Tactile Sensing with Artificial Skin – Simulation and Deep Learning

Master Research Project - Semester 2

Abel de Wit, Diego Di Benedetto, Guillaume Franzoni Darnois, Julia Hindel, Rik Dijkstra, Simone Grassi

Supervised by:
Rico Möckel, Lucas Dahl

Department of Data Science and Knowledge Engineering
Maastricht University
Netherlands

Contents

Human skin is made of an intricate network of sensory inputs useful not only to detect temperature or other potentially hazardous contacts with the external environment, but also for a better object manipulation. Robots can exploit these peculiarities using artificial skins composed by a silicon like layers with a dense sensor grid. Usually the sensors pattern completely covers the skin surface making it particularly expensive. The main focus of this project is to develop a classification system, able to predict some characteristics of the object with the skin is interacting and if it can be potentially dangerous. These machine learning models usually require a big amount of training data, difficult and time consuming to obtain from online experiments, so one of the main problem that will be faced is the simulation and generation of a sensors reading data-set more as realistic as possible.

1 Introduction

1.1 Motivation

Skin, the largest organ in humans, protects the underlying muscles, bones, ligaments and internal organs. The skin is used to regulate temperature, keep water in the body, and protect it from external environments. The skin has an intricate network of sensory inputs to alert of sudden temperature changes and potentially hazardous contacts such as sharp objects or hard impacts. The skin consists of several layers: The Epidermis (the outer layer), Dermis, and Hypodermis.[1] These layers have their own properties and sensory inputs, which makes the skin a complex system. This inspires researchers to recreate flexible and stretchable skin properties in the form of electronic or artificial skin. With the development of new materials and smaller sensors with better sensing capabilities, researchers are getting closer to re-creating a realistic skin.

Another objective is to correctly interpret what the sensors of this artificial skin perceive. To classify the objects that touch the artificial skin, the sensors need to be interpreted correctly to prevent misfires. An understanding of their relative location is needed to understand movement along the skin. Once this is achieved, the next step is to have a (pre-trained) model that can classify whether the interaction with the skin can be deemed harmful. This research will focus on these two aspects of simulation and classification. The results could open possibilities for research into prostheses both as body parts or as part of an exoskeleton.

1.2 State of the Art

1.2.1 Simulation

Simulation plays a significant role in many aspects of computer science as it allows for offline learning, opening the possibilities of quick tweaks to otherwise complex real-world systems. A simulation, which has a suitable mathematical model, allows the researchers to create a vast database of information in a relatively short time. For a simulation to be efficient, it is necessary to approximate natural systems, especially complex ones like skin. To replicate such an intricate and complex organ, the technique of Finite Element Analysis (FEA) is often adopted. FEA follows a *divide et impera* approach. It divides the system in smaller and simpler objects called Finite Elements on which the computations are performed. Subsequently, the algorithm recombines the elements to obtain an alteration to the entire system. [2]

The skin, however, is extremely challenging to model as it can be deformed by stretching, contracting or even pushing the Epidermis, and it might require a more precise approach depending on the demanded accuracy of the simulation. J. Wu et al.[3], model the internal structures of a fingertip very precisely, which allows them to simulate the deformation of a finger as it presses on a surface. This deformation is particularly interesting for our simulation of artificial skin material as it will deform with increased pressure and, therefore, trigger sensors that might not be directly underneath the point of contact.

The extensive research, which has been conducted on modelling a skin structure, allows us to implement a model with tension lines, material density, viscosity, and elasticity, for each layer. Additionally, the forces applied to the outer layer of the skin can be translated to the forces that the sensors will experience within the skin with the outlined models.

Moreover, some simulation models for muscles have been created [4]. This direction of research is particularly important since the skin not only interacts with external objects, but also with the internal structure of the human body. One particular research is especially interesting for this project. In [5], B. Angles et al. were able to simulate the contraction of muscles in real-time. This research differs from other publications because the introduced methodology can reproduce simulations with low computation time, which enables a generation of a vast database, even with a more sophisticated model. On top of that, the simulation takes into account the expansion of the muscles during the contraction, which is key to provide the skin with internal interactions.

In Section 2.1, we further discuss our step-by-step approach and implemented techniques.

1.2.2 Classification

The classification task of tactile input is based on pressure readings from sensors placed on the surface of the artificial skin. The classification itself is executed by machine learning models that are able to make predictions from the available data. The main difficulty lies in the simulation and suitable representation of the sensor devices.

Tactile sensors have been applied with increasing frequency in robotics in recent years [6, 7]. Contact data processing is largely based on the exploitation of tactile images, where each pixel corresponds to a detected pressure value. The generation of these images is performed when tactile sensors are integrated on a flat surface, and their placement follows a non-uniform distribution [8]. Albini et al.[9, 10] showed the process of the image generation originated from a non-regular large-area distribution of tactile sensors in their publications. The process was divided into two steps: the representation of the sensor placement is exploited using a Delaunay triangulation[11], which defines a list of topological relations to create a 2D mesh that simulates the robot surface; Successively, when contact occurs, the measurements of the sensors involved are mapped to a 3D heat map of the pressure distribution. In their works, Albini et al. created a state-of-the-art method to obtain images from a distributed tactile sensor surface and applied classical machine learning models to process the images. However, this methodology cannot be applied in our scenario without significant adjustments. In fact, the artificial skin upon which this research is based can only contain a limited amount of sensors. Therefore, an estimation of pressure readings using triangulation is insufficient to generate a complete tactile image.

A possible approach that does not rely on image representations in sensor processing and classification is utilising graphs as a data structure. Garcia et al.[12] faced the problem of predicting grasp stability using a robotic arm called Shadow hand [13]. In their approach, they used a graph to represent the measured values of the sensors because it can constitute the spatial distribution and local connectivity of these sensing points more accurately. In this data structure, each node encodes the detected pressure. Successively, a prediction is performed using a graph convolutional neural network (GCN), which has been studied by Kipf [14]. He showed experiments on several network data-sets suggesting that the proposed GCN model can encode both graph structure and node features for supervised and semi-supervised classification.

1.3 Research questions

The prospects of our project are to contribute to the development of an artificial soft skin controller. Our main focus is finding a computationally and time-efficient way to generate a data-set using an offline simulation that can be used to train a touch classifier. From this perspective, the main questions that are addressed in our research are:

1. How can we use Finite Element analysis techniques to model a soft artificial skin with a good similarity to the real prototype?
Finite element analysis is a powerful approach to the material-physics simulations. We want to apply it to model a soft silicone-like sensory layer to obtain measurements that are considered similar enough to the real artificial skin, according to our project specification.

2. How can we use an offline simulation to generate a touch data-set that generalises well on real word sensor data?
An extensive data-set is required to classify the interaction-type of contact with the artificial skin effectively. Its generation using the prototype is costly in terms of time and resources. For this reason, we want to investigate the exploitation of an accurate simulation.

