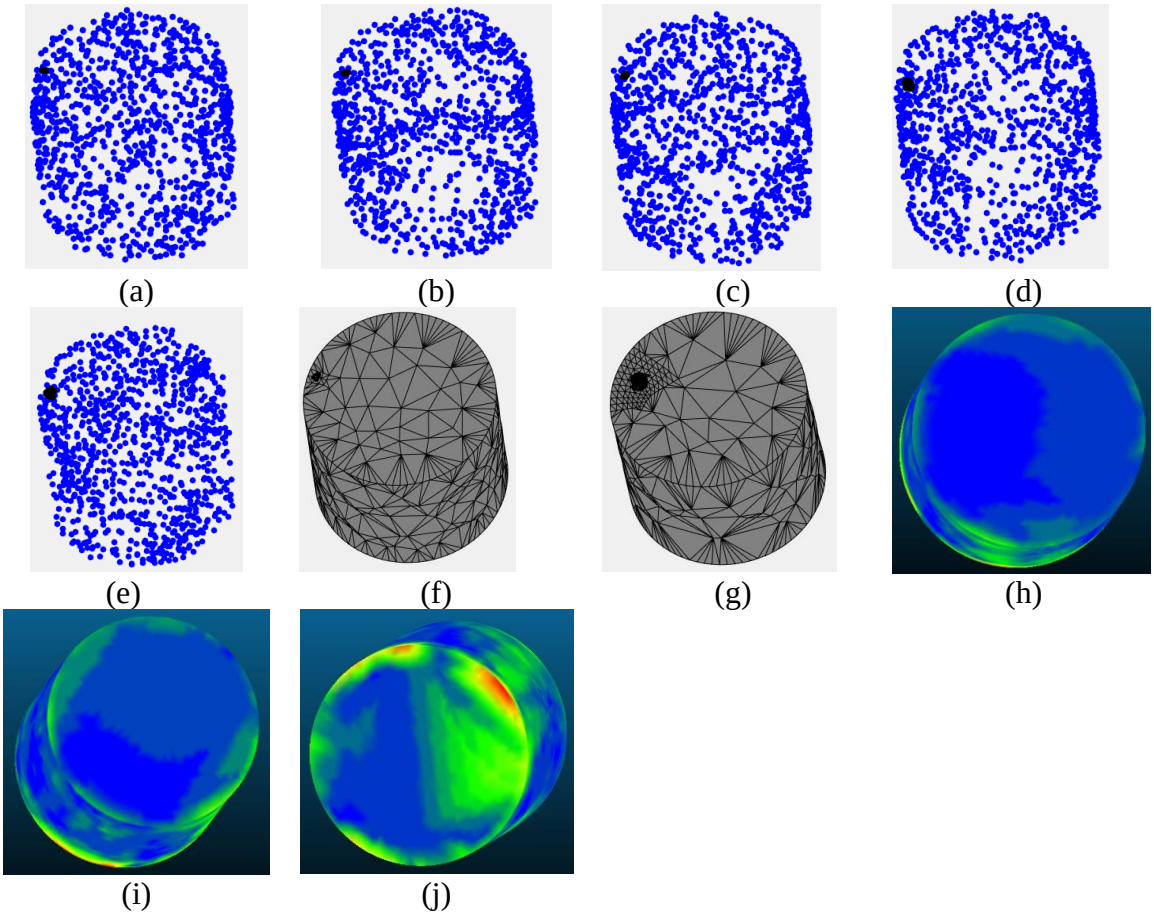
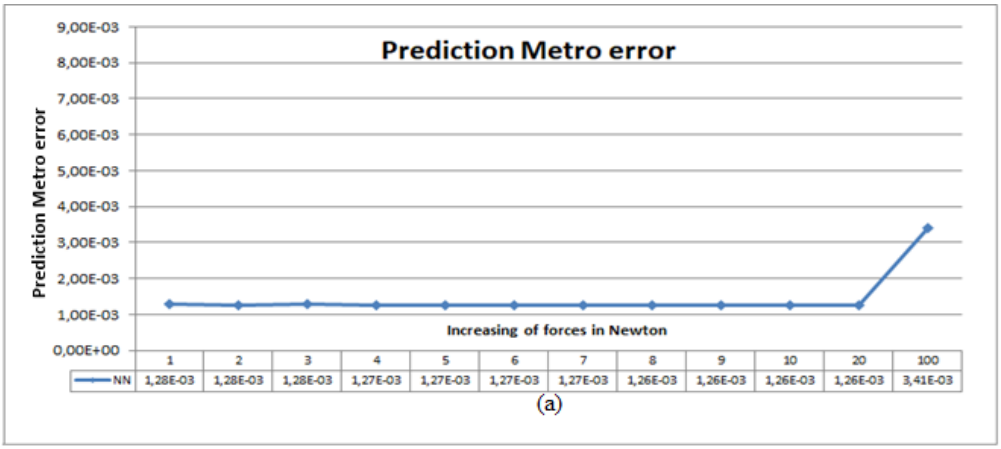


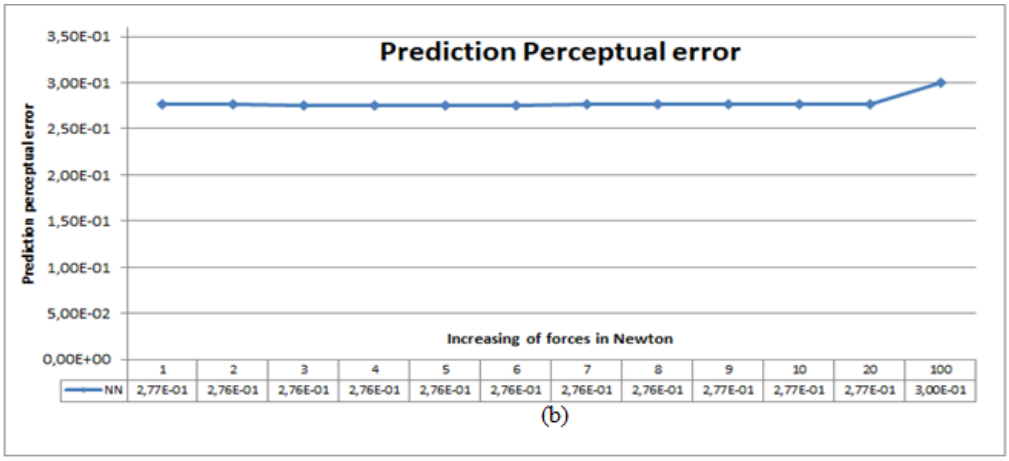
Figure 4.15. Cylinder point clouds predicted with a force increased of: (a) 1 N; (b) 5 N; (c) 10 N; (d) 20 N; (e) 30 N; mesh predicted with a force increased of (f) 1 N; (g) 30 N; Cylinder color-coded by CloudCompare for: (h) 1 N; (i) 30 N; Cylinder color-coded by CloudCompare displayed on the other side for: (j) 30 N.

4.4.6. Prediction of the Cube

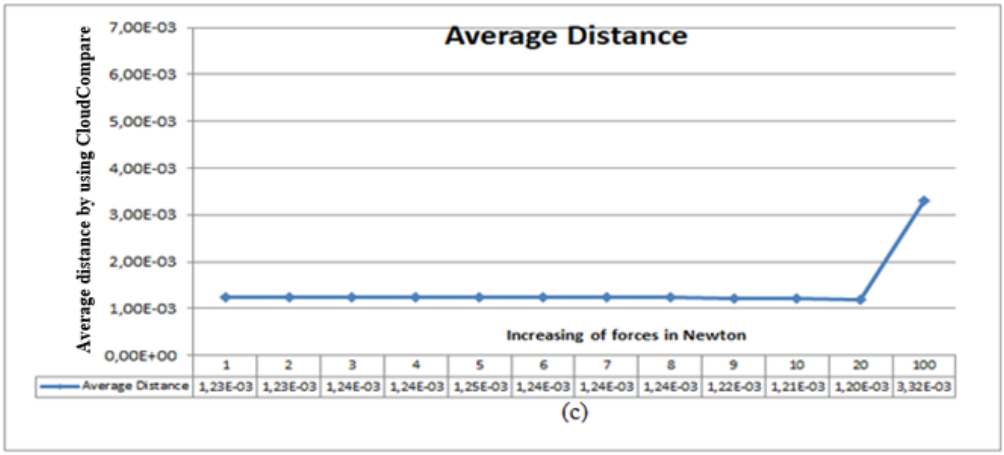
For the prediction of the deformations over the surface of the cube, we repeated the same process we used for the ball and the cylinder. Figures 4.16 a, b and c present respectively the evolution of the average of Metro errors, perceptual errors and the average distance for the predicted deformations for the cube.



(a)

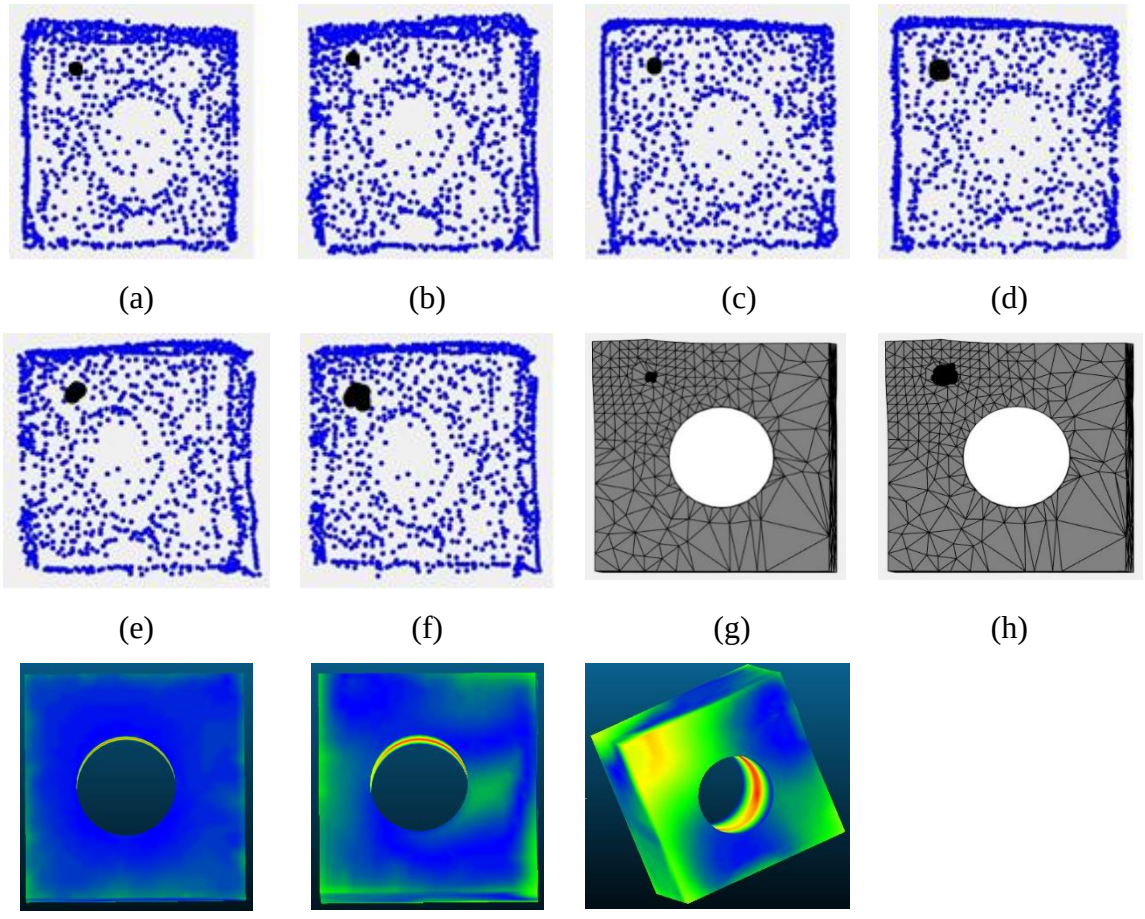


(b)



(c)

Figure 4.16. (a) Prediction Metro error for the cube; (b) prediction perceptual error for the cube; and (c) average distance obtained by using CloudCompare.



(a) (b) (c) (d)

(e) (f) (g) (h)

(i) (j) (k)

Figure 4.17. Cube point clouds predicted with a force increased of: (a) 1 N; (b) 5 N; (c) 10 N; (d) 20 N; (e) 30 N; (f) 40 N; mesh predicted with a force increased of (g) 1 N; (h) 40 N; Cube color-coded by CloudCompare for: (i) 1 N; (j) 40 N; Cube color-coded by CloudCompare displayed on the other side for: (k) 40 N.

Studying Figures 4.17 that illustrates results for the cube predicted for forces increased between 1 N and 40 N, one can notice that in presence of sufficient data the network is able to capture implicitly the shape deformation generated by various magnitude of force. Figures 4.17 i and j show the color-coded objects built using CloudCompare that result by comparing the predicted cube with the deformed cube obtained with the same force.

4.4.7. Prediction of the Table

Finally, for the case of the table, Figures 4.18 a, b and c show respectively the evolution of the average Metro, perceptual error and average distance errors.

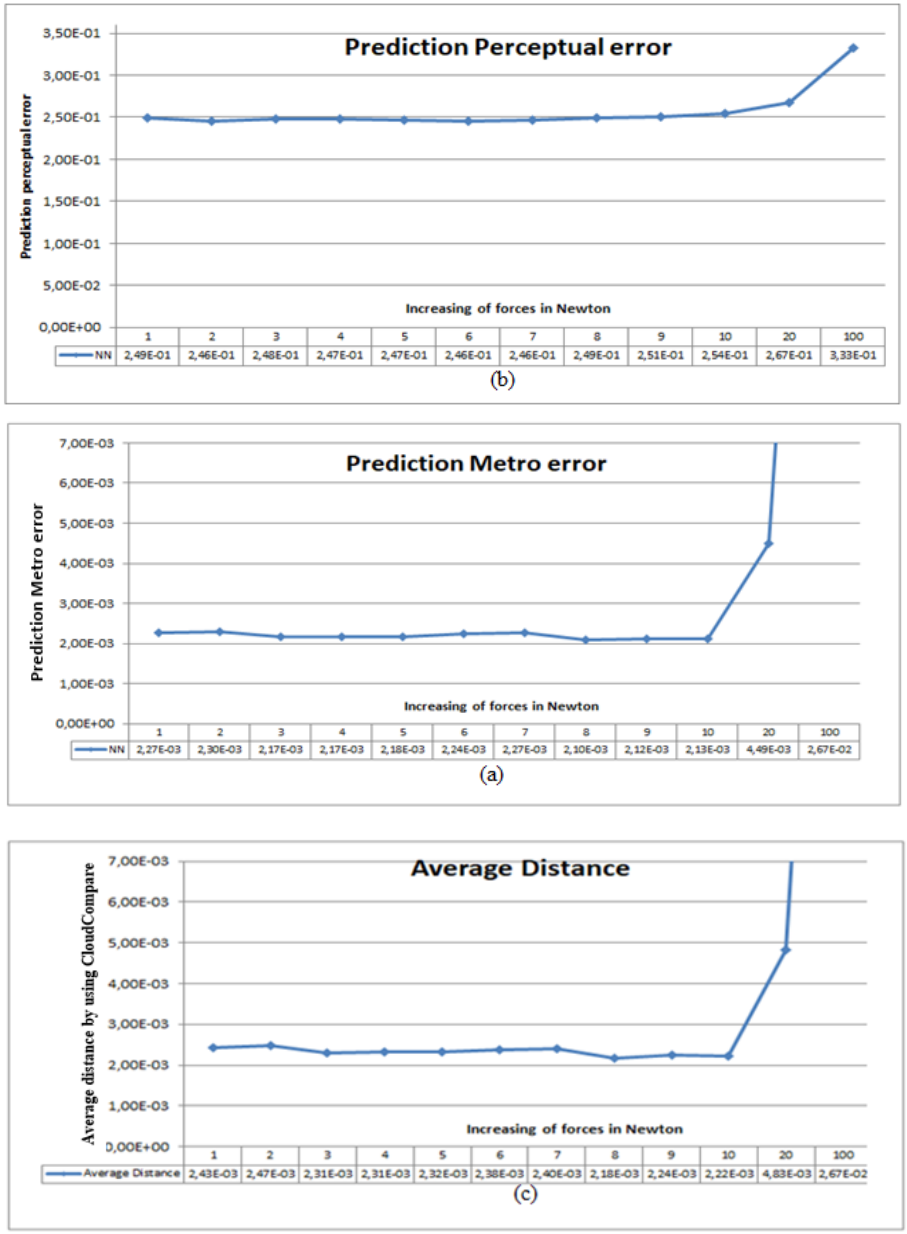


Figure 4.18. (a) Prediction Metro error for the table; (b) perceptual error for the table; and (c) average distance obtained using CloudCompare.