3. How can we develop a neural network classifier that works on a graph or a touch image representation of a low-density sensor pattern?
We face the problem that the artificial skin can only have a relatively low-density sensor distribution. Moreover, we have to consider a slight inaccuracy in the position of the sensors. Therefore, we want to investigate the strengths and weaknesses of five promising techniques in terms of performance to find out if one technique outperforms the other.

2 Concepts and approach

An overview of the individual steps of our approach is illustrated in Figure 1. In this section, it is discussed what concepts we use and how these differ from previous work. Therefore, the approach for the simulation and touch classification tasks are addressed in detail in sections 2.1 and 2.2, respectively.

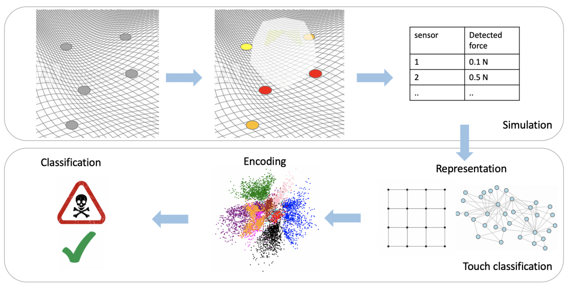


Figure 1: Overview of our approach

This project requires us to face two main challenges that differ from the state-of-the-art techniques:

1. We cannot assume a fixed sensor pattern, nor can we assume that the surface is completely and densely covered by sensors. This increases the uncertainty of predicting the location of any point, which is based on the three known closest neighbours.

2. The artificial skin is not a solid surface, but a soft and deformable layer in which the sensors are embedded. This increases the uncertainty of the sensors' positions and forces us to consider the propagation of forces applied to the skin. We know the physical properties of our skin, but we cannot assume that they are homogeneous.

2.1 Simulation

There are several benefits that come with a realistic simulation, one of them being the ability to perform experiments at little to no cost. Furthermore, data generation can be automated, sparing valuable time and resources.

To classify touches on the arm by a machine learning model, we need a relatively large amount of data. This data is not readily available, let alone for any specific prototype. However, it can be simulated to approach reality. Different types of interactions with the skin can be coded as input. Its output will be labelled accordingly, which allows for the use of supervised learning techniques in our classification tasks.

In order to accomplish the tasks that come with this project, it is necessary to implement a module capable of computing the interaction between mesh objects while replicating accurate physics dynamics. This is achieved using Finite Element Analysis, which consist of studying phenomena simulated through the Finite Element Method technique. This method is generally used in engineering modelling to solve systems of partial differential equations numerically.[2] For this method to work efficiently, it is necessary to build meshes of the objects involved in the simulation. The goal is to discretise the surfaces and mass of these objects to make such computations feasible.

2.1.1 Naive simulation

Before creating a FEM simulation, it is important to build a structure that allows such computation to be integrated in the simulation process. For this reason, in an early stage of the project, we develop a simulation framework that handles all the tasks related to input generation, simulation, and data-set creation. The method used is considered to be naive since it does not follow any accurate physics law.

3 Steps Framework:

- *Input Generation*: For this phase, each combination of the 4 base classes is created and an input is created based on this This ensures the creation of a balanced data-set where all possible combinations of classes are explored. The data-set creation is separately discussed in detail in a following paragraph. To create the input objects for every combination, the values of velocity, time span, force, shape and initial position are randomly selected. Those objects are described by two components: an image that represents the shape, and the weights which define how the image affects the skin as pressure.
- *Simulation*: Initially, we follow a very fast, yet less accurate approach to produce data in order to test different architectures and analyse the performances of different approaches of the classification task. The way the simulation is performed is by sliding the input image ("shape") over a pixel map, which simulates a discretized sheet of skin. During this process, the pressure is applied by multiplying its intensity with the normalized value of the image. Then, only the values observed on the pixels corresponding to the sensors are saved with the goal to create a data-set.
- *Data-set Creation*: To create the data-set, we use three main types of data. First, the metadata of the simulation containing the id of the simulation and for each row the frame count which suggests the temporal position of the specific reading. Additionally, the labels are appended for classification, encoded as four binary digits:
  - *big vs. small*: includes the division of gestures by the size of the object they have been performed

with. Objects that have a contact surface of over 30% of the skin are considered *big*.

- *dynamic vs. static*: gestures are classified as *dynamic* if the object slides across the surface of skin, i.e. the movement across the vertical axis is greater than 0. Otherwise, the movement is denoted as *static* as the object approaches the skin in a straight line that is perpendicular to the sheet of skin.
- *press vs. tap*: concerns the duration of contact of the object with the artificial skin. If constant pressure is applied for more than three consecutive frames, the gesture is considered a *press*.
- *dangerous vs. safe*: The last binary classification examines the comfort or danger of the experienced interaction. For this purpose, we define a touch as dangerous if it contains a sequence of frames in which at least one sensor detects a pressure of over 90.5 kPa. This threshold is reasoned with a medical research conducted by Jesperson et. al [15]. In this study, they examine the pressure pain threshold with cuff pressure algometry to determine differences in pain sensitivity. We, therefore, use the average pressure pain threshold of healthy participants as our guideline. Nevertheless, the authors want to highlight that perceived pain is very subjective and the used threshold is highly biased to the participants of the experiment.

Finally, the sensor readings are a list of n values where n represents the number of sensors used for the simulation. The readings are saved starting from the upper left corner of the sheet of skin and are recorded in a right and downwards direction.

Visualization: Since most of the data is handled using matrices, we also use such data structure to create animations showing the progression of a simulation over time. The data shown is the force readings of each sensor and if desired also the complete overview of the simulation. The representation can be produced in both 2D, by creating a simple GIF showing an "aerial view", and 3D, which gives a more precise idea of the intensity of the readings.

2.1.2 Finite Element Modelling

To create a more realistic simulation of the silicone skin, several applications and implementations of Finite Element Modelling have been tried. The most accessible program that was easily integrated with our existing naive simulation was PyNite [16]. With PyNite it is possible to create a three-dimensional model consisting of nodes, beams, and plates. A node is a point in 3D-space which has no properties. Nodes can be connected with either plates or beams (denoted as 'Member' in the module). A plate is a connection between four nodes, while two nodes can be connected with a beam.

Using these components, a layer can be constructed consisting of nodes in a grid. These nodes will then be connected by plates to model the top, bottom, and cross-sections of the skin. A stack of these layers can then be connected to each other through beams, allowing the transfer of force throughout multiple layers.

Another important aspect of Finite Element Modelling is the fact that it needs boundary conditions to allow the model to converge to a stable state. If there are no restrictions and force is applied to the model, it would continue in infinite space, or until the maximum iterations of 30 calculations are reached (as defined by PyNite). Hence, PyNite allows setting constraints on each node.

Constraints can be restrictions in movements or rotations in any of the three dimensions. A node that has restricted movement, but not rotation is called 'Pinned', while a node that has both movement and rotation restrained is called 'Fixed'.

To simulate the sheet of skin sitting on a tabletop, we assumed that the nodes that 'touch' the table, i.e. are on the bottom, will be fixed. For every other layer, the options exist to restrict any of the above-mentioned movements. From pre-existing experiments with the real silicone skin in the laboratory, it was concluded that the most important motion is in the vertical ('z') direction, as sensors that are just a fraction outside of the pressing area would not fire anymore. To incorporate this behaviour in our FEM, we restricted all nodes above the tabletop layer to only move along the vertical direction, and any rotation was not restricted. This approach limits the type of displacements to the vertical axis as well, but unrestricting the nodes should allow for shear force to be applied to the model as well.

When the model is constructed and has correctly defined supports, the next step is to apply the types of touches that will constitute the touch database. To achieve inputs of irregular shapes such as blurred circles, or the top of a fist, gray-scale images are used as input, with black pixels representing no force, and completely white pixels as the maximum force that is specified.

The image is first normalized so the sum of all pixels in the image correspond to the force that is passed to the input method, as can be seen in formula 1.

$$M_{norm} = \frac{M}{Sum(M)} \tag{1}$$

To decide what total force has to be applied with the image, we need to compare it to the pascal value, since this measurement is used to determine if the gesture is dangerous or safe. The formula used for this conversion can be seen below in Equation 2.

$$F = \frac{t_{Pa} * A_{pixel}}{max(M)} \tag{2}$$

Then, the image is sub-divided in the resolution of the mesh size to create sections of the image for each plate in the upper layer of the skin. The intensity of each subsection is translated to a force that is applied to the corresponding plate. This allows for images to be of different sizes and changes in the coarseness or resolution of the mesh. A finer mesh size allows for more detailed input at the cost of additional computational time.

After the model is fully configured and has the correct loads applied, it is analysed until it converges to a steady state. When the steady state is reached or the maximum number of iterations, the model knows the forces that each node endured, and the displacement that occurred. The results can be visualized as can be seen in Figure 2 and Figure 3

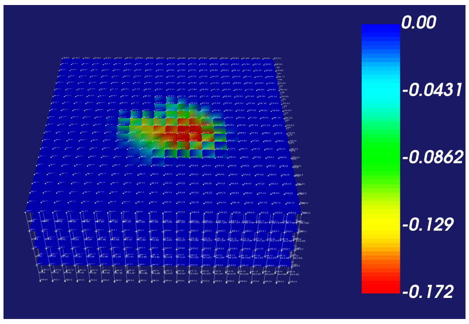


Figure 2: FEM of Hand - Top

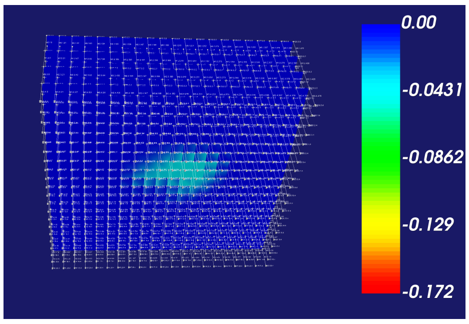


Figure 3: FEM of Hand - Bottom

From this information an overview of the displacements can be made. This overview allows to access the displacement at the points where the sensors are supposed to be. From these sensor displacements a data-set can be created.

2.1.3 Material Properties

To accurately simulate the artificial skin using our Finite Element Model, we model it according to the prototype of a robotic arm. This prototype is created using molded silicone of various types. The various silicone materials used are manufactured by the American company *Smooth-On*, and are a part of their *EcoFlex* product-line. *EcoFlex 00-30* was selected to represent the skin of the robot, *EcoFlex DragonSkin 10 Medium* is used for the muscle fiber and *EcoFlex Gel* for the fat. The reasoning and research that have led to the selection of these materials is not part of this project.

The material properties needed to run the Finite Element Model are listed in Table 1.

Symbol	**Title**	**Unit**
E	Young's Modulus	ksi — Pa
ν	Poisson Ratio	$n.a.$
G	Shear Modulus	ksi — Pa
Iy	Second Moment of Inertia	in^4 — m^4
Ix	Second Moment of Inertia	in^4 — m^4
J	Polar Moment of Inertia	in^4 — m^4
A	Cross-Sectional Area	in^2 — m^2

Table 1: Material properties required for Finite Element Modeling

The Young's Modulus (E) is a measure for elasticity, which is measured by pulling on either side of a beam of material and measuring the expansion and contraction in the axial and lateral directions, as well as the force applied. The formula can be seen below, in formula 3.

$$E = \frac{\sigma}{\epsilon} = \frac{F/A}{\Delta L/L} \tag{3}$$

The variable σ is the unaxial stress and ϵ represents the strain. These are calculated using the applied force F in N (Newton), the cross sectional area A in m^2 and the initial length and difference in length L and ΔL. This gives the Young's Modulus in Pa, which is later converted to ksi for the use in the Finite Element Model, which takes all values in imperial notation.

The Poisson's Ratio (ν) is a measure of the Poisson Effect [17], which is the effect of materials contracting in the lateral directions when expanding in the axial direction. The Poisson's Ratio is quite simply the transverse strain $\Delta L'$ over the axial strain ΔL as seen in formula 4. The Poisson's Ratio is independent from the force applied, but can be more accurately measured at higher forces in the elastic range [18].

$$\nu = \frac{\Delta L'}{\Delta L} \tag{4}$$

The Shear Modulus is similar to the Young's Modulus, but it measures the shear elasticity. This is achieved by pulling only the top of the material sideways with some force. It is slightly more cumbersome to measure, but for isotropic materials it can be estimated with the formula 5.

$$G = \frac{E}{2(1+\nu)} \tag{5}$$

The remaining values are not related to the material but rather to the shape they take on in our simulation. The moments of inertia and the cross-sectional area are defined by formula's 6, 7, 8 and 9. In these formula's, w stands for the width and h for the height. It is important to note that these equations are different for other shapes. The presented formulas only apply for rectangles.

$$Iy = \frac{wh^3}{12} \tag{6}$$

$$Ix = \frac{hw^3}{12} \tag{7}$$

$$J = Iy + Ix \tag{8}$$

$$A = wh \tag{9}$$

A lot of papers have been considered for gathering the correct values, but the references will be restricted to the papers used. The reported values differ between papers because of the mixture of silicone under varying circumstances. Furthermore, several papers discussed their own composite materials, which have significantly different values and are, therefore, useless for this research. To determine the values to be used, a small scale experiment was done to test the Young's Modulus and Poisson's Ratio for the *EcoFlex 0030* and the *EcoFlex Dragonskin 10 Medium.* The *EcoFlex gel* yielded unreliable results, and because of limited time we restricted our FEM to model only parts of skin without fat. Coincidentally, in the current prototype, there are no sensors above any substantial amount of fat. If these values can be provided for the *EcoFlex gel*, it can be implemented with relative ease. Our experiment is by no means the most accurate, hence we do not use its values for calculations. However, the experiments show that the materials on the prototype are most similar to the results of Fouillet el al. [19]. The Poisson's Ratio is not mentioned in this paper and retrieved from the work of An [20], note that this for the *EcoFlex 0030* only. The Poisson's Ratio for the *EcoFlex Dragonskin 10 Medium* was taken from our experiment, because no records could be found online, within reasonable time. In table 2 the material properties that are used can be seen.