Figures 4.19 a to d show the point clouds predicted for forces increased between 1 N and 20 N, while Figures 4.19 e and f show the predicted mesh and the color-coded objects

obtained using CloudCompare for forces of 1 N and 20 N. The conclusions are similar as well to the ones achieved for the previous objects.

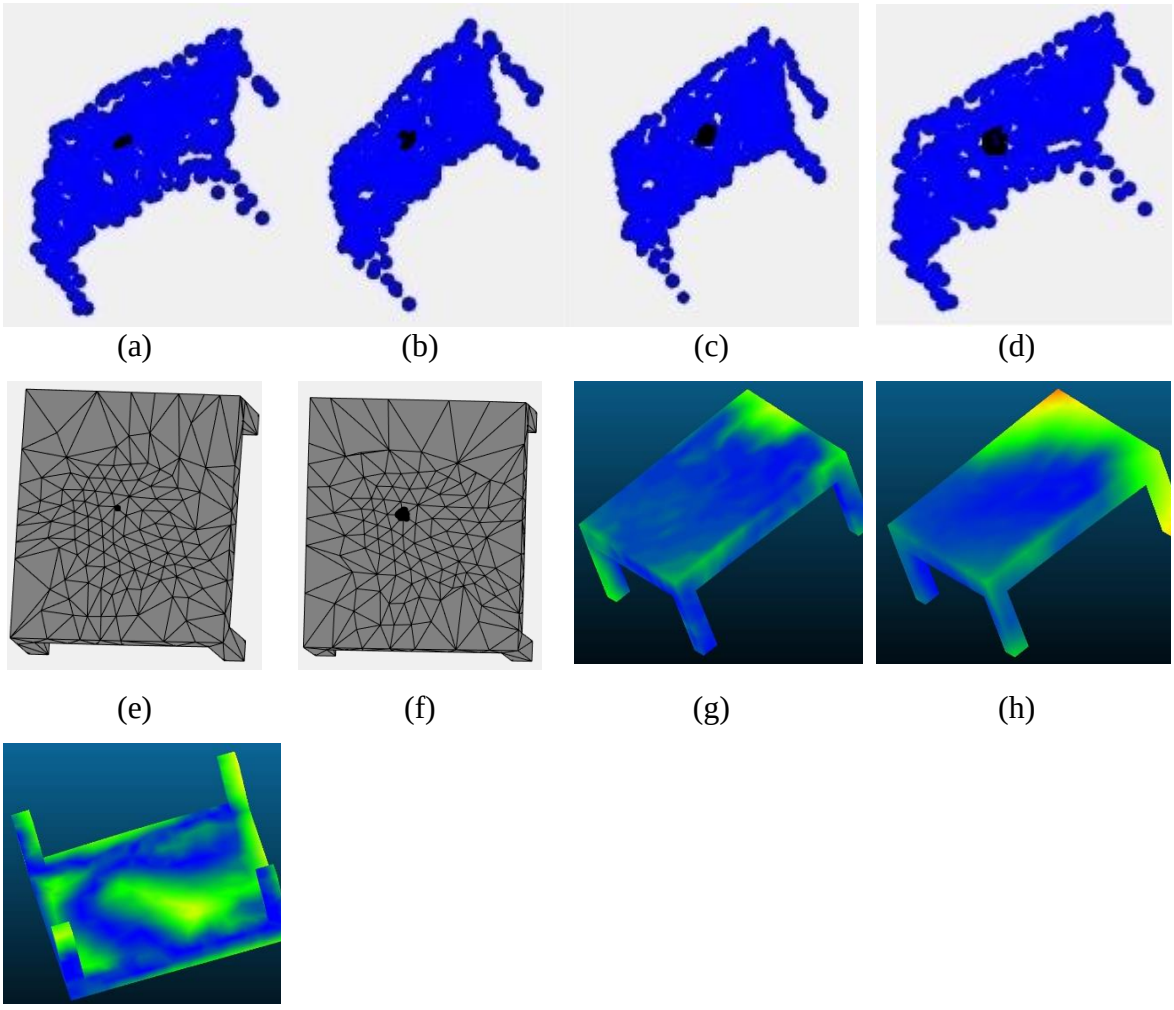


Figure 4.19. Table point clouds predicted with a force increased of: (a) 1 N; (b) 5 N; (c) 10 N; (d) 20 N; mesh predicted with a force increased of (e) 1 N; (f) 20 N; Table color-coded by CloudCompare for: (g) 1 N; (h) 20 N; Cube color-coded by CloudCompare displayed on the other side for: (i) 20 N.

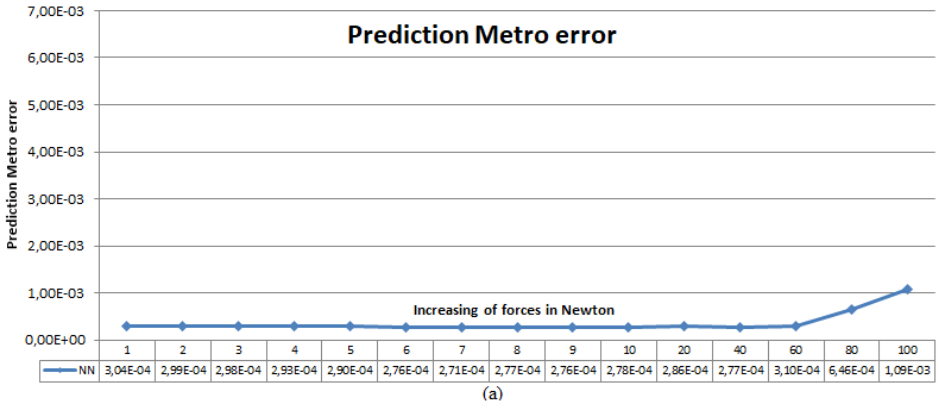
4.4.8. Prediction of Elasto-Plastic and Rigid Objects

To further expand the testing of the proposed method, we also tested it on an elasto-plastic (i.e. a cylinder of 50% foam and 50% ABS Plastic) object, a semi-rigid object (i.e. a cylinder of tin, which remains relatively rigid when a small force is applied) and a rigid

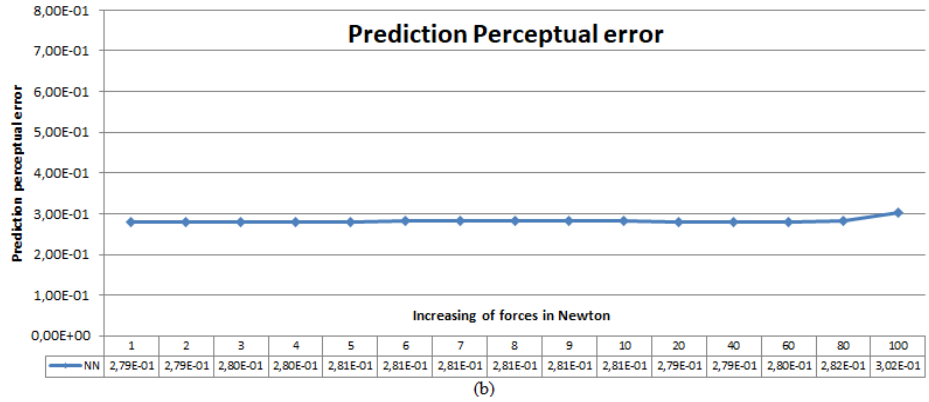
object (i.e. a cylinder of wood). This goes beyond the initial intention of this work to model deformable objects and demonstrate that the proposed method can be used without changes to capture objects that are not fully deformable (such as elasto-plastic and rigid). An elasto-plastic object is an object that has elastic and plastic behavior, while a rigid object has an unaltered body in general. While the tin cylinder is not rigid, it remains rigid for small forces, so we only tested it under this condition. The wood cylinder is rigid, and it cannot be deformed. For each cylinder, we used the same approach as for the soft cylinder; as such, we have created 50 deformed instances using different forces and applied the preprocessing steps of our method (simplification, stratified sampling and neural gas fitting). Then, we trained the same neural network architecture as in the previous sections (with two hidden layers) and predicted multiple instances of each object.

As for the previous objects, Figures 4.20 a to c illustrate respectively the evolution of the average Metro, perceptual error and the average distance errors for the predicted elasto-plastic cylinder with increased force magnitudes between 1 N and 100 N.

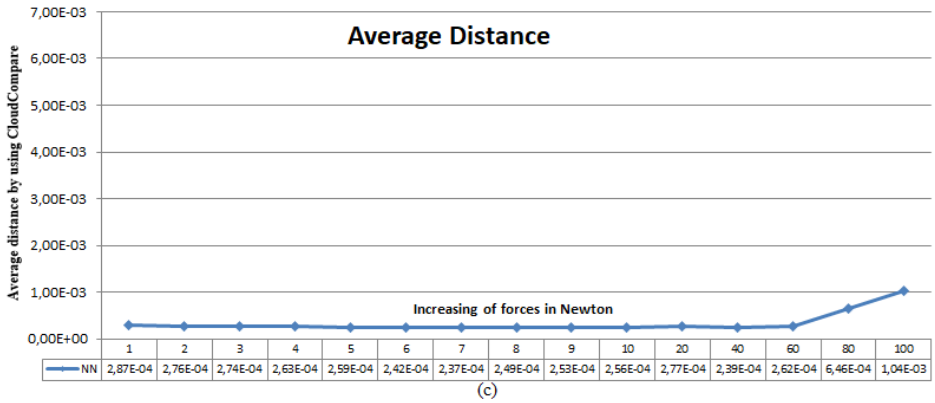
Figures 4.21 a illustrates the original non-deformed elasto-plastic cylinder and Figures b, c and d show the points cloud, the mesh and the color coded representation obtained by CloudCompare for a predicted elasto-plastic cylinder with a force of 20 N.



(a)

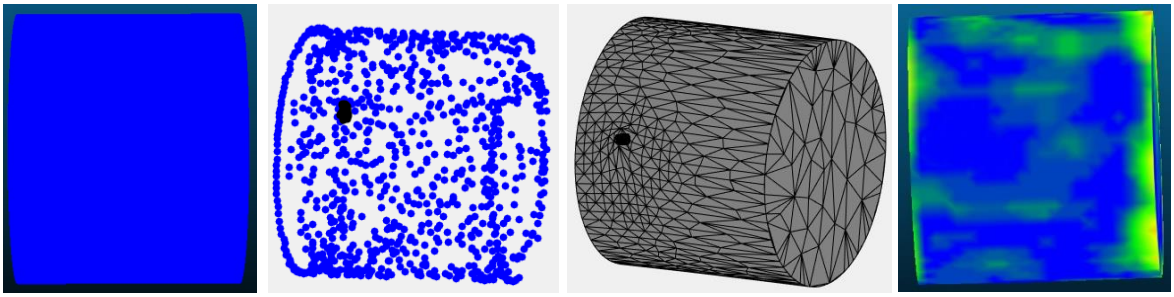


(b)



(c)

Figure 4.20. (a) Prediction Metro error for the elasto-plastic cylinder; (b) prediction perceptual error for the elasto-plastic cylinder; and (c) average distance obtained by using CloudCompare.



(a) (b) (c) (d)

Figure 4.21. (a) Non-deformed elasto-plastic cylinder; (b) points clouds predicted for a force of 20 N applied on the top right corner; (c) mesh predicted with a force 20 N; and (d) cylinder using CloudCompare mapping of errors between the modeled and predicted cylinder.

Figure 4.22 shows the average errors for the predicted rigid cylinder of tin, with the same forces used for the elasto-plastic cylinder. For the elasto-plastic and the tin cylinders the level of similarity is increased and remains stable up to 20 N (Figure 4.20 and Figure 4.22), while in the case of the rigid cylinder of wood, the level of similarity remains more than 90% because it is not deformable (Figure 4.24).

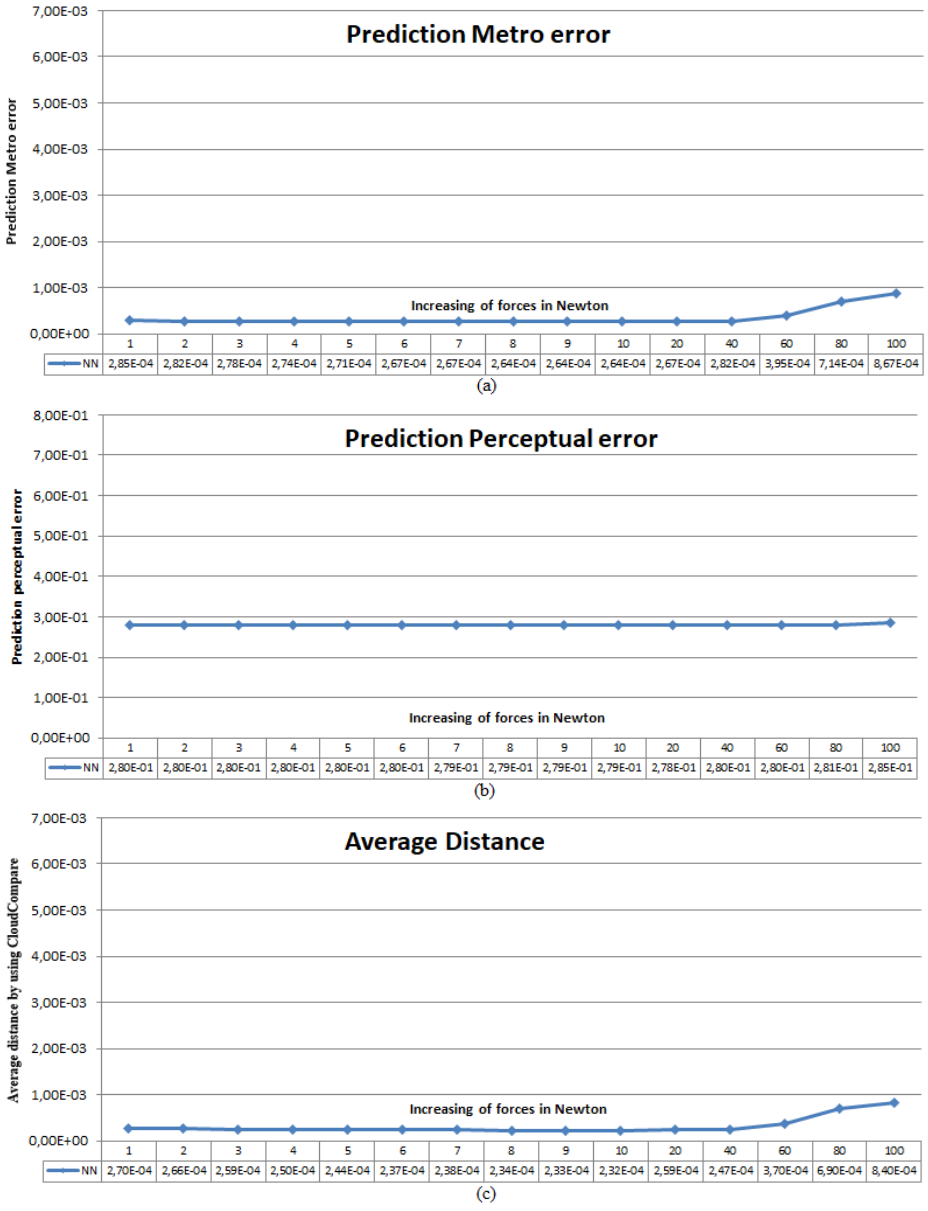
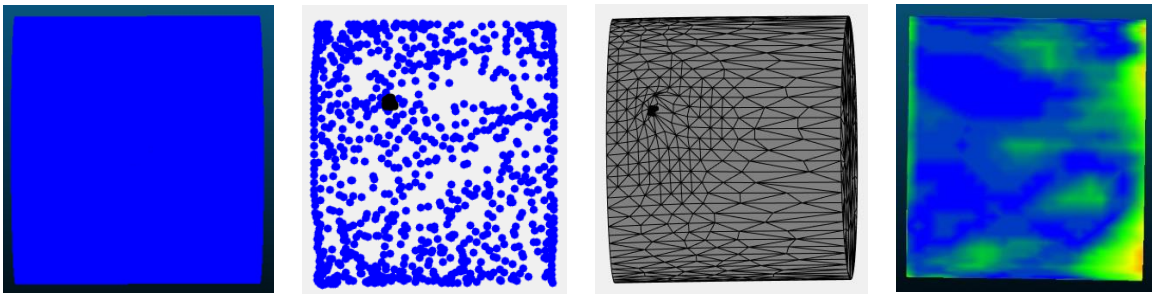