Symbol	**0030**	**10 Medium**	**Unit**
E	0.02466	0.08122	ksi
ν	0.499	0.54	$n.a.$
G	0.008224	0.02637	ksi
Iy	0.0833	0.0833	in^4
Ix	0.0833	0.0833	in^4
J	0.1666	0.1666	in^4
A	1	1	in^2

Table 2: Material properties required for Finite Element Modeling

2.1.4 Real world calibration

To make our data as realistic as possible, and the model more robust, we have to account for noise and imperfections in the data. Furthermore, we have to account for the force absorption by the silicone. Then there are some physical limitations which the sensors are subject to, such as the maximum measurement range of around $10N$.

The raw data is read into a dataframe, containing id numbers, frame numbers, labels, input force and displacement information. This displacement data is then used to calculate the resulting force.

$$F_i = \frac{D_s}{D_t} * F_{in,i} \tag{10}$$

Where F_i is the force in N at sensor i, D_s the displacement of the sensor, D_t the total displacement of that coordinate (all layers above and below the sensor included) and $F_{in,i}$ is the input force at the surface above sensor i.

After calculating the remaining force at each sensor, clipping is applied, which simulates the clipping of the sensors themselves at 10N. Then noise and offset is added according to a normal distribution. The offset matrix stays the same, whereas the noise matrix changes every frame.

For future research, the noise distributions and the exact dimensions of the prototype should be calibrated with the post-processing, as it is now merely an approximation.

2.2 Classification

2.2.1 Representation of sensor pattern

The human body is made up of more and less sensitive parts. Moreover, the skin covers materials that have different physical properties, such as their density and elasticity. Consequently, it is necessary to allow for a non-uniform distribution of sensors as an input of the classification model. Our first representation is based on images, where activated sensors have the value of the applied pressure while non-activated sensors are equated to no-information regions which have the value 0. The proposed image-based approach has proven successful in previous research [9]. We use images of size 64x64 as a standard configuration but also analyse the effects of using smaller and larger image sizes. Additionally, it is tested if applying a Fourier transform or nearest, linear or cubic interpolation between the sensors to create an image representation of the sensors improves the result. The Fourier transform is used to make regular patterns in images more visible by changing to a frequency domain. We, therefore, hypothesize that a representation of the structure of activated sensors in the frequency domain is distinctly different which will help the network for the classification task. Interpolation is a technique which approximates values between a set of discrete ones. Nearest interpolation assigns pixels the value of the closest neighbors, instead linear interpolation creates straight lines between known sensor values. Cubic interpolation calculates the derivative within two points using a cubic polynomial function. A visualisation of the the representations are shown in Figure 4. Our second approach encodes the displacement of sensors in a graph data structure where the nodes represent the sensors values and the edges encode the distance between them. Therefore, the nodes can include a single sensor reading (later denoted as sequence graph) or a fixed window of sensor readings over time as a vector (temporal graph). Each node is linked to the others by bidirected edges which values can encode the distance or other physical information such as the material density. Consequently, it is possible to effectively represent any possible sensors placement directly in the data structure.

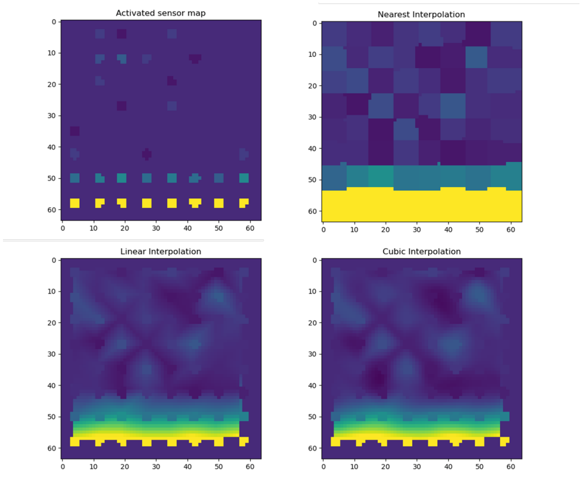


Figure 4: Overview of interpolation techniques

2.2.2 Encoding and Classification

In the following, we present the multiple tested architectures. The approaches can be subdivided into a temporal connected graph, late and early fusion. For all approaches, the last layer is represented by four nodes with a sigmoid activation to represent the four binary classifications.

Temporal connected Graph Classification The representation of an entire gesture is completely based on a single graph which means that each frame is represented by a sub-graph that encodes the spatial nearness of the sensors. The temporal evolution of these sub-graphs is performed by linking the nodes of the previous and following sub-graph in the sequence with a directed edge. The result is a big graph with thousands of nodes and exponential number of edges. Consequently, we decide to perform the linking through time for a single sensor. The approach consists of learning an effective embedding of the entire graph using convolution followed by a dense layer for classification.

Early Fusion This approach stacks multiple frames at an early stage in the network over the channel dimension. The resulting 3D representation is processed with

2D convolutional layers with a kernel size of same depth. Thereby, parameters are not shared over different frames of sensor maps and the temporal ordering of sensor measurements is only taken into account in the first layer. For video classification this approach has proven less successful than late fusion [21]. However, we argue that due to the simplicity of our input data, this approach is also promising.

- *Temporal Graph Classification* The architecture is based on a temporal graph which incorporates a fixed window of sensor readings over time as a vector. We then use graph convolutional layers to create an embedding for each node, that represents its features and neighbouring components. Then a compressed representation is extracted by applying max pooling. Afterwards, dense layers are added to classify the instances in the introduced four binary classifications.

- *Temporal Image Classification* For this approach, the first convolutional layer converts the entire sequence of sensor mappings directly into an internal embedding by using the fixed window size as the number of channels. Afterwards, multiple convolutions and dense layers are applied.

Late Fusion This approach is composed of two parts: an encoder and a classifier. The first takes care of converting the sensor representation to an embedding. Subsequently, the classifier will contain recurrent or pooling elements to perform the binary classifications as introduced above. Late fusion is a common approach in video classification [22].

- *Sequence graph embedding* For this approach, an embedding/feature map is computed for each frame of the sequence. Successively, the embeddings are fed into a recurrent neural network to process the temporal variation of the touch and classify the entire sequence. This method models the classification without the need for a fixed size sequence for the gesture, which makes it appropriate for real time classification. We test pre-trained graph embeddings as well as train our own graph convolutions to create a representation for the individual sensor graphs.