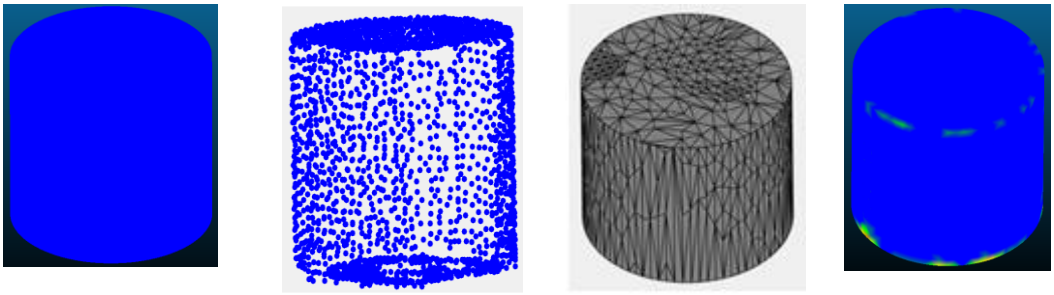
Figure 4.22. (a) Prediction Metro error for the tin cylinder; (b) prediction perceptual error for the tin cylinder; and (c) average distance obtained by using CloudCompare.



(a) (b) (c) (d)

Figure 4.23. (a) Tin cylinder; (b) points cloud predicted for a force of 20 N applied on the top right corner; (c) mesh predicted with a force 20 N; and (d) cylinder using CloudCompare.

Figure 4.23 a shows the original non-deformed tin cylinder and Figures b,c and d show the points cloud, the mesh and the color encoded difference between the reference object and a predicted rigid cylinder for a force of 20 N, respectively. In this case, the object deforms much less for the applied force, as one can notice in Figure 4.23d. The impact of the force is more local and impacts less the entire area of the object. In this case, the green areas are mostly due to prediction errors. Finally, Figure 4.24 a illustrates the results for the wood cylinder, including the point cloud (b), the selectively simplified mesh (c) and the color coded representation for a predicted wood cylinder for a force of 20 N (d), respectively.



(a) (b) (c) (d)

Figure 4.24. (a) Non-deformed rigid cylinder; (b) points cloud predicted for a force of 20 N applied on the cylinder; (c) mesh predicted with a force 20 N; and (d) cylinder encoding the difference between the predicted and reference model using CloudCompare.

The rigid cylinder is not deformable, and thus constitutes a special, extreme case for our prediction method. The computed Metro and perceptual errors are minimal in this case (converge to 0) and the similarity to the original non-deformable is more than 90%. One can notice that the proposed prediction method works for this extreme case without any adjustment. All these results demonstrate that the proposed method can deal successfully with various types of materials.

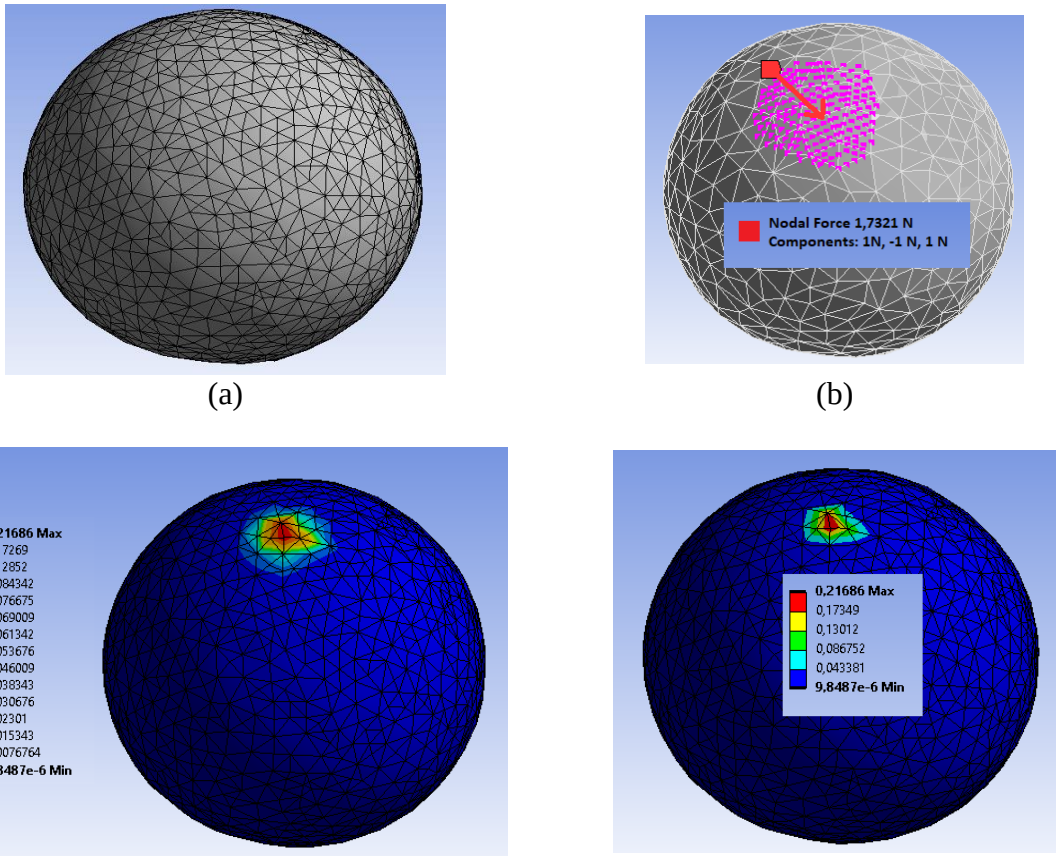
4.5. Comparison with FEM Method

To further evaluate our proposed modeling and predicton method, we compared it with a classical solution for the modeling of deformable objects, namely the FEM method. We chose FEM because it is one of the well-known methods and its use expands over many areas [9,10,11,12,17,125,126]. To perform the comparison, we compared objects predicted by our method with objects whose deformation was modeled through FEM based on the same interaction parameters. To obtain a quantitative measure, we computed the Metro errors, the perceptual errors as well as the average distance errors between the two objects. To deform the objects based on the FEM method, we used the Ansys academic software simulator which is a FEM generator.

Figure 4.25 shows an example of the steps used to import and deform the original non-deformed mesh of the ball measured in ground truth, with Figure 4.25 a showing the original non-deformed mesh and Figure 4.25 b the nodal force of 1,73 N (F_x=1, F_y=-1 and F_z=1) applied on the ball. The pink color represents the nodes to which the force was applied. It is important to clarify that this is for illustration purposes, as during our tests we focused on a single point of interaction (as detailed in section 4.1.4). Figure 4.25 c shows the different scales (15 clusters) of deformation after applying this force, while Figure 4.25 d shows the 5 different clusters of deformation after applying this force. In these figures, the most deformed region is illustrated in red and the least deformed in blue.

We compared our method with the FEM method by applying both methods on the objects in section 4.3.5, and the results are presented in the following sections. All results are shown for the chosen two hidden-layer architecture with 82 neurons in the first layer and

18 in the second layer (identified in section 4.3) as well as with the identified best parameters for the preprocessing steps (identified in section 4.1).



(a) (b)

(c) (d)

Figure 4.25. (a) Original mesh; (b) nodal force applied; (c) 15 scales of deformation after applied the nodal force; and (d) scales of deformation divided in 5 clusters.

4.5.1. Comparison for the Ball

In the case of the ball object, to compare our prediction method with the FEM method, we predicted deformed instances of the ball using our prediction method, and then we deformed the object mesh using the Ansys software using the same interaction parameters (i.e. same force magnitudes and angle and point of application that we used for prediction) and also by increasing the force magnitude, as illustrated in the first column of Table 4.10. We then compared the results by computing the average of Metro errors, perceptual errors and the average distance between the objects.

The first column in Table 4.10 represents the force magnitude range. The first range is between 0.25 N and 2 N and the last one between 80 N and 100 N. For example, in the first range [0.25 2[, we increased the force magnitude used in training between 0.25N and 1.75N (i.e. performed tests for 0.25 N, 0.50 N, 0.75 N, 1 N, 1.25 N, 1.50 N and 1.75 N, respectively). The same process was repeated for the other ranges; for example, in the last range between 80 N and 100 N, tests were performed for 80 N, 85 N, 90 N, 95 N and 100 N respectively. Table 4.10 summarizes the comparison results obtained for the ball (for the case when using 10 deformed instances for training). The columns 2, 3 and 4 of Table 4.10 represent, respectively, the average of Metro and perceptual errors, and the average distance error obtained by CloudCompare. As one can notice, the results show that our method achieves a Metro error average of $34.17e^{-3}$ and a similarity average of 68.67%. These results are not very accurate, the reason being the one that we identified before, namely the lack of sufficient data to train the network.

Table 4.10. Metro and perceptual error and distance average for the ball.

Force range in Newton	**Metro Error Max/Mean/Rms (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Average Distance Obtained by CloudCompare (e^{-3})**
[0.25 2[	21.039/4.741/6.608	0.2738 (72.62%)	4.429
[2 5[	23.998/6.268/8.139	0.2783 (72.17%)	5.861
[5 10[	32.348/9.876/12.057	0.2788 (72.12%)	9.344
[10 20[	42.995/14.273/17.112	0.2832 (71.68%)	13.654
[20 30[	49.824/19.952/23.161	0.2972 (70.28%)	19.159
[30 40[	52.791/26.251/29.507	0.3028 (69.72%)	25.226
[40 60[	113.314/46.064/54.102	0.3421 (65.79%)	44.907
[60 80[	121.159/49.095/58.409	0.3699 (63.01%)	48.121
[80 100]	156.408/91.182/98.487	0.3943 (60.57%)	90.760
Average	**68.208/29.745/34.176**	**0.3133 (68.67%)**	**29.051**

We therefore repeated the same comparisons but using the 50 deformed instances (obtained as detailed in section 4.3.4). Table 4.11 represents the errors measures obtained,

and also the average prediction time obtained by our method and by the FEM method for this case.

One can notice that, the Metro error and the average distance remain relatively stable when a force between 0.25 N and 59 N is used. As such, the Metro error remains smaller than $2.93e^{-3}$, the perceptual similarity greater than or equal 73.46% and the average distance smaller than or equal $2.36e^{-3}$. All these errors increase when using forces of larger magnitudes. These error measures show an improvement with respect to the results in Table 4.10, the performance of our method is still lower in terms of accuracy than the one of FEM. However, the prediction time achieved is much smaller (i.e. 0.026 seconds) than the deformation time taken by the FEM method (i.e. 6.67 seconds).

Table 4.11. Metro and perceptual error and distance average for the ball when using additional training data.

Force range in Newton	**Metro Error Max/Mean/Rms (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Average Distance (e^{-3})**	**Prediction Time (our method)**	**Deformation Time (FEM)**
[0.25 2[	6.903/2.800/2.926	0.2595 (74.05%)	2.260	0.045 s	6.763 s
[2 5[	6.978/2.723/2.847	0.2598 (74.02%)	2.200	0.032 s	6.401 s
[5 10[	7.107/2.623/2.750	0.2607 (73.93%)	2.095	0.026 s	6.932 s
[10 20[	7.286/2.557/2.699	0.2612 (73.88%)	2.043	0.025 s	6.672 s
[20 30[	7.445/2.572/2.758	0.2616 (73.84%)	2.040	0.024 s	7.215 s
[30 40[	7.587/2.658/2.888	0.2614 (73.86%)	2.114	0.020 s	6.328 s
[40 60[	10.149/2.928/3.371	0.2654 (73.46%)	2.364	0.025 s	6.742 s
[60 80[	16.281/4.477/5.449	0.2704 (72.96%)	3.852	0.021 s	6.129 s
[80 100]	23.247/7.221/8.627	0.2719 (72.81%)	6.469	0.024 s	6.849 s
Average	**10.760/3.469/3.924**	**0.2635 (73.65%)**	**2.897**	**0.027 s**	**6.670 s**

4.5.2. Comparison Using the Cylinder, the Cube and the Table

Additional tests were performed for the additional, synthetic objects (i.e. cylinder, cube and table that we used in the prediction method). The same forces were used as in the case of the ball to predict deformed instances of these objects. For the training of the neural network, we used again 50 deformed instances of each object. For each object, we computed the error measure as well as the average prediction time taken by our method and by the FEM method. Tables 4.12, 4.13 and 4.14 show respectively the comparison results obtained for the cylinder, cube and table. One can notice in Table 4.12 that the Metro error and the average distance remain stable for forces between 0.25 N and 59 N. The Metro error remains smaller than or equal $1.47e^{-3}$, the perceptual error and the average distance remains smaller than or equal $1.37e^{-3}$. These errors increase using a force between 60 N and 100 N. We obtain an average Metro error of $6.27e^{-3}$ and an average distance of $6.11e^{-3}$ for forces between 80 N and 100 N.

Table 4.12. Metro, perceptual errors and distance average for the cylinder.

Force range in Newton	**Metro Error Max/Mean/Rms (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Average Distance (e^{-3})**	**Prediction Time (our method)**	**Deformation Time (FEM)**
[0.25 2[	2.357/0.647/0.782	0.2108 (78.92%)	0.575	0.022 s	5.600 s
[2 5[	2.630/0.637/0.772	0.2116 (78.84%)	0.566	0.046 s	5.423 s
[5 10[	2.740/0.644/0.776	0.2157 (78.43%)	0.576	0.025 s	5.619 s
[10 20[	2.844/0.675/0.811	0.2655 (73.45%)	0.606	0.024 s	5.562 s
[20 30[	2.750/0.744/0.900	0.2734 (72.66%)	0.678	0.021 s	5.5267 s
[30 40[	2.996/0.871/1.024	0.2921 (70.79%)	0.799	0.020 s	5.656 s
[40 60[	4.590/1.476/1.723	0.3057 (69.43%)	1.370	0.022 s	5.531 s
[60 80[	9.518/3.603/4.208	0.3230 (67.7%)	3.424	0.021 s	5.648 s
[80 100]	16.414/6.274/7.110	0.3656 (63.44%)	6.113	0.031 s	5.552 s
Average	**5.204/1.730/2.018**	**0.2737 (72.63%)**	**1.634**	**0.026 s**	**5.569 s**

As in the case of the ball, the prediction time taken by our method (i.e. 0.025 seconds) is much smaller than the deformation time by the FEM method (i.e. 5.568 seconds). Similar conclusions can be drawn for the other objects in Tables 4.13 and Table 4.14.

Table 4.13. Metro and perceptual error and distance average for the cube.

Force range in Newton	**Metro Error Max/Mean/Rms (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Average Distance (e^{-3})**	**Prediction Time (our method)**	**Deformation Time (FEM)**
[0.25 2[	8.286/1.276/2.112	0.2765 (72.35%)	1.233	0.026 s	5.105 s
[2 5[	8.242/1.2763/2.106	0.2759 (72.41%)	1.237	0.024 s	5.068 s
[5 10[	8.224/1.265/2.0936	0.2761 (72.39%)	1.239	0.022 s	5.182 s
[10 20[	8.239/1.258/2.092	0.2763 (72.37%)	1.208	0.020 s	5.000 s
[20 30[	8.301/1.273/2.0937	0.2758 (72.42%)	1.217	0.024 s	5.057 s
[30 40[	8.170/1.393/2.172	0.2744 (72.56%)	1.343	0.020 s	5.031 s
[40 60[	8.334/1.810/2.455	0.2911 (70.89%)	1.753	0.025 s	5.071 s
[60 80[	7.575/2.267/2.756	0.2943 (70.57%)	2.189	0.022 s	5.102 s
[80 100]	9.073/3.039/3.329	0.2979 (70.21%)	2.948	0.023 s	5.093 s
Average	**8.272/1.651/2.356**	**0.2820 (71.80%)**	**1.596**	**0.023 s**	**5.079 s**

Finally, Table 4.15 summarizes the results achieved for the four objects (ball, cylinder, cube and table). The average Metro error for these objects is $4.82e^{-3}$, while the average similarities of these objects is 72.65% and the distance average is $4.03e^{-3}$. We can notice that, again, the average prediction time of our method (i.e. 0.024 seconds) is much smaller than the average deformation time taken by FEM (i.e. 5.36 seconds) for all objects. This makes our method more appropriate for applications in which the speed of prediction is important.

Table 4.14. Metro and perceptual error and distance average for the table.

Force range in Newton	**Metro Error Max/Mean/Rms (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Average distance (e^{-3})**	**Prediction time (our method)**	**Deformation time (FEM)**
[0.25 2[	12.772/2.299/3.331	0.2140 (78.6%)	2.473	0.022 s	4.105 s
[2 5[	12.814/2.214/3.243	0.2471 (75.29%)	2.364	0.018 s	4.188 s
[5 10[	12.867/2.180/3.204	0.2478 (75.22%)	2.303	0.020 s	4.203 s
[10 20[	14.283/2.627/3.797	0.2577 (74.23%)	2.781	0.012 s	4.059 s
[20 30[	19.239/4.555/6.519	0.2618 (73.82%)	4.887	0.021 s	4.083 s
[30 40[	21.418/5.520/7.682	0.2974 (70.26%)	5.873	0.017 s	3.923 s
[40 60[	34.071/16.398/18.405	0.3062 (69.38%)	16.728	0.016 s	4.414 s
[60 80[	35.266/20.780/21.349	0.,3125 (68.75%)	21.736	0.016 s	4.126 s
[80 100]	41.236/23.594/25.370	0.3267 (67.33%)	24.733	0.0215 s	4.104 s
Average	**22.663/8.907/10.322**	**0.2746 (72.54%)**	**9.319**	**0.0182 s**	**4.134 s**

Table 4.15. Metro and perceptual error and average distance for all objects.

Force range in Newton	**Metro Error Max/Mean/Rms (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Average Distance (e^{-3})**	**Prediction Time (our method)**	**Deformation Time (FEM)**
Ball	10.760/3.469/3.924	0.2635 (73.65%)	2.897	0.027 s	6.670 s
Cylinder	5.9821/2.3969/2.6787	0.2737 (72.63%)	2.3008	0.026 s	5.569 s
Cube	8.2713/1.6509/2.3563	0.2820 (71.80%)	1.596	0.023 s	5.079 s
Table	22.6631/8.9075/10.3222	0.2746 (72.54%)	9.3199	0.0182 s	4.134 s
Average	**11.9191/4.1061/4.8203**	**0.2735 (72.65%)**	**4.0284**	**0.024 s**	**5.363 s**

4.5.3. Tuning of Preprocessing Steps of the Proposed Prediction Method to Achieve Same Accuracy as FEM

Table 4.15 demonstrated that for the best choice of parameters that was identified for the preprocessing steps and the chosen neural network architecture (two hidden layers with 100 neurons), the proposed method, while much faster than the FEM method does not achieve the same quality of model (the average perceptual error is on average 72.65% and the mean Metro error is $4.11e^{-3}$). For this reason, additional tests were performed to identify how a different choice of preprocessing parameters can lead to a similar quality to the one achieved by FEM and identify the computational time required by our method to achieve this quality.

Table 4.16 shows that by modifying the percentage of points preserved during the selectively-densified simplification in the processing step of our method (section 3.2.3, Chapter 3) from the initial 40% to higher percentages leads to an increase in the training time, but also in the performance. The tests in this figure are reported for the case of the ball object. Increasing the percentage of points preserved during selectively densified simplification to 75% (instead of 40% preserved during the previous experiments) reaches virtually the same performance (99.77% similarity) as the FEM method. While the training time increases, the prediction time remains much smaller than the one incurred by FEM.

Table 4.16. Increasing quality of prediction by modifying the parameters of selectively-densified simplification (i.e. the percentage of points retained in the simplification).

Selectively-densified simplification %	**Overall Perceptual Error (Similarity %)**	**Prediction Time (our method)**	**Training Time / Iteration (our method)**
40%	0.2636 (73.61%)	0.033 s	81.188 s
45%	0.2313 (76.87%)	0.039 s	97.634 s
55%	0.1565 (84.35%)	0.073 s	145.242 s
65%	0.0735 (92.66%)	0.098 s	156.744 s
75%	0.0023 (99.77%)	0.430 s	310.474 s

4.6. Experimental Results for the Learning Transfer

This section presents experimental results that demonstrate the adaptability of our method for objects of different size, shapes and material by transfer of learning between neural networks, in particular demonstrates the ability to predict small objects from large objects and large objects from small objects. It is aligned with objective 3 (Chapter 1, section 1.1) and contribution 3 (Chapter 1, section 1.3).

For this purpose, two series of tests were executed. In the first test, we used the point clouds which represent 50 deformed instances of each object to tune the second network Net2. Then, in an attempt to reduce the training time, we have performed experiments using less data for tuning Net2. As such, we performed a second series of tests using only 20% of the data (point clouds which represent 10 deformed instances). In each test performed, we computed the training time of the neural network after the learning transfer and we compared it to the training time without it (when no transfer of learning is performed) to study the gains of using the transfer of learning.

As explained in Chapter 3, section 3.3.8, we used two neural networks (Net1 and Net2) to transfer the learning between objects of different sizes. Each network has two hidden layers (82 neurons in the first layer and 18 neurons in the second layer); this is the best architecture that was identified and used during experimentation (section 4.3). The first neural network is used to capture implicitly the object model and the second fine-tuned and used for prediction (Figure 3.10, Chapter 3). For example, to predict small objects from large objects, we train the first neural network Net1 using the deformed instances of the large object. After training of Net1, we transfer its learning to a second neural network Net2 by initially transferring from Net1 to Net2 all the weights matrices and biases as follows: (1) the weights matrix between the neurons of the input and the first hidden layer (matrix of 82 by 9); (2) the weights matrix between the neurons of the first and second hidden layers (matrix of 18 by 82); (3) the weights matrix between the neurons of the second hidden layer and the output layer (matrix of 3 by 18);(4) the biases between the input and the first hidden layer (vector of 82 by 1); (5) the biases between the first and second hidden layers (vector of 18 by 1), and (6) the biases between the second hidden and the output layer (vector of 3 by 1).

Once the architecture of Net2 constructed, we fine-tune it (train it shortly) using new data representing the deformed instances of the small object. Finally, the deformation for small objects is predicted using the tuned Net2. We proceed in a similar way for the prediction of large objects based on pretrained networks using small objects. For experimentation, we used the same objects used in the last sections (ball, cylinder, cube and table) and created (using Ansys) other objects of the same shape as these four objects but with different sizes (i.e. specific details on the number of vertices are provided in the following sections).

4.6.1. Transfer of Learning and Prediction for the Small Ball

To transfer the learning and predict the small balls, we used 50 deformed instances of the large ball with the different forces presented in Table 4.9. Following simplification, clustering, sampling, and neural gas tuning (with the parameters discussed in section 4.1) we obtained 1681 vertices for each instance, resulting in a total of 84050 vertices which represent all the large deformed balls. From these, we used, as for all the other experiments performed, 50% for training Net1, 5% for validation and 45% for testing. After the training and testing of Net1, we transferred its learning to Net2 (as explained in the previous section), and we fine-tuned Net2 on data representing deformed instances of the small ball. We have used 19050 vertices that represent all the deformed small balls. After training, we used it Net2 to predict the deformation on small balls with forces that we did not use during training. Table 4.17 presents a comparison of computation times without transfer of learning and with transfer of learning with different quantities of training data. We train Net2 without transfer of learning and we stopped the training when we obtained a small training error (i.e., mse=$2.76e^{-5}$). For tuning Net2, we also stopped the training when we obtained the same small training error (i.e., mse=$2.76e^{-5}$). Note that, the time taken to tune Net2 (413 seconds using 50 instances deformed of the small ball for training and 96.88 seconds using 10 deformed instances) is much smaller than the one taken without the transfer of learning (the training time is 512.50 seconds). The average Metro error for the predicted small ball is $4.30e^{-3}$ and the average of perceptual similarity is 73.26%.

Table 4.17. Average training time and prediction errors for small balls.

Training Time Without Learning Transfer (s)	**Training Time Using Learning Transfer (s)**		**Overall Metro Error (e^{-3}) Max/Mean/Rms**	**Overall Perceptual Error (Similarity %)**
	50 instances	**10 instances**		
512.50	431.60	96.88	10.626/4.301/4.689	0.2674 (73.26 %)

4.6.2 Transfer of Learning and Prediction of the Large Ball

In this case, the process is similar with the one in the previous section, but the transfer occurs from a small ball to a larger size one. As such, for evaluating the transfer, we train Net1 using 84050 vertices (coming from 50 deformed instances of the ball). After we transfer its learning to Net2, and tune Net2 on data representing 50 deformed instances of a newly created large ball. As each instance contains 4197 vertices, we use a total of 209850 points for training Net2. After training, we used it to predict the deformation on the large ball (with forces that we did not use in training). Table 4.18 presents the computation time for training Net2 after the learning transfer and a comparison with the time required to train Net2 without transfer of learning.

Table 4.18. Average of training time and prediction errors for the large ball.

Training Time Without Learning Transfer (s)	**Training Time Using Learning Transfer (s)**		**Overall Metro Error (e-3) Max/Mean/Rms**	**Overall Perceptual Error (Similarity %)**
	50 instances	**10 instances**		
1320.97	655.441 s	145.705 s	5.782/0.816/1.11	0.2308 (76.91%)

We stopped the training of Net2 when we obtained a good performance (i.e., mse=1.94e^{-5}). One can note that the training time without transfer of learning is very high (1320.97 seconds) with respect to the training time with transfer of learning (413 seconds using 50 deformed instances of the large ball for training and 96.88 seconds using 10 deformed

instances). The average Metro error for the predicted large ball is $0.816e^{-3}$ and the average of perceptual similarity is 76.91%.

4.6.3. Transfer of Learning and Prediction of Other Objects

To further demonstrate the abilities of our transfer method, we applied it on the three objects (cylinder, cube, and table) that were used for the prediction method. We repeated in this case the same process described in the previous section for training Net2 without transfer of learning or with transfer of learning. We also repeated the same series of tests the deformation on large objects (large cylinder, large cube and large table) from small objects. Table 4.19 summarizes the results obtained.

Table 4.19. Average of training time and prediction errors for the three objects.

Object	**Training Time Without Learning Transfer (s)**	**Training Time Using Learning Transfer (s)**		**Overall Metro Error (e-3) Max/Mean/Rms**	**Overall Perceptual Error (Similarity %)**
		50 instances	**10 instances**		
Cylinder	578.904 s	393.164 s	116.035 s	2.96/0.422/0.599	0.2834 (71.66%)
Cube	343.829 s	158.540 s	98.353 s	2.133/0.363/0.496	0.2656 (73.44%)
Table	281.076 s	188.508 s	48.857 s	5.32/1.145/1.467	0.2862 (71.38%)
Average	**401.269 s**	**246.737 s**	**87.748 s**	**3.47/0.643/0.85**	**0.2784 (72.16 %)**

As one can notice, the average Metro error for all the objects is $0.64e^{-3}$ and the average perceptual similarity is 72.16%.

Finally, we have also tested the ability of predicting an elasto-plastic object starting from a network trained on a deformable foam object with a slightly different shape as well (ball vs. cylinder). For experimentation, we trained the first neural network (Net1) on the dataset representing 50 deformed instances of the foam ball (same deformed instances used in section 4.6.1). After training Net1, we transferred its learning to Net2 an fine-tuned it on a dataset representing 50 deformed instances of an elasto-plastic cylinder (the

same one used in section 4.4.8). Table 4.20 shows the computation time that we took to train Net2 with and without the learning transfer method. We again stopped the training of Net2 when we obtained a good performance (i.e., mse=$4.48e^{-5}$). As for the previous objects, the time to train Net2 after the transfer of learning transfer method is lower than the one obtained without transfer of learning. The average Metro error for the predicted cylinder is $0.816e^{-3}$ and the average of perceptual similarity is 61.57%. One can notice that, as expected, the performance is deteriorated in this case with respect to the case when we predict for the same object shape or same object material.

Table 4.20. Average of training time and prediction errors for the elasto-plastic cylinder.

Training Time Without Learning Transfer (s)	**Training Time Using Learning Transfer (s)**		**Overall Metro Error (e-3) Max/Mean/Rms**	**Overall Perceptual Error (Similarity %)**
	50 instances	**10 instances**		
560.225 s	426.297 s	184.334 s	5.782/0.816/1.11	0.2309 (61.57%)

Overall, the training time using the transfer of learning is shorter for all the objects that we used in the experiments (Tables 4.17, 4.18, 4.19 and 4.20). Metro and perceptual errors tend to remain stable whether we use learning transfer or not because we can achieve almost the same performance by training the neural network. We can conclude that, with the learning transfer, we save a lot of training time and we are able to predict objects of different sizes and similar materials with a relatively good performance. However, the performance decreases, as expected, when transferring learning from one object to another object different in both shape and material (elasto-plastic cylinder from a foam ball).

CHAPTER 5

5. CONCLUSION

In this thesis, we addressed the problem of 3D object deformation through a novel computational intelligence approach. This approach is aimed at building a system that learns from examples how to predict automatically deformations of 3D objects that are subjected to point forces. To build such a system, three major issues have been addressed. The first issue is related to the need for a learning-based system that can be applied in a variety of real-world applications (e.g., robotics, manufacturing, etc.) involving contact forces with physical objects for which explicit knowledge of the physical parameters is not available. The second issue is related to ensuring that the prediction is performed in a computationally efficient way so as to facilitate that the system can be deployed in real-time scenarios where computation and memory resources are limited (e.g., robots, drones, etc.). The third issue is related to the capacity of the system to deal with the variability of objects in terms of size and physical properties.

To address the first issue, we proposed a method based on feed-forward networks (FFN) for predicting the 3D deformation of elastic and elasto-plastic objects that are subjected to point forces. The prediction does not make any assumption about the material of the objects. The whole prediction system is composed of a preprocessing step that simplifies the representation of objects and a prediction module that maps an input object, on which a force is applied at a certain point, to a deformed version of it.