  To evaluate whether the embeddings of the pre-trained models are suitable for the classification task, the samples are clustered with the t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm after the dimensions of the representation are reduced with the Principal Component Analysis (PCA). It is recommended to apply PCA before the t-SNE algorithm to improve the results. PCA is a unsupervised, deterministic technique for linear data dimensionality reduction which filters the dimensions is which the data is maximum differential [23]. The process is based on transforming the original set of vectors that are correlated to each other to a new set which is known as principal component. This set represents the global structure of the data in a lower dimension. On the other hand, t-SNE is based on a similar idea and is commonly used for data exploration and visualization of high-dimensional data. The algorithm is non-linear and the main difference to PCA is that it preserves the local structure of the data by minimizing the Kullback–Leibler divergence between the two distributions with respect to the locations of the points in the map[24].

  Along with visual inspections, the retrieved representations are further analysed by applying the Support Vector Machine (SVM) algorithm to classify whether the embeddings can be correctly labeled. SVM is a popular machine learning algorithm for classification [23].

- *Image embedding* When converting the sensor grid to an image, we can use convolutional neural networks that have shown very good performance in different scenarios. Therefore, we test pre-trained image networks in addition to our own convolutional model. As a pre-trained network, we have selected the state-of-the-art network EfficientNetB0 due to its superior results in common image classification tasks [25].

- *Classification of Image and Sequence graph embeddings* We combine the late fusion approach with the attention mechanism, in order to contextualize the temporal information. For sequence graph embeddings, global attention pooling is used to extract a representation of the nodes for each frame graph. The resulting output is a sequence that is fed into a recurrent layer. The context vector is then fed into dense layers, although it could be biased towards last frames, is commonly used for classification purposes instead of using all the outputs of the sequence merged by a pooling layer. We combine the late fusion approach with the attention mechanism, in order to contextualize the temporal information. For images, self-attention and recurrent layers are used to process the embeddings with respect to time.

3 Experiments

3.1 Simulation

3.1.1 Naive v.s. FEM

To give an idea of the improvement given by the FEM simulation we show the difference in the plot of the most active sensor during simulations in Figure 10. This, other than showing a completely different type of activity, also highlights how the units used are closer to the ones utilized in the LAB on the arm. For this reason, it is very difficult, if not impossible, to perform a meaningful comparison between the two graphs. What can be analysed is the difference in the graph, having a more accurate representation for time and force leads to a better generation of data.

3.1.2 Simulation v.s. Real-world

In order to validate the Finite Element Model, comparisons between virtual sensor readings and real-world ones need to be made. To achieve this, we plot the final data that has been post-processed with offsets, noise and clipping, for classes that are similar to experiments that were conducted on the real skin in the laboratory. The class that is used is the circle with blur, which corresponded the most with the rounded head of the force gauge that was used on the real skin. With this, each sample in the created data-set is taken and its most active sensor is plotted with the corresponding force. A subset of the resulting images can be seen in Figure 11. This was done to create comparisons with the plots of the actual sensor readings, which can be seen in Figure 12. By comparing the images, it can be noted that the model seems like a good fit for the problem at hand. In the plots of the real sensors it can be observed that the force that is applied to the outer layer of the skin is mainly absorbed by the silicone both above and below the sensor. This behaviour is replicated in the plots of the FEM.

The ratio between the applied force and the force that is registered in the sensor is however not perfectly matched between the simulation and the real-world experiments. An applied force of 40 Newton is registered by the sensor as almost 2 Newton, which translates to around 5% of the force that is applied. In the Finite Element Model, paired with the post-processing of the displacements, the total applied force is at most 1000 Newton. This force is absorbed and the resulting sensor reading amounts to almost 10 Newton. This ratio is not a perfect match with the real-world data, but does show that the inner workings of a piece of silicone embedded with sensors can be modelled quite accurately.

3.2 Classification

In the following, the performed experiments are outlined for all five proposed architectures. For preliminary evaluations, we used the CoST: Corpus of Social Touch data-set [26]. The data includes 7805 gesture captures performed by humans on a robotic arm. We used a sub-section of the data which only incorporated gestures of a grab and hit with various strengths and aimed to differentiate those gestures. The final evaluation is based on the outlined simulated data, firstly without the FEM process (2350 gestures) and lately using the methodology for more realistic values (4038 gestures). However, due to time constraints, it is not possible to extensively fine-tune our models on the data including FEM calculations and as being more complex, we notice a decrease in accuracy in all models. For the below approaches, the Adam optimizer with a learning rate of 0.001 is utilised.

3.2.1 Temporal connected Graph Classification

We use the Deep Graph Library (DGL)[27] to process the graph representation. This framework provides an efficient directed graph data structure with tensors as features, graph data loader with auto batching and framework-specific neural network modules. As outlined in 2.2, the first approach encodes both spatial and temporal relationships in its edges. Despite using a shallow architecture with only one convolutional and dense layer, the high computational complexity showed to be problematic. The training of five epochs required around six hours of processing time. Additionally, it is not possible to load a sub-part of the graph, so the complete structure has to be loaded at once. Furthermore, creating and destroying the entire graph structure slows down the training drastically compared to the approach used in the Temporal Graph Classification. Moreover, due to the limited number of parameters of the simple architecture, the network is not able to generalise on the data resulting in a binary accuracy of 13% on the CoST data-set. Consequently, we argue that this representation is not suitable for the faced problem and further attention is directed towards the other four approaches.

3.2.2 Temporal Graph Classification

For this approach, we create a single graph where the features of the nodes encode the sensor values over time as a vector. We start with an easy architecture with a convolutional layer, a pooling and two dense layers. In order to understand if the embedding dimension has the same effect as in a CNN encoder, where a bottleneck in the encoder forces learning of more meaningful features, we tested different dimensions observing the opposite correlation. This is caused by a convolution that takes into account features from the neighbouring nodes too. Moreover, we test max pooling and average pooling and more complex strategies such as global attention pooling. The pooling layer is the key for this strategy, because the convolution phase outputs an embedding for each node, that takes into account the features of the current node and the neighbouring ones, but we need a meaningful representation of the entire graph. Average pooling shows to be less effective because of the large number of inactive sensors that affect the result, for the same reason max pooling provides the best results. This simple network proves to be able to classify the data accurately. However, when increasing the depth of the network, it tends to easily overfit. Additionally, we test the activation functions ReLU, ELU, and SiLU whereas the last one has a significant impact on the achieved accuracy. Consequently, SiLU activation is utilised in all the hidden layers of the network due to its severe benefits [28]. Furthermore, we apply transformer modules. However, these structures require a significant quantity of data, resulting in strong underfitting in our scenario. This final architecture variant has 145,774 parameters which allows to train it in 68 seconds per epoch. We conclude from our tests that an

easy network is able to reach the better results than more complex or structured ones, because it avoids overfitting and is able to learn with the amount of data we are able to provide. The best network exploits a convolutional encoder with 2 layers that creates a 500-dim embedding followed by max pooling, the hidden representation is then feed into two dense layers, with Swish activation function, and an output with sigmoid.
Data without FEM: the simpler version of the simulated data achieves a final overall accuracy amount to 85% after 70 epochs. The architecture generalizes sufficiently, despite the spatial classification (big/small and dangerous/safe) are less accurate.
Data with FEM: the overall accuracy drops to 73% and the per-class accuracy is more similar. Therefore, we reason that the previous observed bias towards a better classification on temporal relations (dynamic/static and press/tap) has been dependent on the data. We reason the behaviour of worse performance with the padding of the advanced data: in the simple simulation the process for a gesture is very similar whereas the advanced simulation includes padding before or after a recorded gesture to approximate a continuous sequence of sensor readings. As this model learns unique weights for frames in the sequence, it cannot generalise well when the peak of the gesture is reached at a different frame number.