To address the second issue, we proposed a novel preprocessing method to create compact 3D object representation by combining selectively-densified simplification, stratified sampling and neural gas fitting. This combination reduces the complexity of the deformed objects by creating compact and accurate representations of them; thus, allowing to bring savings in computation time and memory requirements.

To address the final issue, we improved our prediction method using transfer learning between neural networks models. This allowed us to use the learning of a pre-trained network on one object to accelerate the training of another object network given that the physical properties of the objects are similar except for their size and/or slight geometrical differences. Therefore, we can predict the deformation of a small ball from training on large ball and vise-versa. Transfer of learning is also useful to reduce the amount of data required for training the feedforward neural network architectures.

Our prediction system has been successfully tested and validated using several examples of object deformations and achieved promising results. Among other things, our system allows to predict deformations in a similar way to FEM despite the lack of knowledge about the physical parameters of the objects. Our prediction is also computationally efficient, which makes it applicable to any real-time scenario.

Although we proposed an efficient and promising method to address the challenges in the literature for 3D deformation estimation, there are still some limitations for predicting accurately complex object deformations. First, since surface point clouds (vertices) are processed separately, the predicted deformations might not be consistent with the local geometry of the object. Second, all objects have to be aligned in the world coordinate system in order to be processed correctly. Consequently, alignment errors could result in prediction errors. To address the first issue, one potential way is to consider the local geometry of objects. To address the second issue, the objects themselves could be endowed with an intrinsic coordinate system (i.e. spherical or geodesic coordinates).

We can apply our prediction method in several fields such as robotics, manufacturing, and agriculture. Our method can be adapted and tested in the context of robotic manipulation tasks, in which the force-torque sensor measurements will be replaced by force measures exercised at the level of each robotic finger. For example, robotic arms could be designed to automatically evaluate the ripeness of fruits and vegetables and also help harvest them automatically without damaging them. As such, the robotic arm requires the ability to detect, recognize, and determine if the fruit or vegetable is ripe

enough to be picked or rotten (to throw it away). In a more general context, the same technology can be applied in other industries where robots are required to explore (sometimes dangerous, contaminated or inaccessible) environments autonomously and to handle and predict the deformations on new objects without knowing their physical parameters.

Bibliography

[1] A. S. Sekercioglu and A. Duysak. Application of molecular modeling with mass-spring systems for computer simulation and animation. International Journal of Physical Sciences, vol. 4, no. 9, pp. 500-504, September 2009.

[2] C. Garre and A. Pérez. A simple Mass-Spring system for Character Animation. Computer Animation Final Project, June 2008.

[3] C. Paloc, F. Bello, R. I. Kitney and A. Darzi. Online Multiresolution Volumetric Mass Spring Model for Real Time Soft Tissue Deformation. Proc. Medical Image Computing and Computer-Assisted Intervention, pp. 219-226, Tokyo, Japan, 2002.

[4] T. D. Giacomo and N. Magnenat-Thalmann. Bi-Layered Mass-Spring Model for Fast Deformations of Flexible Linear Bodies. Proc. Int. Conf. Computer Animation and Social Agents, pp. 48-54, May 2003.

[5] M. Camara, E. K. Mayer, A. Darzi and P. Pratt. Soft tissue deformation for surgical simulation: a position-based dynamics approach. International Journal of Computer Assisted Radiology and Surgery, vol. 11, no. 6, pp. 919-928, 2016.

[6] B. Frank, R. Schmedding, C. Stachniss, M. Teschner and W. Burgard. Learning the elasticity parameters of deformable objects with a manipulation robot. In Proceedings of International Conference on Intelligent Robots and Systems(IROS), pp. 1877-1883, 2010.

[7] S. Cui, R. Wang, J. Wei, F. Li and S. Wang. Grasp State Assessment of Deformable Objects Using Visual-Tactile Fusion Perception. 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 538-544, 2020, doi: 10.1109/ICRA40945.2020.9196787.

[8] M. C. Gemici and A. Saxena. Learning haptic representation for manipulating deformable food objects. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014), pp. 638-645, September 2014.

[9] V. Rochus and D. Rixen. Modelling the electromechanical coupling of RF switch using Extended Finite Element. International Conference on Thermal, Mechanical and Multi-Physics Simulation Experiments in Microelectronics and Micro-Systems, pp. 1-8, 2007.

[10] M. Becker and M. Teschner. Robust and Efficient Estimation of Elasticity Parameters Using the Linear Finite Element Method. In Proc. of Simulation and Visualization, pp. 15-28, 2007.

[11] R. Wang and J.-M. Jin. A Symmetric Electromagnetic-Circuit Simulator Based on the Extended Time-Domain Finite Element Method. IEEE Transactions on Microwave Theory and Techniques, vol. 56, no. 12, pp. 2875-2884, 2008.

[12] B. A. T. Iamamura, Y. L. Menach, A. Tounzi, N. Sadowski and E. Guillot. Study of Static and Dynamic Eccentricities of a Synchronous Generator Using 3-D FEM. IEEE Transactions on Magnetics, vol. 46, no. 8, pp. 3516-3519, 2010.

[13] L. Zaidi, B. Bouzgarrou, L. Sabourin and Y. Menzouar. Interaction modeling in the grasping and manipulation of 3D deformable objects. In Proceedings of the IEEE International Conference on Advanced Robotics, Istanbul, Turkey, doi:10.1109/ICAR.2015.7251503, 27-31 July 2015.

[14] C. Elbrechter, R. Haschke and H. Ritter. Folding paper with anthropomorphic robot hands using real-time physics-based modeling. In Proceedings of the IEEE International Conference Humanoid Robots, pp. 210-215, Osaka, Japan, 29 November–1 December 2012.

[15] X. Tu and D. Terzopoulos. Artificial fishes: physics, locomotion, perception, behavior. In Proceedings of SIGGRAPH, pp. 43-50, 1994.

[16] G. Miller. The motion dynamics of snakes and worms. In Proceedings of SIGGRAPH, pp. 169-178, 1988.

[17] Y. Li, H. Xu and J. Barbič. Enriching Triangle Mesh Animations with Physically Based Simulation. IEEE Transactions on Visualization and Computer Graphics, vol. 23, no. 10, pp. 2301-2313, October 2017.

[18] H. Pfister, M. Zwicker, J. van Baar and M. Gross. Surfels: Surface Elements as Rendering Primitives. In: SIGGRAPH 2000, Proceedings of the 27th Annual Conference on Computer Graphics, pp. 335-342, July 2000.

[19] V. Andersen, H. Aans, J. A. Brentzen, J. P. Collomosse and I. J. Grimstead. Surfel based geometry reconstruction. Theory and Practice of Computer Graphics, pp. 39-44, 2010.

[20] M. R. Ruggeri, D. V. Varnic and D. Saupe. Shape similarity search for surfel-based models. In Proc. of Int. Conf. on Computer Vision and Graphics 2004, pp. 131-140, September 2004.

[21] R. L. Carceroni and K. N. Kutulakos. Multi-view scene capture by surfel sampling: From video streams to non-rigid 3D motion, shape and reflectance. IJCV, vol. 49, no. 2-3, pp. 175-214, 2002.

[22] G. R. Liu. Mesh Free Methods – Moving beyond the Finite Element Method, CRC Press, USA, 792 p., 2003.

[23] Y. Zhao and W. N. Fu. A fast remesh-free mesh deformation method based on radial basis function interpolation and its application to optimal design of electromagnetic devices. Progress in Electromagnetic Research Symposium (PIERS), pp. 313-314, August 2016.

[24] B. Honarbakhsh and A. Tavakoli. Numerical solution of electromagnetic integral equations by the meshfree collocation method. Applied Computational Electromagnetics Society Journal, vol. 27, no. 9, pp. 706-716, 2012.

[25] M. Desbrun and M.-P. Cani. Animating soft substances with implicit surfaces. In Computer Graphics Proceedings (1995), ACM SIGGRAPH, vol. 2, no. 6, pp. 287-290, 1995.

[26] J. Lang, D. K. Pai and R. J. Woodham. Acquisition of elastic models for interactive simulation. The International Journal of Robotics Research, vol. 21, no. 8, pp. 713-733, 2002.

[27] A. Jordt and R. Koch. Fast tracking of deformable objects in depth and colour video. In Proceedings of the British Machine Vision Conference, Dundee, UK, 30 August–1 September 2011; Hoey, J., McKenna, S., Trucco, E., Eds., pp. 114.1-114.11, 2011.

[28] V. Mozafary and P. Payvandy. Study and comparison techniques in fabric simulation using mass spring model. International Journal of Clothing Science and Technology, vol. 28, no. 5, pp.634-689, 2016.

[29] A. S. Saada. Elasticity theory and applications. 2nd Edition, Krieger Publishing Company, United States, 1993.

[30] A. Petit, F. Ficuciello, G. A. Fontanelli, L. Villani and B. Siciliano. Using Physical Modeling and RGB-D Registration for Contact Force Sensing on Deformable Objects. IEEE International Conference on Informatics in Control Automation and Robotics, pp. 24-33, 2017.

[31] U. Meier, O. Lopez, C. Monserrat, M. C. Juan and M. Alcaniz. Real-time deformable models for surgery simulation: a survey. Computer Meth. and Programs in Biomedicine, vol. 77, no. 3, pp. 183-197, 2005.

[32] F. Devernay, D. Mateus and M. Guilbert. Multi-Camera Scene Flow by Tracking 3-D Points and Surfels. 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06), vol. 2, pp. 2203-2212, 2006.

[33] J. S. Chen, C. T. Wu, S. Yoon and Y. You. A stabilized conforming nodal integration for Galerkin mesh-free methods. International Journal for Numerical Methods Engineering, vol. 50, no. 2, pp. 435-466, 2001.

[34] M. R. Yousefi, R. Jafari and H. A. Moghaddam. A Combined Wavelet-Based Mesh-Free Method for Solving the Forward Problem in Electrical Impedance Tomography. Instrumentation and Measurement IEEE Transactions on, vol. 62, no. 10, pp. 2629-2638, 2013.

[35] F. S. Cohen, W. Ibrahim and C. Pintavirooj. Ordering and Parameterizing Scattered 3D Data for B-Spline Surface Approximation. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 6, pp. 642-648, June 2000.

[36] P. Cignoni, C. Rocchini and R. Scopigno. Metro: Measuring error on simplified surfaces. Computer Graphics Forum, vol. 17, no. 2, pp. 167-174, June 1998.

[37] Z. Wang, A. Bovik, H. R. Sheikh and E. P. Simoncelli. Image quality assessment: From error visibility to structural similarity. IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-6012, April 2004.

[38] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle. Mesh optimization. Proceedings of the 20th annual conference on Computer graphics and interactive techniques, pp. 19-26, 1993.

[39] R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. Computer Graphics Forum (Proceedings of Eurographics 96), vol. 15, no. 3, pp. 67-76, 1996.

[40] A. Guéziec. Surface simplification with variable tolerance. In Second Annual international symposium on Medical Robotics and Computer Assisted Surgery, pp. 132–139, 1995.

[41] D. Arnas, C. Leake and D. Mortari. Random Sampling Using K-Vector. Computing in Science & Engineering, vol. 21, no. 1, pp. 94-107, 2019, DOI: 10.1109/MCSE.2018.2882727.

[42] A. Doucet, S. Godsill and C. Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. Statistics and Computing, vol.10, pp. 197-208, 2000.

[43] M. Ghosh, H.-Y. Yeh, S. Thomas and N. M. Amato. Nearly Uniform Sampling on Surfaces with Applications to Motion Planning. Technical Report, Parasol Lab, Department of Computer Science and Engineering, Texas A&M University, April 2013.

[44] M. A. Bujang, P. A. Ghani, N. A. Zolkepali, M. M. Ali, T. H. Adnan, S. Selvarajah and J. Haniff. Modification of systematic sampling: A comparison with a conventional approach in systematic sampling. 2012 International Conference on Statistics in Science, Business and Engineering (ICSSBE), pp. 1-4, 2012.

[45] A.-M. Cretu, P. Payeur and E. M. Petriu. Neural network mapping and clustering of elastic behavior from tactile and range imaging for virtualized reality applications. IEEE Transactions on Instrumentation and Measurement, vol. 57, no. 9, pp. 1918-1928, 2008.

[46] A.-M. Cretu, P. Payeur and E. M. Petriu. Selective range data acquisition driven by neural gas networks. IEEE Transactions on Instrumentation and Measurement, vol. 58, no. 8, pp. 2634-2642, 2009.

[47] W. C. Oliver and G. M. Pharr. Measurement of hardness and elastic modulus by instrumented indentation: Advances in understanding and refinements to methodology. Journal of Materials Research, vol. 19, no. 1, pp. 3-20, 2004.

[48] Y.-Q. Leng, Z.-C. Chen, X. He, Y. Zhang and W. Zhang. Collision Sensing Using Force/Torque Sensor. Journal of Sensors, vol. 2016, no. 3, pp. 1-10, 2016.

[49] M. Kaneko and K. Tanie. Contact point detection for grasping an unknown object using self-posture changeability. IEEE Transactions on Robotics and Automation, vol. 10, no. 3, 1994.

[50] C. Ho, C. Basdogan and M. Srinivasan. Haptic Rendering: Point- and Ray-Based Interactions. Proceedings of the Second PHANToM Users Group Workshop, Dedham, October 1995.

[51] C. Forest, H. Delingette and N. Ayache. Surface Contact and Reaction Force Models for Laparoscopic Simulation. In International Symposium on Medical Simulation, vol. 3078, pp. 168-176, 204.

[52] M.A. Srinivasan and C. Basdogan. Haptics in Virtual Environments: Taxonomy, Research Status, and Challenges. Computers and Graphics, vol. 21, no. 4, pp. 393-404, 1997.

[53] W. McNeely, K. Puterbaugh and J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. In Proceedings of ACM SIG GRAPH, pp. 401-408, 1999.

[54] C. Ho, C. Basdogan and M.A. Srinivasan. Ray-Based Haptic Rendering: Force and Torque Interactions Between a Line Pobe and 3D Objects in Virtual Environments. International Journal of Robotics Research, vol. 19, no. 7, pp. 668-683, 2000.

[57] Y.-K. Zhang and Y. Xiao. A practical method of 3d reconstruction based on uncalibrated image sequence. In Signal Processing, ICSP 2008. 9th International Conference on, pp. 1368-1371, October 2008.

[58] J. Ze-tao, Z. Bi-na, W. Min and W. Wen-huan. A Fully Automatic 3D Reconstruction Method Based on Images. 2009 WRI World Congress on Computer Science and Information Engineering, vol. 5, pp. 327-331, 2009.

[59] S. Y. Lee, Z. Majid and H. Setan. 3D Data Acquisition for Indoor Assets using Terrestrial Laser Scanning. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, vol. II-2/W1, ISPRS 8th 3D Geo Info Conference & WG II/2 Workshop, Istanbul, Turkey, November 2013.

[60] X-J. Cheng, H-F. Zhang and R. Xie. Study on 3D laser scanning modeling method for Large-Scale history building. International Conference on Computer Application and System Modeling (ICCASM 2010), vol. 7, pp. V7-573-V7-577, 2010.

[61] K. Bartol, D. Bojanic, T. Petkovic and T. Pribanic. A Review of Body Measurement Using 3d Scanning. Ieee Access, 9 DOI: 10.1109/ACCESS.2021.3076595, 2021.

[62] A. Haleem and M. Javaid. Clinical Epidemiology and Global Health, published by Elsevier, a division of RELX India 2018, https://doi.org/10.1016/j.cegh.2018.05.006.

[63] D. L. Yang, J. H. Chang, M. C. Hung and J. S. Liu. An efficient k-means-based clustering algorithm. In Proceedings of 1st Asia-Pacific Conference on Intelligent Agent Technology, pp. 269-273, 1999.

[64] H.-Y. Lin and Y.-H. Xiao. Free-viewpoint image synthesis based on non-uniformly resampled 3D representation. IEEE 17th International Conference on Image Processing, pp. 2745-2748, September 2010.

[65] S. Ding, X. Zhang, H. Chen and J. Wang. Large Scale Object's Measurement Method Research Based on Multi-View Reconstruction. 2nd International Conference on Image, Vision and Computing (ICIVC), pp. 555-558, June 2017.

[66] D. Fritsch, M. Abdel-Wahab, A. Cefalu and K. Wenzel. (2012) Photogrammetric Point Cloud Collection with Multi-Camera Systems. In: Ioannides, M.; Fritsch, D.; Leissner, J.; Davies, R.; Remondino, F.; Caffo, R. (eds) Progress in Cultural Heritage Preservation. Lecture Notes in Computer Science, vol. 7616. Springer, Berlin, Heidelberg, 2012.

[67] H. Wang. Three-dimensional medical ct image reconstruction. In Proceedings of the 2009 International Conference on Measuring Technology and Mechatronics Automation, vol. 01, pp. 548-551, April 2009.

[68] Q. Huang, H. Wang and V. Koltun. Single-view reconstruction via joint analysis of image and shape collections. ACM Transactions on Graphics, vol. 34, no. 4, pp. 1-10, August 2015.

[69] A. Nazar, V. Tataryn, Y. L. Bobitski and S. Ponomarenko. 3D scanner with modulation of light intensity. 2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), pp. 176-178, 2017.

[70] R. Wang, A.C.C. Law, D. Garcia and Z.J. Kong. Development of Structured Light 3D-Scanner with High Spatial Resolution and its Applications for Additive Manufacturing Quality Assurance. The International Journal of Advanced Manufacturing Technology, vol. 117, pp. 845-862, 2021.

[71] P.-H. ALLARD, J.-A LAVOIE. Differentiation of 3D scanners and their positioning method when applied to pipeline integrity. 11th European Conference on Non-Destructive Testing (ECNDT 2014), Prague, Czech Republic, October 2014.

[72] A. Malhotra, K. Gupta and K. Kant. Laser Triangulation for 3D Profiling of Target. International Journal of Computer Applications, vol. 35, no. 8, pp. 47-50, December 2011.

[73] X. Li. 3D body scan line point cloud processing. The 2nd Conference on Environmental Science and Information Application Technology, vol. 3, pp. 333-336, 2010.

[74] F. Buonamici, M. Carfagni, L. Puggelli, M. Servi and Y. Volpe. A Fast and Reliable Optical 3D Scanning System for Human Arm. In book: Advances on Mechanics, Design Engineering and Manufacturing III, Proceedings of the International Joint Conference on Mechanics, Design Engineering & Advanced Manufacturing, JCM 2020, June 2-4, 2020.

[75] H. Lin, F. Guo, F. Wang, and Y.B. Jia. Picking up a Soft 3D Object by Feeling the Grip. The International Journal of Robotics Research, vol. 34, no. 11, pp. 1361-1384, 2015.

[76] D. Navarro-Alarcon, H.M. Yip, Z. Wang, Y.H. Liu, F. Zhong, T. Zhang, and P. Li. Automatic 3-D Manipulation of Soft Objects by Robotic Arms with an Adaptive Deformation Model. IEEE Transactions on Robotics, vol. 32, no. 2, pp. 429-441, 2016.

[77] A.N. Ruchay, K.A. Dorofeev and V.I. Kolpako. Fusion of information from multiple Kinect sensors for 3D object reconstruction. Computer Optics, vol. 42, no. 5, pp. 898-903. DOI: 10.18287/2412-6179-2018-42-5-898-903, 2018.

[78] T. Gupta and H. Li. Indoor mapping for smart cities — An affordable approach: Using Kinect Sensor and ZED stereo camera. International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1-8, 2017.

[79] B. Shen, F. Yin and W. Chou. A 3D Modeling Method of Indoor Objects Using Kinect Sensor. 10th International Symposium on Computational Intelligence and Design (ISCID), vol. 1, pp. 64-68, 2017.

[80] H. Xu, L. Shi and X. Wang. Fast 3D-Object Modeling with Kinect and Rotation Platform. 2015 Third International Conference on Robot, Vision and Signal Processing (RVSP), pp 43-46, 2015.

[81] A. Ballit, I. Mougharbel, H. Ghaziri and T.T. Dao. Visual Sensor Fusion with Error Compensation Strategy Toward a Rapid and Low-Cost 3D Scanning System for the Lower Residual Limb. In IEEE Sensors Journal, vol. 20, no. 24, pp. 15043-15052, December 15, 2020, doi: 10.1109/JSEN.2020.3011172.

[82] A. Delgado, C.A. Jara, and F. Torres. Adaptive Tactile Control for In-Hand Manipulation Tasks of Deformable Objects. The International Journal of Advanced Manufacturing, vol. 91, pp. 4127-4140, 2017.

[83] M. L. R-Rodriguez, V. Kober. Reconstruction of 3D deformable objects using a single Kinect sensor. Proc. SPIE 11137, Applications of Digital Image Processing XLII, vol. 1113725, 6 September 2019. https://doi.org/10.1117/12.2528526.

[84] C. Chen, W. Zou and J. Wang. 3D surface reconstruction based on Kinect. IEEE Third International Conference on Information Science and Technology (ICIST), pp. 986-990, March 2013.

[85] A. Ruchay, K. Dorofeev and V. Kalschikov. Real-time dense 3D object reconstruction using RGB-D sensor. Conference Proceedings Article Applications of Digital Image Processing XLIII, vol. 11510, Aug 21, 2020.

[86] T. Wiemann, I. Mitschke, A. Mock and J. Hertzberg. Surface Reconstruction from Arbitrarily Large Point Clouds. 2018 Second IEEE International Conference on Robotic Computing (IRC), pp. 278-281, 2018.

[87] D. Osmanković and J. Velagić. Reconstructing the 3D thermal model of indoor environment from unorganized data set acquired by 3D laser scans and thermal imaging camera. IEEE International Symposium on Intelligent Control (ISIC), pp. 13-18, October 2012.

[88] K. Wang, G. Zhang and X. Shihong. Templateless Non-Rigid Reconstruction and Motion Tracking with a Single RGB-D Camera. IEEE Transactions on Image Processing, vol. 26, no. 12, pp. 5966-5979, December 2017.

[89] K. Li, T. Pham, H. Zhan and I. Reid. Efficient Dense Point Cloud Object Reconstruction using Deformation Vector Fields. Proceedings of the European Conference on Computer Vision (ECCV), pp. 497-513, 2018.

[90] H. Monette-Thériault, A.-M. Cretu and P. Payeur. 3D object modeling with neural gas based selective densification of surface meshes. In Proceedings of the IEEE Conference on Systems, Machine and Cybernetics, pp. 1373-1378, San Diego, CA, USA, 5–8 October 2014.

[91] N. Altman and M. Krzywinski. Clustering. Nature Methods, vol. 14, no. 6, pp. 545-546, 2017, DOI: 10.1038/nmeth.4299.

[92] G.J. McLachlan, R.W. Bean and S.K. Ng. Clustering. Methods in Molecular Biology, vol. 1526, pp. 345-362, 2017.

[93] S. Wang, Y. Fan, C. Zhang, H. Xu, X. Hao and Y. Hu. Subspace Clustering of High Dimensional Data Streams. Seventh IEEE/ACIS International Conference on Computer and Information Science (icis), pp. 165-170, 2008.

[94] M. Van de Velden, A. Iodice D'Enza and A. Markos. Distance-Based Clustering of Mixed Data. Wiley Interdisciplinary Reviews: Computational Statistics, vol. 11, no. 3, 2019, DOI: 10.1002/wics.1456.

[95] C. Chauvel, A. Novoloaca, P. Veyre, F. Reynier and J. Becker. Evaluation of Integrative Clustering Methods for the Analysis of Multi-Omics Data. Briefings in Bioinformatics, vol. 21, no. 2, pp. 541-552, 2020, DOI: 10.1093/bib/bbz015.

[96] J. Ren, L. Li and C. Hu. A Weighted Subspace Clustering Algorithm in High-Dimensional Data Streams. Innovative Computing Information and Control (ICICIC) 2009 Fourth International Conference on, pp. 631-634, 2009.

[97] O. Abu Abbas. Comparison between data clustering algorithms. The international Arab Journal of Information Technology, vol. 5, no. 3, pp. 320-325, July 2008.

[98] M. Singh, P. Patel, D. Khosla and T. Kim. Segmentation of functional MRI by k-means clustering. IEEE Transactions on Nuclear Science, vol. 43, pp. 2030-2036, 1996.

[99] F. Hržić, V. Jansky, D. Sušanj, G. Gulan, I. Kožar and D. Z. Jeričević. Information entropy measures and clustering improve edge detection in medical X-ray images. 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), pp. 0164-0166, 2018.

[100] C. W. Chen, J. Luo and K. J. Parker. Image segmentation via adaptive k-mean clustering and knowledge-based morphological operations with biomedical applications. IEEE Transactions on Image Processing, vol. 7, pp. 1673-1683, 1998.

[101] R. Harikumar and P. S. Kumar. Classifiers for the Epilepsy Risk Level Classification from Electroencephalographic Signals. Research Journal of Pharmaceutical, Biological and Chemical Sciences, vol.6, no. 4, pp: 469-473, July 2015.

[102] R. Evans, B. Pfahringer and G. Holmes. Clustering for classification. In Proceedings of the 2011 7th International Conference on Information Technology in Asia, pp. 1-8, 2011.

[103] T. Zhang, R. Ramakrishnan and M. Livny. BIRCH: a new data clustering algorithm and its applications. Data Mining and Knowledge Discovery, vol. 1, pp. 141-182, 1997.

[104] D. Jain, M. Singh and A. K. Sharma. Comparative Study of Density Based Clustering Algorithms for Data Mining. 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), pp. 1559-1564, 2017.

[105] K.K. Pandey and D Shukla. Stratified Sampling-Based Data Reduction and Categorization Model for Big Data Mining. Lecture Notes in Networks and Systems, vol. 120, pp. 107-122, 2020, DOI: 10.1007/978-981-15-3325-9_9.

[106] E. Rodolà, A. Albarelli, D. Cremers and A. Torsello. A simple and effective relevance-based point sampling for 3D shapes. Pattern Recognition Letters, vol. 59, pp. 41-47, 2015.

[107] G. Turk. Re-tiling polygonal surfaces. In Proceedings of SIGGRAPH'92, ACM Press, pp. 55-64, 1992.

[108] C. Shih, L. Gerhardt, W. Chu, C.-H Lin, C. H. Chang, C.-H Wan and C.-S. Koong. Non-uniform surface sampling techniques for three-dimensional object inspection. Optical Engineering (OPT ENG), vol. 47, no. 5, May 2008.

[109] T. Nguyen and B. Boyce. An inverse finite element method for determining the anisotropic properties of the cornea. Biomechanics an Modeling in Mechanobiology, vol. 10, no. 3, pp. 323-337, 2011.

[110] J. Schulman, A. Lee, J. Ho and P. Abbeel. Tracking deformable objects with point clouds. In Proceedings of the IEEE International Conference Robotics and Automation, pp. 6-10, Karlsruhe, Germany, May 2013.

[111] A.M Cretu, P. Payeur, and E.M. Petriu. Soft Object Deformation Monitoring and Learning for Model-Based Robotic Hand Manipulation. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) , vol. 42, no. 3, pp. 740-753, 2012

[112] R. M. Koch, M. H. Gross, F. L. Carls, D. F. Buren, G. Fankhauser and Y. I. H. Parish. Simulating facial surgery using finite element methods. In Proceedings of SIGGRAPH, pp. 421-428, 1996.

[113] M. Krainin, P. Henry, X. Ren and D. Fox. Manipulator and object tracking for in-hand 3D object modeling. The International Journal of Robotics Research, vol. 30, no. 11, pp. 1311-1327, 2011.

[114] J. Chang and J. J. Zhang. Mesh-free deformations. Computer Animation and Virtual Worlds, vol. 15, n0. 3-4, pp. 211-218, July 2004.

[115] S. Cotin, H. Delingette and N. Ayache. Real-Time Elastic Deformations of Soft Tissues for Surgery Simulation. IEEE Transactions on Visualization and Computer Graphics, vol.5 no.1, pp. 62-73, January 1999.

[116] S. Arnab and V. Raja. Simulating a Deformable Object using a Surface Mass Spring System. 3rd International Conference on Geometric Modeling & Imaging, pp. 21-26, 2008.

[117] D. Terzopoulos and A. Witkin. Physically Based Models with Rigid and Deformable Components. IEEE Computer Graphics and Applications, pp. 41-51, 1988.

[118] Y. Wang, Y. Xiong, K. Xu, K. Tan and G. Guo. A mass-spring model for surface mesh deformation based on shape matching. Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, pp. 375-380, November 2006.

[119] A. Ballit, I. Mougharbel, H. Ghaziri and T.T. Dao. Fast soft tissue deformation and stump-socket interaction toward a computer-aided design system for lower limb prostheses. Irbm, vol. 41, no. 5, pp. 276-285, DOI: 10.1016/j.irbm.2020.02.003.

[120] S. Burion, F. Conti, A. Petrovskaya, C. Baur and O. Khatib. Identifying physical properties of deformable objects by using particle filters. In Proceedings of the IEEE International Conference Robotics and Automation, pp. 1112-1117, Pasadena, CA, USA, 19–23 May 2008.

[121] M.-H. Choi, S. C. Wilber and M. Hong. Estimating material properties of deformable objects by considering global object behavior in video streams. Multimedia Tools and Applications, vol. 74, no. 10, pp. 3361-3375, 2015.

[122] Y. Wang, Y. Xiong, K. Xu, K. Tan and G. Guo. A mass-spring model for surface mesh deformation based on shape matching. Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, pp. 375-380, November 2006.

[123] D. F. Pilkey. Computation of a damping matrix for finite element model updating. Ph.D. Thesis, Department of Engineering Mechanics, Virginia Polytechnical Institute and State University, 1998.

[124] Z. Liang and G. C. Lee. Representation of a Damping Matrix. Journal of Engineering Mechanics, vol. 117, no. 5, pp. 1005-1020, 1991.

[125] M. Ferrant, A. Nabavi, B. Macq, P. M. Black, F. A. Jolesz, R. ikinis and S. K. Warfield. Registration of 3D intraoperative MR images of the brain using a finite element biomechanical model. IEEE Transactions on Medical Imaging, vol. 20, pp. 1384-1397, 2001.

[126] M. Bro-Nielsen. Finite element modeling in surgery simulation, Proceedings of the IEEE, vol. 86, no. 3, pp. 490-503, 1997.

[127] Z. Wang and S. Hirai. Green Strain Based FE Modeling of Rheological Objects for Handling Large Deformation and Rotation. IEEE International Conference on Robotics and Automation, pp. 4762-4767, May 2011.

[128] S. Wuhre, J. Lang and C. Shu. Tracking complete deformable objects with finite elements. Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission, pp. 1-8, 2012.

[129] A. Petit, V. Lippiello and B. Siciliano. Real-time tracking of 3D elastic objects with an RGB-D sensor. In Proceedings of the IEEE International Conference Intelligent Robots and Systems, pp. 3914-3921, Hamburg, Germany, 28 September-2 October 2015.

[130] G. Bianchi, B. Solenthaler, G. Szekely and M. Harders. Simultaneous topology and stiffness identification for mass-spring models based on FEM reference deformations. In Medical Image Computing and Computer-Assisted Intervention (MICCAI), Saint-Malo, France, 26–29 September 2004; Barillot, C., Haynor, D.R., Falcao e Cunha, J., Hellier, P., Eds.; Springer Heidelberg, Germany, LNCS 3217; pp. 293–301.

[131] R. M. Bosse and A. T. Beck. Simplified Finite Elements model to represent Mass-Spring structures in dynamic simulation. Proceedings of the joint ICVRAM ISUMA UNCERTAINTIES conference, Florianópolis, SC, Brazil, pp.1-20, April 2018.