3.2.3 Image Classification with Early Fusion

This approach treats frames through time as channels. Therefore, 2D convolutional layers in addition to max pooling, flattening and dense layers are applied. Even with a simple configuration this architecture seems to overfit quickly hence why dropout is employed. The best configuration has two convolutional layers followed by max pooling and spatial dropout. Afterwards, the data is flattened and passed into two dense layers with 32 and 4 nodes respectively. This model has 209,844 parameters and allows training in 57 seconds per epoch.
Data without FEM: this configuration achieves a binary accuracy of 84% for the simple simulation without the FEM model after 20 epochs. Afterwards, the model starts to overfit, however reducing the complexity resulted in direct underfitting.
Data with FEM: the model performs poorly on the advanced simulation reaching a binary accuracy of 66% after additional fine-tuning. A similar observation has already been made for the temporal graph classification, hence why we reason that this is a weakness of Early Fusion models.

3.2.4 Sequence Graph Classification

The embeddings for the sequence graph model are firstly computed with pre-trained models such as Graph2Vec[29], Invariant Graph Embedding(IGE)[30] and GL2Vec[31]. The results of t-SNE show that the tested pre-trained graph embeddings mostly emphasize in the morphology of the graph instead of the values of the nodes. The visualization of the feature space for three graphs belonging to different classes is shown in Figure 5, which evidences our assumption. Therefore, using pre-trained graph embeddings is not purposeful as the robotic arm has a fixed absolute sensor displacement. Consequently, an increased focus is set on computing feature maps with graph convolution.

The resulting architecture is composed of two graph convolution layers resulting in a vector of 128 dimensions per node. The feature map of each graph is then pooled with the global attention pooling mechanism and fed into a recurrent layer. Finally, two dense layers are added. Furthermore, different variants are tested by ensuring a symmetric structure within the initial graph convolution and dense layers. The effects on the results when adding layers or increasing the dimensionality of the features map are analysed. When increasing the embedding size to 512, the network does not generalise on the data. Lower dimensions resulted in an increased underfitting.

To incorporate the temporal domain, we test average and max pooling, convolution along the temporal dimension as well as recurrent layers such as LSTM and GRU. The binary accuracy is compared for each approach. Using GRU resulted in the best performance when paired with global attention pooling for the graph feature maps. GRU and LSTM show similar performances, however, GRU is preferred due to its lower computational complexity [32]. In comparison, employing the Set2Set[33] mechanism does not improve the accuracy. Moreover, layer and instance normalization are tested but this extension does not show any noticeable improvement in the final score. The presented architecture has 105,621 parameters and one epoch takes 750 seconds.
Data without FEM: using the simpler version of the simulated data, the final accuracy amounts to 82.9% after 70 epochs which take roughly 540 seconds per epoch. The model generalize on the data sufficiently well without the need for a particular regularization technique.
Data with FEM: this configuration achieves a binary classification accuracy of 75.1% after 20 epochs. As mentioned, this architecture is remarkably slow and a single epoch takes 800 seconds on average, making it difficult to extensively fine-tune it.

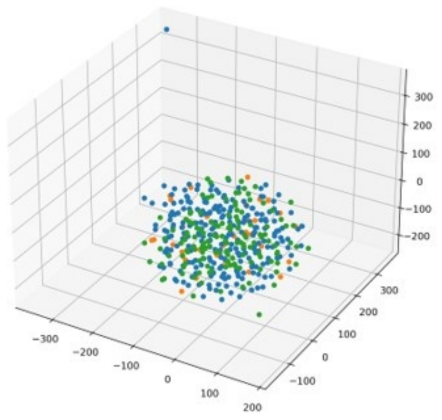
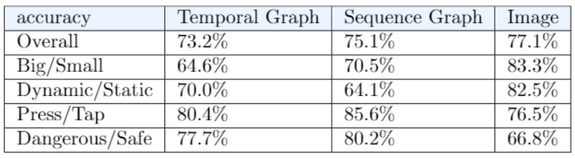


Figure 5: Visualization of the sub-space graph feature

3.2.5 Image Classification with Late Fusion

For this approach, an embedding is constructed for each sensor map. Afterwards, the temporal domain is incorporated with pooling or a recurrent layer. In a first test, the pre-trained EfficientNetB0 is used to create an embedding for the images individually. As this model expects three channels as an input, each sensor map is tripled. Unfortunately, first tests indicate that this model is not suitable for the outlined classification problem. When combined with global max pooling or fine-tuned LSTM layers, the model is highly biased towards one class and always predicts the same class. Consequently, the authors have developed a simpler convolutional architecture by alternating convolutional, max pooling and spatial dropout layers. For the temporal domain, using self-attention in combination with one GRU layer results in the best performance. Global max pooling and LSTM with attention only show a slightly lower performance (roughly 3%), indicating that these layers are similarly suitable. In summary, the presented architecture has 1,634,341 parameters. For all three data-sets, applying a Fourier transform decreases the performance in comparison to the previously shown image map. For the advanced simulation, the binary accuracy remains constant at 38% after 20 epochs. Additionally, using a combination of interpolations for the three channels does not affect the achieved accuracy positively. However, computing these different representations increases the training time per epoch from 74 seconds to 450 seconds per epoch. Consequently, it is decided to fed the raw data three times to the model. Additionally, it is estimated what effects different image sizes have on the network. When doubling the image size to 128x128, the training duration increases to 730 seconds per epoch while the number of parameters is quadrupled. A positive effect on the achieved performance is not observed. In comparison, a reduction of the image size to 48x48 results in

a stagnation of the achieved binary accuracy at 50%.
Data without FEM: the model achieves an accuracy of 81% with this data-set after 20 epochs.
Data with FEM: for this data a binary accuracy of 77% is reached after 20 epochs and one epoch amounts to 74 seconds of processing time. The model is slightly goes into overfit in the end. We conclude that this model is very robust as it shows the least drop in performance between the two different data-sets. The final results are displayed below:



accuracy	Temporal Graph	Sequence Graph	Image
Overall	73.2%	75.1%	77.1%
Big/Small	64.6%	70.5%	83.3%
Dynamic/Static	70.0%	64.1%	82.5%
Press/Tap	80.4%	85.6%	76.5%
Dangerous/Safe	77.7%	80.2%	66.8%

Figure 6: results of the best performing three models

In the following, we compare the various approaches and give indications about the classification. Our experiments proved that images and graphs can encode temporal and spatial relationship and have the ability to generalize on the data. Moreover, both representations have the possibility to represent every possible 2D sensors placement. The table in Figure 6 shows good and uniform results. This is proven by the analysis of the missclassified samples, which are correlated within the different models as shown in Figure 7, where it is compared how the size of the object is wrongly classified with different velocity in the gesture. Considering our problem definition, the biggest disadvantages of all approaches is the impossibility to find a suitable pre-trained model. The training time is strongly influenced by the dimension of the representations: when using graph data structure with dense sensors placement, the number of edges grows exponentially. Also for images, our experiments proved that doubling the resolution results in a tenfold of the training time.

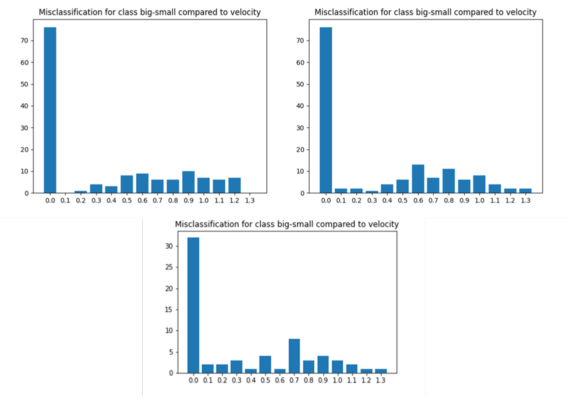


Figure 7: Miss-classification vs interaction speed, in order for the Temporal Graph, Sequence Graph and Image Later fusion approaches

In the scenario of the early fusion approach with graph data structure, we observe that the graph convolution does include the temporal features for the calculation. On the other hand, when employing images we reason that the convolution is applied channel-wise individually, decreasing the overall accuracy considerably. In general, the window size needs to be fixed for the application as the dimension of the weight matrices is directly influenced by the input features. Consequently, when the window changes, previously trained models cannot be adapted and transfer learning is not possible. Additionally, for the Temporal graph classification, the macro structure that defines the neighbouring relationship needs to be constant over time, otherwise the execution is heavily slowed down.

Late fusion is considerably slower than the other approaches for both images and graphs. This issue arises as each convolution is applied individually on each frame before applying the temporal layer. However, this design removes every constraint on the window dimension, bringing to the maximal representation power. The main disadvantage is in the depth of the network and the recurrent network complexity which results in a slower training time.

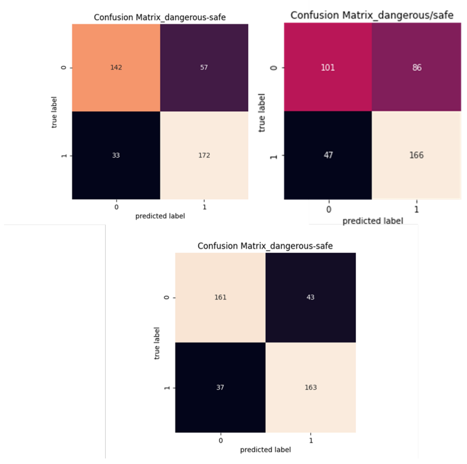


Figure 8: Dangerous/Safe confusion matrices, in order for the Temporal Graph, Sequence Graph and Image Later fusion approaches

Considering the application which our research is focused on, the most important class is Dangerous/Safe, because despite the heuristic classification is the more meaningful for the real world experiments that will follow. The best approach reaches an accuracy of 80.2%. When examining the confusion matrices of all models, we notice that the falsely classified samples are mostly safe gestures (labeled as 0) which are classified as dangerous. The confusion matrices for the models are showun in Figure 8 We argue that this missclassification is acceptable to some extent as the robot arm needs to have a high recall on harmful interactions.

4 Discussion

4.1 Simulation

Although the FEM is a very high precision technique some limitations are imposed by the library used and the missing information about materials. In specific, there are no complete and well-documented libraries in Python to make use of the FEM completely, which leads us to consider other languages as better suited options for the simulation part of the project, although this has been investigated in Java but not other programming languages. Regarding the material limitations, the specifics were not available on the product used. This lead to the usage of approximated or incorrect values as mentioned in Section 2.1.3 Therefore, using FEM improves the naive implementation a lot and allows us to have direct references in terms of forces and distances to the real-world experiments performed in the LAB.

For future research, it would be valuable to investigate 3D modeling of FEM. We achieved the implementation of a 3D model (Figure 9) but were unable to map the forces accordingly. Therefore, we do not use it for data creation. Furthermore, shear force could be simulated as it is unrealistic that only perpendicular forces interact with the arm. Tilted sensors could be placed to possibly be able to distinguish between the direction of the force.

Lastly, with better computers and possibly GPU acceleration a more detailed FEM model could be implemented, which results in more accurate results.

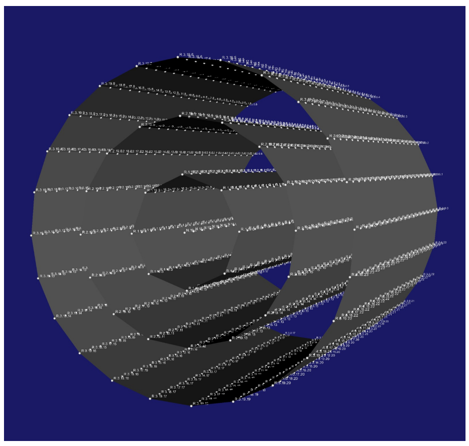


Figure 9: Concept of Arm Model

4.2 Classification

In summary, we observe that image and graph representations are suitable for the problem faced due to their capability in modelling non-uniform distribution of values. In comparison to images, graphs are more flexible as additional information can be added to the edges. Furthermore, this data type can also be directly used for 3D simulations, whereas the image approach has difficulties to represent such a complex sensor map. However, models based on images showed to be a lot faster for training and predictions, making them more suitable for real-time applications. The presented models based on late fusion approaches outperformed architectures based on early fusion in all cases. We argue this behaviour with the different structure of the gesture which favors shared weights across the sequences in the form of recurrent layers.

The artificial skin has properties that are remarkably difficult to handle such as elasticity and dilation. However, it is possible to solve this problem if the relative positions are estimated during simulation, making it possible to create a new sensor map or adjacency matrix for each frame.