[132] I. Tobor, C. Schlick and L. Grisoni. Rendering by surfels. Technical report, LaBRI, Université de Bordeaux 1, France, 2000.

[133] S. Zahiri. Meshfree Methods. In J. Awrejcewicz (Ed.), Numerical Analysis–Theory and Application, Chap. Meshfree Methods, InTech, pp. 231-250, Rijeka, Croatia, 2011.

[134] NURBS. Available online: https://slideplayer.com/slide/8436775/ (accessed on 10 September 2018).

[135] Z. Wang, S. Rosa, L. Xie, B. Yang, S. Wang, N. Trigoni, and A. Markham. Defo-Net: Learning body deformation using generative adversarial networks. 2018 IEEE International Conference on Robotics and Automation (ICRA), pp. 2440-2447, 2018.

[136] A. J. Valencia, F. Nadon and P. Payeur. Toward Real-Time 3D Shape Tracking of Deformable Objects for Robotic Manipulation and Shape Control. IEEE SENSORS, pp. 1-4, doi: 10.1109/SENSORS43011.2019.8956623, Montreal, QC, Canada, 2019.

[137] B. Fritzke. A Growing Neural Gas Network Learns Topologies. In Advances in Neural Information Processing Systems 7. MIT Press pp. 625-632, 1995.

[138] D. Mrowca, C. Zhuang, E. Wang, N. Haber, L.F. Fei-Fei, J. Tenenbaum, and D.L. Yamins. Flexible neural representation for physics prediction. In Advances in Neural Information Processing Systems 31, pp. 8799-8810, 2018.

[139] Z. Hu, T. Han, P. Sun, J. Pan, and D. Manocha. 3-D Deformable Object Manipulation Using Deep Neural Networks. In IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 4255-4261, Oct. 2019.

[140] A.-M. Cretu. Experimental data acquisition and modeling of three-dimensional deformable objects using neural networks. PhD thesis, University of Ottawa, 2009.

[141] A. Tsoli and A. A. Argyros. Patch-Based Reconstruction of a Textureless Deformable 3D Surface from a Single RGB Image. 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 4034-4043, doi: 10.1109/ICCVW.2019.00498, Seoul, Korea (South), 2019.

[142] V.E. Arriola-Rios and J.L. Wyatt. A Multimodal Model of Object Deformation Under Robotic Pushing. IEEE Transactions on Cognitive and Developmental Systems, vol. 9, no. 2, pp. 153-169, 2017.

[143] Z. Wang, S. Rosa, B. Yang, S. Wang, N. Trigoni, and A. Markham. 3d-physnet: Learning the intuitive physics of non-rigid object deformations. In Proceedings of the 26th International Joint Conference on Artificial Intelligence, IJCAI-18, pp. 4958-4964, 2018.

[144] E. Yumer and N. J. Mitra. Learning semantic deformation flows with 3d convolutional networks. In ECCV, 2016.

[145] B. Deng, J.P. Lewis, T. Jeruzalski, G. Pons-Moll, G. Hinton, M. Norouzi, and A. Tagliasacchi. Neural articulated shape approximation. arXiv preprint arXiv:1912.03207, 2019.

[146] A. Jacobson, Z. Deng, L. Kavan, and J. Lewis. Skinning: Real-time shape deformation. In: ACM SIGGRAPH Courses, 2014.

[147] Y. Li, Y. Wang, Y. Yue, D. Xu, S.F. Chang and E. Grinspun. Model-Driven Feedforward Prediction for Manipulation of Deformable Objects In IEEE Transactions on Automation Science and Engineering, vol. 15, no. 4, pp. 1621-1638, October 2018, doi: 10.1109/TASE.2017.2766228.

[148] S. Orts-Escolano, U. V. More, J. Garcia-Rodriguez and M. Cazorla. Point cloud data filtering and down sampling using growing neural gas. The 2013 International Joint Conference on Neural Networks (IJCNN), pp. 1-8, August 2013.

[149] M. Martinetz, S. G. Berkovich and K. J. Schulten. Neural-gas network for vector quantization and its application to time-series prediction. IEEE Transactions on Neural Networks, vol. 4, no. 4, pp. 558–568, July 1993.

[150] J. Garcia-Rodriguez, S. Orts-Escolano, J. A. Serra-Perez, A. Jimeno-Morenilla, A. Garcia-Garcia, V. Morell and M. Cazorla. 3D model reconstruction using neural gas accelerated on GPU. Applied Soft Computing, vol. 32, pp. 87-100, 2015.

[151] A.-M. Cretu, E. M. Petriu and G. G. Patry. Neural-network-based models of 3-D objects for virtualized reality: a comparative study. IEEE Transactions on Instrumentation and Measurement, vol. 55, no. 1, pp. 99-111, 2006.

[152] J. Frey. Dynamical Deformation Network: A deep generative model for 3d voxel object deformation using Kalman variational auto-encoder. Bachelor's Thesis, KIT Department of Informatics Institute for Anthropomatics and Robotics (IAR), High Perfomance Humanoid Technologies Lab (H2T), Karlsruhe Institute of Technology, Germany, September 2018.

[153] L. Puig and K. Daniilidis. Monocular 3D tracking of deformable surfaces. IEEE International Conference on Robotics and Automation (ICRA), pp. 580-586, June 2016.

[154] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H. P. Seidel and S. Thrun. Performance capture from sparse multi-view video. ACM Transactions on Graphics (TOG), vol. 27, no. 3, pp. 1-10, August 2008.

[155] S. Caccamo, P. Güler, H. Kjellstrom, and D. Kragic. Active Perception and Modeling of Deformable Surfaces Using Gaussian Processes and Position-Based Dynamics. In Proceedings of the 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids), pp. 530-537, 2016.

[156] C. Elbrechter, R. Haschke and H. Ritter. Folding paper with anthropomorphic robot hands using real-time physics-based modeling. In Proceedings of the IEEE International Conference Humanoid Robots, pp. 210-215, Osaka, Japan, 29 November-1 December 2012.

[157] J. Park, W. Cho, S. Park, S. Kim, G. Ahn, M. Lee and G. Lee. Segmentation of 3D object in volume dataset using active deformable model. IEEE International Conference on Image Processing, pp. 4121-4124, September 2010.

[158] F. Conti, O. Khatib and C. Baur. Interactive rendering of deformable objects based on a filling sphere modeling approach. Accepted for presentation at the IEEE International Conference on Robotics and Automation, September 2003.

[159] Z. Li, L. Jin, J. Lang and E. Petriu. Dissection simulation of deformable objects using the extended finite element method. IEEE International Symposium on Haptic, Audio and Visual Environments and Games (HAVE) Proceedings, October 2014.

[160] T. W. Sederherg and S. R. Parry. Free-Form Deformation of Solid Geometric Models. Computer Graphics, vol. 20, no. 4, pp. 151-160, August 1986.

[161] I. Peterlık, M. Sedef, C. Basdogan and L. Matyska. Real-time visio-haptic interaction with static soft tissue models having geometric and material nonlinearity. Computers & Graphics, vol. 34, no. 1, pp. 43–54, February 2010.

[162] W. Yan, A. Vangipuram, P. Abbeel and L. Pinto. Learning predictive representations for deformable objects using contrastive estimation. arXiv preprint arXiv:2003.05436, 2020.

[163] A. C. M. T. G. Oliveira, R. Tori, W. Brito, J. dos Santos, H. H. Biscaro and F. L. S. Nunes, F.L.S. Simulation of soft tissue deformation: A new approach. Computer-Based Medical Systems (CBMS), IEEE 26th International Symposium on, pp. 17-22, June 2013.

[164] A. C. M. T. G. Oliveira, R. Tori, J. L. Bernardes, R. S. Torres, W. Brito and F. L. S. Nunes. Simulation of Deformation in Models of Human Organs Using Physical Parameters. Virtual and Augmented Reality (SVR), XVI Symposium on, pp. 277-286, May 2014.

[165] A. C. M. T. G. Oliveira, R. Tori, J. L. Bernardes, R. S. Torres, W. Brito and F. L. S. Nunes. Deformation Method Using Physical Parameters Composed of Different Tissue Structures. IEEE 27th International Symposium on Computer-Based Medical Systems, pp. 94-99, May 2014.

[166] F. Alambeigi, Z. Wang, R. Hegeman, Y.H. Liu, and M. Armand. A Robust Data-Driven Approach for Online Learning and Manipulation of Unmodeled 3-D Heterogeneous Compliant Objects. IEEE Robotics and Automation, vol. 3, no. 4, pp. 4140-4147, 2018.

[167] J. Schulman, A. Lee, J. Ho and P. Abbeel. Tracking deformable objects with point clouds. In Proceedings of the IEEE International Conference Robotics and Automation, Karlsruhe, Germany, 6–10 May 2013.

[168] J. Park, D. Mataxas, A. Young and L. Axel. Deformable models with parameter functions for cardiac motion analysis from tagged mri data. IEEE Transactions on Medical Imaging, vol. 15, no. 3, pp. 278-289, 1996.

[169] W.-S. Tang and K. C. Hui. Constraints Based Deformation. International Conference on Manufacturing Automation, pp. 87-93, December 2010.

[170] R. Yu, C. Russell, N. D. F. Campbell and L. Agapito. Direct, Dense, and Deformable: Template-Based Non-rigid 3D Reconstruction from RGB Video. IEEE International Conference on Computer Vision (ICCV), pp. 918-926, December 2015.

[171] H. Bin, T. Wen and C. Jintao. Virtual prototyping-based multibody systems dynamics analysis of offshore crane. The International Journal of Advanced Manufacturing Technology, vol. 75, no. 1, pp. 161-180, 2014.

[172] M. G. Mero and A. Susin. Deformable 3D Objects for a VR medical application. IEEE 617 Computers in Cardiology, no. 29, pp. 617−620, 2002.

[173] F. Hui, P. Payeur, and A.-M. Cretu. Visual Tracking of Deformation and Classification of Non-Rigid Objects with Robot Hand Probing. Robotics, vol. 6, no. 1, March 2017.

[174] A. Drimus, G. Kootstra, A. Bilberg, and D. Kragic. Design of a Flexible Tactile Sensor for Classification of Rigid and Deformable Objects. Robotics and Autonomous Systems, vol. 62, no. 1, pp. 3-15, 2014.

[175] P. Jiménez. Survey on Model-Based Manipulation Planning of Deformable Objects. Robotics and Computer-Integrated Manufacturing, vol. 28, no. 2, pp 154-163, 2012.

[176] J. Sanchez, J.A. Corrales, B.C. Bouzgarrou, and Y. Mezouar. Robotic Manipulation and Sensing of Deformable Objects in Domestic and Industrial Applications: A Survey. The International Journal of Robotics Research, vol. 37, no. 7, 2018.

[177] F. Nadon, A. Valencia, and P. Payeur. Multi-Modal Sensing and Robotic Manipulation of Non-Rigid Objects: A Survey. Robotics, vol. 7, no 4, pp. 74, November 2018.

[178] Skanect. Available online: http://skanect.occipital.com/ (accessed on 15 March 2017).

[179] Tera Term. Available online: https://ttssh2.osdn.jp/index.html.en (accessed on 15 March 2017).

[180] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, vol. 24, no. 6, pp. 381-395, 1981.

[181] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis Machine Intelligence, vol. 14, no. 2, pp. 239–256, 1992.

[182] K. Rzazewska and M. Luckner. 3D model reconstruction and evaluation using a collection of points extracted from the series of photographs. Proceedings of the 2014 Federated Conference on Computer Science and Information Systems, vol. 2, pp. 669–677, 2014.

[183] Meshmixer. Available online: http://www.meshmixer.com/ (accessed on 15 March 2017).

[184] CloudCompare−3D Point Cloud and Mesh Processing Software. Available online: http://www.danielgm.net/cc/ (accessed on 15 March 2017).

[185] M. Garland and P. S. Heckbert. Surface simplification using quadric error meshes. In Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH ’97), pp. 209–216, Los Angeles, CA, USA, 3–8 August 1997.

[186] B. Wang, L. Wu, K. Yin, U. Ascher, L. Liu and H. Huang. Deformation capture and modeling of soft objects, ACM Transactions on Graphics (TOG), vol. 34, no. 4, article no. 94, August 2015.

[187] B. Tawbe and A.-M. Cretu. Acquisition and Neural Network Prediction of 3D Deformable Object Shape Using a Kinect and a Force-Torque Sensor. Mariani S, ed. Sensors (Basel, Switzerland), vol. 17, no. 5:1083. doi:10.3390/s17051083, 2017.

[188] B. Tawbe and A.-M. Cretu. Data-Driven Representation of Soft Deformable Objects Based on Force-Torque Data and 3D Vision Measurements. In Proceedings of the 3rd International Electronic Conference on Sensors and Applications, Online, 15–30 November 2016; Sciforum Electronic Conference Series, vol. 3, p. E006, doi:10.3390/ecsa-3-E006, 2016.

[189] P. Payeur, P. Curtis and A.-M. Cretu. Computational Methods for Selective Acquisition of Depth Measurements in Machine Perception. IEEE International Conference on Systems, Man, and Cybernetic, pp. 876-881, Manchester, UK, 2013.

[190] D. Svozil, V. Kvasnicka and J. Pospichal. Introduction to multi-layer feedforward neural networks. Chemometrics and Intelligent Laboratory Systems, vol. 39, no. 1, pp. 43-62, 1997.

[191] M. Kazhdan, M. Bolitho and H. Hoppe. Poisson surface reconstruction. In Proceedings of the Eurographics Symposium on Geometry Processing, pp. 61-70, Sardinia, Italy, 26–28 June 2006.

[192] N. Aspert, D. Santa-Cruz and T. Ebrahimi. Mesh: Measuring Errors Between Surfaces Using The Hausdroff Distance. In Proc. of the IEEE International Conference in Multimedia and Expo (ICME), vol. 1, pp. 705-708, 2002.

[193] S. Kalatehjari and F. Yaghmaee. Using structural information for reduced reference image quality assessment. 2014 4th International Conference on Computer and Knowledge Engineering (ICCKE), pp. 537-541, 2014.

[194] A.-M. Cretu, E. M. Petriu and P. Payeur. Data acquisition and modeling of 3D deformable objects using neural networks. Proceedings of the 2009 IEEE International Conference on Systems, Man, and Cybernetics, pp. 3383-3388, 2009.

[195] Ansys academic software. Available online: https://www.ansys.com/ (accessed and used on 1 December 2019).

[196] Material data for simulation. Available online:https://digitallabs.edrmedeso.com/ansys-granta-materials-data-for-simulation/ (accessed and used on 7 December 2019).

[197] SpaceClaim Software. Available online: http://www.spaceclaim.com/ (accessed and used on 10 December 2019).

[198] DesignerModeler Software. Available online: https://www.ansys.com/services/training-center/fluids/introduction-to-ansys-designmodeler/ (accessed and used on 10 December 2019).

[199] Mechanical Software. Available online: https://www.ansys.com/products/structures/ansys-mechanical-enterprise/ (accessed and used on 15 December 2019).

[200] A. Pumarola, A. Agudo, L. Porzi, A. Sanfeliu, V. Lepetit, and F. Moreno-Noguer. Geometry-Aware Network for Non-rigid Shape Prediction from a Single View. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4681-4690, doi: 10.1109/CVPR.2018.00492, Salt Lake City, UT, 2018.

Annex A – Error measures after simplification

Table 4.2.1. Error measures and computing time for ball object after simplification.

Ball #	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object**
	Max/Mean/Rms		**Max/Mean/Rms**		
1	17.076/3.289/4.024	0.2945 (70.55%)	36.13/3.82/7.03	0.1437 (85.63%)	0.156 s
2	15.816/2.136/2.617	0.2504 (74.96%)	44.12/4.65/8.82	0.1202 (87.98%)	0.172 s
3	24.75/3.757/4.833	0.3246 (67.54%)	55.41/4.43/8.66	0.1534 (84.67%)	0.078 s
4	12.872/1.481/2.067	0.2787 (72.13%)	51.13/4.38/8.54	0.1347 (86.53%)	0.125 s
5	26.67/2.512/3.941	0.2905 (70.95%)	45.92/4.46/8.56	0.1769 (82.31%)	0.140 s
6	13.804/2.002/2.433	0.3078 (69.22%)	46.72/4.5/8.5	0.1529 (84.71%)	0.125 s
7	18.178/3.434/4.221	0.3125 (68.75%)	39.91/4.51/8.52	0.1564 (84.36%)	0.172 s
8	18.186/2.88/3.592	0.2512 (74.88%)	50.24/4.63/8.84	0.1195 (88.05%)	0.172 s
9	15.217/3.195/3.876	0.2677 (73.23%)	42.81/4.27/8.01	0.1197 (88.03%)	0.125 s
10	14.155/4.789/5.523	0.2839 (71.61%)	50.38/5.31/1.02	0.1293 (87.07%)	0.125 s
Average	**17.672/2.948/3.713**	**0.2862 (71.38%)**	**46.277/4.496/7.548**	**0.1407 (85.93%)**	**0.139 s**

Table 4.2.2. Error measures and computing time for cube object after simplification.

Cube #	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object**
	Max/Mean/Rms		**Max/Mean/Rms**		
1	21.443/1.134/4.061	0.2082 (79.18%)	46.88/3.79/7.52	0.0653 (93.47%)	0.016 s
2	11.213/0.45/1.757	0.2073 (79.27%)	46.59/3.52/7.19	0.0560 (94.40%)	0.046 s
3	14.427/0.643/2.445	0.2042 (79.58%)	34.12/2.94/6.09	0.0572 (94.28%)	0.031 s
4	9.798/0.443/1.675	0.2078 (79.22%)	40.56/2.85/6.08	0.0581 (94.19%)	0.063 s
5	13.187/0.568/2.19	0.2072 (79.23%)	44.29/3.56/7.08	0.0613 (93.87%)	0.031 s
6	8.623/0.379/1.473	0.2078 (79.22%)	43.32/3.55/7.05	0.0661 (93.39%)	0.063 s

7	12.303/0.552/2.12	0.2072 (79.26%)	47.7/3.29/7.01	0.0772 (92.28%)	0.047 s
8	14.029/0.633/2.397	0.2061 (79.39%)	38.58/3.07/6.12	0.0651 (93.49%)	0.031 s
Average	**13.128/0.600/2.265**	**0.2070 (79.30%)**	**42.755/3.3213/6.765**	**0.0633 (93.67%)**	**0.033 s**

Table 4.2.3. Error measures and computing time for sponge object after simplification.

Sponge #	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object**
	Max/Mean/Rms		**Max/Mean/Rms**		
1	4.699/0.122/0.597	0.1620 (83.80%)	43.11/3.47/7.26	0.1041 (89.59%)	0.078 s
2	4.312/0.115/0.563	0.1620 (83.80%)	44.09/3.01/6.22	0.0809 (91.91%)	0.109 s
3	3.4/0.095/0.448	0.1621 (83.79%)	43.11/3.45/7.36	0.1097 (89.03%)	0.078 s
4	2.714/0.89/0.387	0.1624 (83.76%)	42.11/2.97/6.01	0.1015 (89.85%)	0.031 s
5	4.174/0.118/0.562	0.1626 (83.74%)	49.78/2.94/5.74	0.0748 (92.52%)	0.031 s
6	4.545/0.125/0.6	0.1623 (83.77%)	37.41/3.05/5.77	0.0753 (92.47%)	0.078 s
7	4.718/0.131/0.63	0.1616 (83.84%)	41.5/3.05/6.34	0.0764 (92.36%)	0.063 s
8	4.142/0.11/0.534	0.1619 (83.58%)	38.66/3.33/6.4	0.0779 (92.21%)	0.109 s
Average	**4.088/0.113/0.540**	**0.1625 (83.7503%)**	**42.471/3.159/6.388**	**0.0876 (91.24%)**	**0.058 s**

Annex B – Error measures after clustering and sampling

Table 4.3.1. Error measures and computing time for ball object after clustering and sampling.

Ball #	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object**
	Max/Mean/Rms		**Max/Mean/Rms**		**Clustering/Sampling**
1	16.83/3.25/4.0	0.2129 (78.70%)	22.7/0.57/2.43	0.070 (93.01%)	0.081 s/0.029 s
2	16.012/2.23/2.78	0.17003 (82.99%)	21.97/0.32/2.49	0.0523 (94.78%)	0.080 s/0.027 s
3	24.75/3.70/4.80	0.2345 (76.55%)	28.22/1.19/2.26	0.0818 (91.82%)	0.083 s/0.026 s
4	12.872/1.47/2.20	0.1984 (80.16%)	24.3/1.42/2.44	0.0613 (93.87%)	0.078 s/0.028 s
5	26.67/2.64/4.23	0.1902 (80.98%)	28.88/0.49/2.29	0.0692 (93.08%)	0.079 s/0.026 s
6	13.793/2.06/2.53	0.2269 (77.31%)	23.91/1.01/2.08	0.0760 (92.40%)	0.082 s/0.112 s
7	18.132/3.6/4.41	0.2316 (76.84%)	27.69/0.66/2.1	0.0874 (90.26%)	0.081 s/0.027 s
8	18.168/3.00/3.8	0.17097 (82.90%)	23.97/1.25/2.2	0.0520 (94.80%)	0.095 s/0.030 s
9	15.151/3.51/4.19	0.1876 (81.24%)	28.09/1.15/2.37	0.0657 (93.43%)	0.087 s/0.029 s
10	14.093/4.56/5.31	0.2004 (79.96%)	27.55/0.25/2.41	0.0609 (93.91%)	0.085 s/0.032 s
Average	**17.647/3.00/3.8**	**0.2024 (79.76%)**	**25.73/0.83/2.31**	**0.0687 (93.14%)**	**0.083 s/0.037 s**

Table 4.3.2. Error measures and computing time for cube object after clustering and sampling.

Cube #	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object**
	Max/Mean/Rms		**Max/Mean/Rms**		**Clustering/Sampling**
1	21.41/0.73/3.25	0.2685 (73.15%)	38.71/2.73/6.43	0.052 (94.80%)	0.024 s/0.023 s
2	11.2/0.24/1.27	0.2657 (73.43%)	38.36/2.38/6.13	0.0597 (94.03%)	0.026 s/0.025 s
3	14.4/0.39/1.91	0.2709 (72.91%)	39.82/2.98/6.16	0.0507 (94.93%)	0.025 s/0.026 s

4	9.73/0.24/1.21	0.2735 (72.64%)	39.35/2.97/6.06	0.0527 (94.73%)	0.025 s/0.022 s
5	13.19/0.32/1.63	0.2644 (73.56%)	39.38/2.16/6.47	0.0551 (94.49%)	0.023 s/0.024 s
6	8.61/0.21/1.09	0.2755 (72.45%)	39.99/2.66/6.32	0.0542 (94.58%)	0.025 s/0.025 s
7	12.30/0.32/1.62	0.2717 (72.83%)	40.31/2.62/6.24	0.0576 (94.24%)	0.025 s/0.050 s
8	14.01/0.37/1.83	0.2649 (73.51%)	39.05/2.56/6.32	0.0554 (94.46%)	0.029 s/0.027 s
Average	**13.11/0.35/1.71**	**0.2694 (73.06%)**	**39.37/2.63/6.27**	**0.0547 (94.53%)**	**0.025 s/0.028 s**

Table 4.3.3. Error measures and computing time for sponge object after clustering and sampling.

Sponge #	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object**
	Max/Mean/Rms		**Max/Mean/Rms**		**Clustering/Sampling**
1	4.699/0.066/0.43	0.2326 (76.74%)	42.09/5.8/6.55	0.0931 (90.69%)	0.031 s/0.036 s
2	4.312/0.073/0.46	0.2268 (77.32%)	42.29/5.93/6.52	0.0941 (90.59%)	0.038 s/0.023 s
3	3.4/0.053/0.33	0.2424 (75.76%)	42.09/5.01/7.15	0.0916 (90.84%)	0.048 s/0.022 s
4	2.714/0.058/0.31	0.2349 (76.51%)	42.44/4.89/7.19	0.0942 (90.58%)	0.035 s/0.030 s
5	4.142/0.065/0.40	0.2294 (77.06%)	42.04/5.11/6.74	0.0941 (90.59%)	0.030 s/0.027 s
6	4.545/0.093/0.52	0.2436 (75.64%)	42.99/4.04/7.22	0.0943 (90.57%)	0.034 s/0.037 s
7	4.646/0.058/0.41	0.2399 (76.01%)	41.44/4.4/6.88	0.0941 (90.59%)	0.059 s/0.024 s
8	4.142/0.063/0.40	0.2269 (77.31%)	41.21/4.02/6.88	0.0933 (90.67%)	0.038 s/0.049 s
Average	**4.075/0.066/0.41**	**0.23457 (76.54%)**	**42.078/4.9/6.89**	**0.0936 (90.64%)**	**0.039 s/0.031 s**

Annex C – Error measures after neural gas

Table 4.4.1. Error measures and computing time for ball object after neural gas.

Ball #	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object/ Iteration**
	Max/Mean/Rms		**Max/Mean/Rms**		**Neural Gas**
1	53.3/1.5/2.4	0.0898 (91.02%)	11.6/0.2/1	0.0241 (97.59%)	35.512 s
2	53.3/1.5/2.5	0.0918 (90.82%)	14.2/0.2/0.8	0.0157 (98.43%)	27.795 s
3	53.3/1.3/2.2	0.0555 (94.45%)	10.3/0.1/0.6	0.0087 (99.13%)	28.045 s
4	56.3/1.3/2.2	0.0552 (94.48%)	10.6/0.1/0.6	0.0105 (98.95%)	17.741 s
5	53.3/1.5/2.4	0.0886 (91.14%)	11.2/0.1/0.8	0.0141 (98.59%)	21.543 s
6	56.3/1.5/2.4	0.0893 (91.07%)	11.2/0.1/0.8	0.0148 (98.52%)	25.902 s
7	53.3/1.4/2.4	0.0885 (91.15%)	11.2/0.2/0.8	0.015 (98.50%)	18.308 s
8	56.3/1.4/2.4	0.0884 (91.16%)	11.2/0.2/0.8	0.0162 (98.38%)	30.406 s
9	53.3/1.4/2.4	0.0932 (90.68%)	11.1/0.2/0.8	0.018 (98.197%)	30.484 s
10	53.3/1.4/2.4	0.0933 (90.67%)	11.1/0.2/0.8	0.0155 (98.45%)	23.923 s
Average	**54.2/1.42/2.37**	**0.083 (91.66%)**	**11.37/0.16/0.78**	**0.0155 (98.47%)**	**25.966 s**

Table 4.4.2. Error measures and computing time for cube object after neural gas.

Cube #	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object/Iteration**
	Max/Mean/Rms		**Max/Mean/Rms**		**Neural Gas**
1	21.1/2/3.6	0.1849 (81.51%)	29.5/0.3/1.5	0.0275 (97.25%)	5.076 s
2	27.9/2.1/3.9	0.1781 (82.19%)	25.5/0.2/1.4	0.009 (99.095%)	4.882 s
3	23.7/2.1/3.8	0.1801 (81.99%)	21.4/0.2/1.2	0.0131 (98.69%)	5.107 s
4	23.4/2/3.7	0.1889 (81.11%)	23.7/0.3/1.4	0.0174 (98.26%)	4.934 s
5	21.1/2/3.7	0.1851 (81.49%)	25.2/0.4/1.8	0.0343 (96.57%)	5.432 s
6	22.3/2/3.7	0.1876 (81.24%)	25.2/0.3/1.5	0.0379 (96.21%)	5.153 s

7	23.7/2.1/3.9	0.1817 (81.83%)	23.7/0.3/1.4	0.0173 (98.27%)	4.845 s
8	23.4/2.1/3.8	0.1817 (81.83%)	23.7/0.2/1.1	0.0139 (98.61%)	4.951 s
Average	**23.325/2.05/3.7625**	**0.1835 (81.65%)**	**26.16/0.30/1.51**	**0.0139 (97.679%)**	**5.047 s**

Table 4.4.3. Error measures and computing time for sponge object after neural gas.

Sponge #	**Overall Metro Error (e^{-3})**	**Overall Perceptual Error (Similarity %)**	**Metro Error in Deformed Area (e^{-5})**	**Perceptual Error (Similarity%) in Deformed Area**	**Computing Time/Object/Iteration**
	Max/Mean/Rms		**Max/Mean/Rms**		**Neural Gas**
1	32.1/2.4/4.4	0.1983 (80.17%)	32/0.8/2.6	0.0852 (91.48%)	7.753 s
2	35.7/2.7/5	0.1994 (80.06%)	37.8/0.8/2.9	0.0512 (94.88%)	7.692 s
3	37.1/2.7/4.9	0.1991 (80.09%)	37.9/0.9/3.1	0.0494 (95.07%)	7.155 s
4	43/2.9/5.6	0.2036 (79.64%)	40.1/1.3/3.8	0.1321 (86.79%)	7,224 s
5	35.9/2.5/4.5	0.1984 (80.16%)	37.9/0.9/3	0.079 (92.097%)	7.253 s
6	33.7/2.7/5	0.2028 (79.72%)	37.8/1.1/3.4	0.0978 (90.22%)	6.548 s
7	37.1/2.7/4.9	0.1977 (80.23%)	37.9/1/3.3	0.0649 (93.56%)	6.583 s
8	37.1/2.6/4.9	0.1991 (80.09%)	37.8/1/3.4	0.0872 (91.28%)	6.242 s
Average	**36.46/2.65/4.9**	**0.1991 (80.02%)**	**40.67/1.01/3.38**	**0.0872 (91.63%)**	**7.056 s**

Annex D - Ansys academic software simulator

Ansys academic software [195] is developed by ANSYS company, this company employs nearly 4,000 professionals and students, many of whom are expert M.S. and Ph.D.-level engineers in finite element analysis and design optimization. ANSYS has been recognized as one of the world's most innovative companies by prestigious publications such as Bloomberg Business Week and FORTUNE magazines. ANSYS Academic engineering simulation software is a free software used by thousands of universities around the world for researchers to solve complex engineering problems, and for students of third cycle to produce data for their master's or doctoral theses. ANSYS Academic engineering simulation software offers a complete solution for FEM. This solution goes through several steps. We start with the choice of materials, then draw the 3D object, meshing the object, then exert forces on the object, finally build and export the deformed mesh.

Annex E - Choice of materials

We can choose among hundreds of materials such as rubber, foam, steel, silicone and plastic [196]. We can choose two or more type of materials for one object; for example, we can choose rubber with plastic. We have the option to group different materials together. We can decide the percentage of each material in the object. Example, we can choose an object with 40% rubber, 20% foam and 40% plastic. Each material has several characteristics that we can choose by ourselves. With each material we have different parameters and different values of these parameters. Some parameters are defined by default independently of the type of material of the object. For example, the stiffness, the damping and the mass matrices. By using some commands under Ansys, we can export these matrices easily and use it in MATLAB.

Annex F - Geometry and creation of 3D objects

Ansys academic software is connected to several other software to ensure a complete solution. Among these software, we have the two software SpaceClaim [197] and DesignerModeler [198] that we can use to create the geometry of any kind of 3D objects. We can easily create 3D objects as cube, sphere and cylinder. We can create large and small complex objects by specifying their parameters. For example, we can create a large

sphere with a radius of 9 cm, then modify the radius to 3mm to create a small sphere. We can design a complex geometry by ourselves and then transform it easily into a 3D object. We can import any kind of geometry or mesh from other software and transform it to 3D object. After creating the object by SpaceClaim or DesignerModeler we can send it automatically to Ansys and associate it with any materials.