Finally, the sliding window approach must be managed in a real word scenario. Therefore, we need to face the problem of classify a window representing a small portion of a particular gesture. Our models proved to be effective despite the fact that not all sequences provide meaningful information. In a real word scenario, the majority of the sequences would be without an interaction, so a suitable approach is to include an additional exclusive class which recognizes if the material is touched or not. This could be implemented as a simple threshold or a more sophisticated approach where the loss function is extended to add a 5th, exclusive binary classification.

In the end, our three approaches showed to have balanced per-class accuracy, symptoms of a good generalization and equilibrium in the feature representation weight, but they look at the spatial and time relationship in a very different way making them more effective in a class or another. For this reason, considering a fusion of the models including a voting scheme to further increase the performances should be considered.

As another possible extension of our research, we encourage to test different sensor patterns with the proposed classification models. Thereby, it's also an interesting direction to estimate a lower bound on the number of sensors that still allow a meaningful classification. In this way, it's possible to respond to our project constraint of hardware requirement reduction.

5 Conclusion

To conclude, we can use Finite Element Analysis to calculate the displacement at any given depth. Using this displacement, we can approximate the remaining force at any depth in the arm. It is a promising technique that, with some fine-tuning, can model reality with great accuracy. A drawback is the computational cost, as with the current implementation data creation takes about 4 hours for 2000 experiments. A potential solution for this could be to implement a Finite Element Model with only the displacement calculations.

By examining interactions with the prototype we can generate data without having the option to calibrate with the prototype, and create data fully offline. Using FEM, clipping and noise we can mimic the behaviour of the sensors in the prototype. Even though, at the moment, it does not correspond, this seems to be caused by a lack of interaction with the prototype and is very feasible within a reasonable amount of time.

The final goal of the project is to create a classifier that correctly predicts the interactions' classes exploiting the data created by the simulation. During this research, different approaches were tested and discussed, we faced the problem of study an efficient data structure that correctly resemble the non-uniform distribution of skin perceptions.

We found that both graphs and images are suitable representations, which enable to use convolution to extract meaningful features. With using architectures based on early and late fusion, the sequence of frames can be correctly classified. Unfortunately, using pre-trained models to create single frames embedding did not prove to be promising due to the singularity of our research problem. However, when relying on a self-trained convolutional network, notable results can be produced. In summary, graphs have a higher potential as a representation due to their flexibility and models based on late fusion with a recurrent layer can achieve better results.

6 Appendix

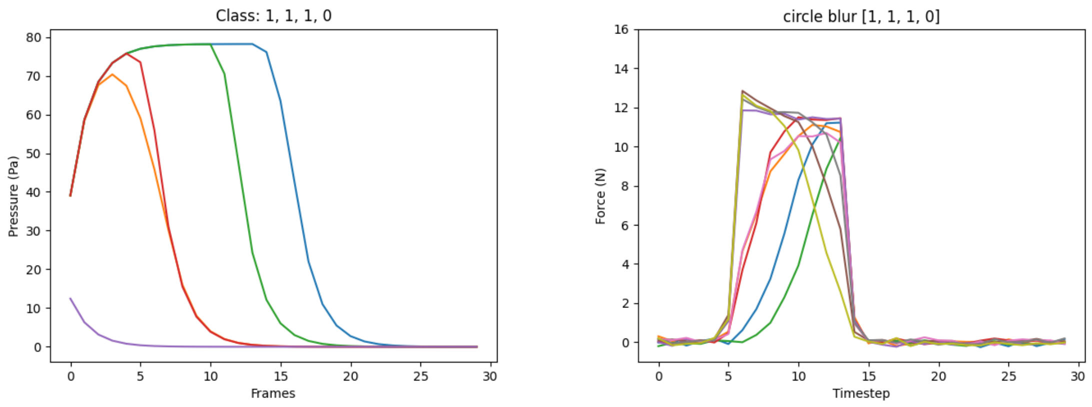


Figure 10: FEM vs Naive Simulation

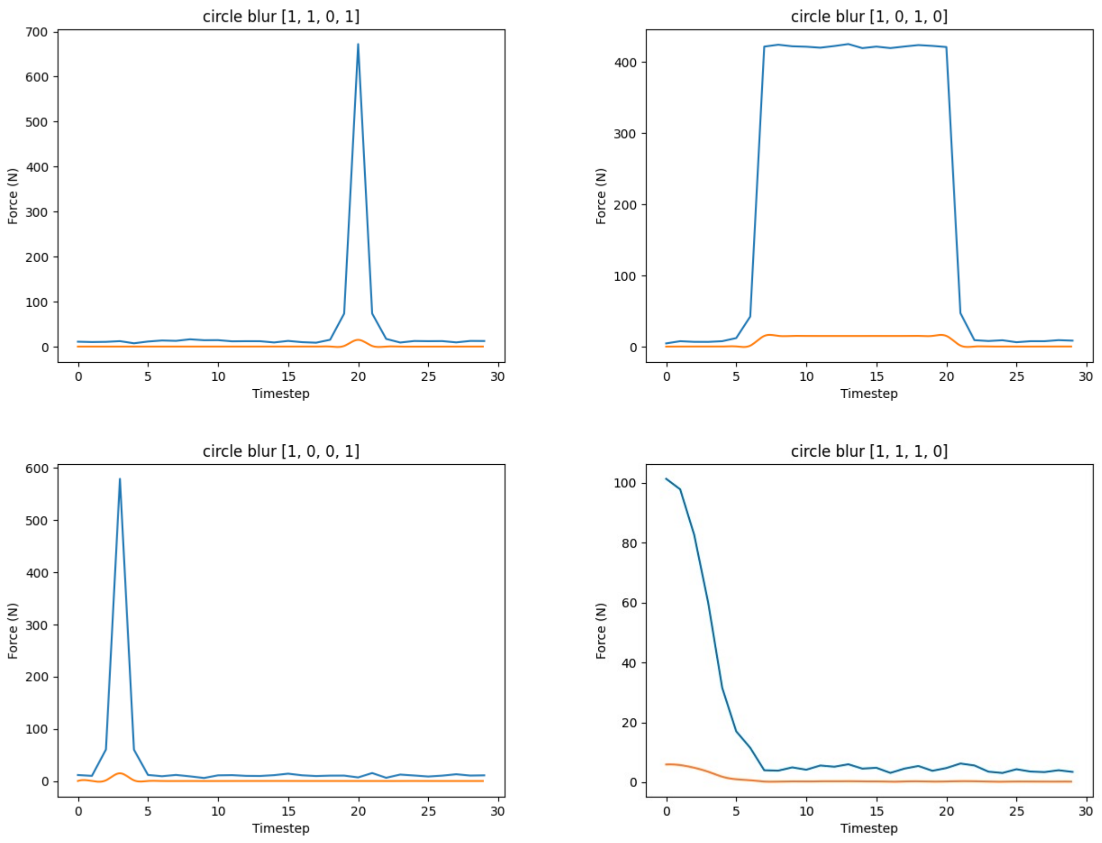


Figure 11: FEM Sensor readings, Blue: Total Applied Force on Skin, Orange: Sensor reading

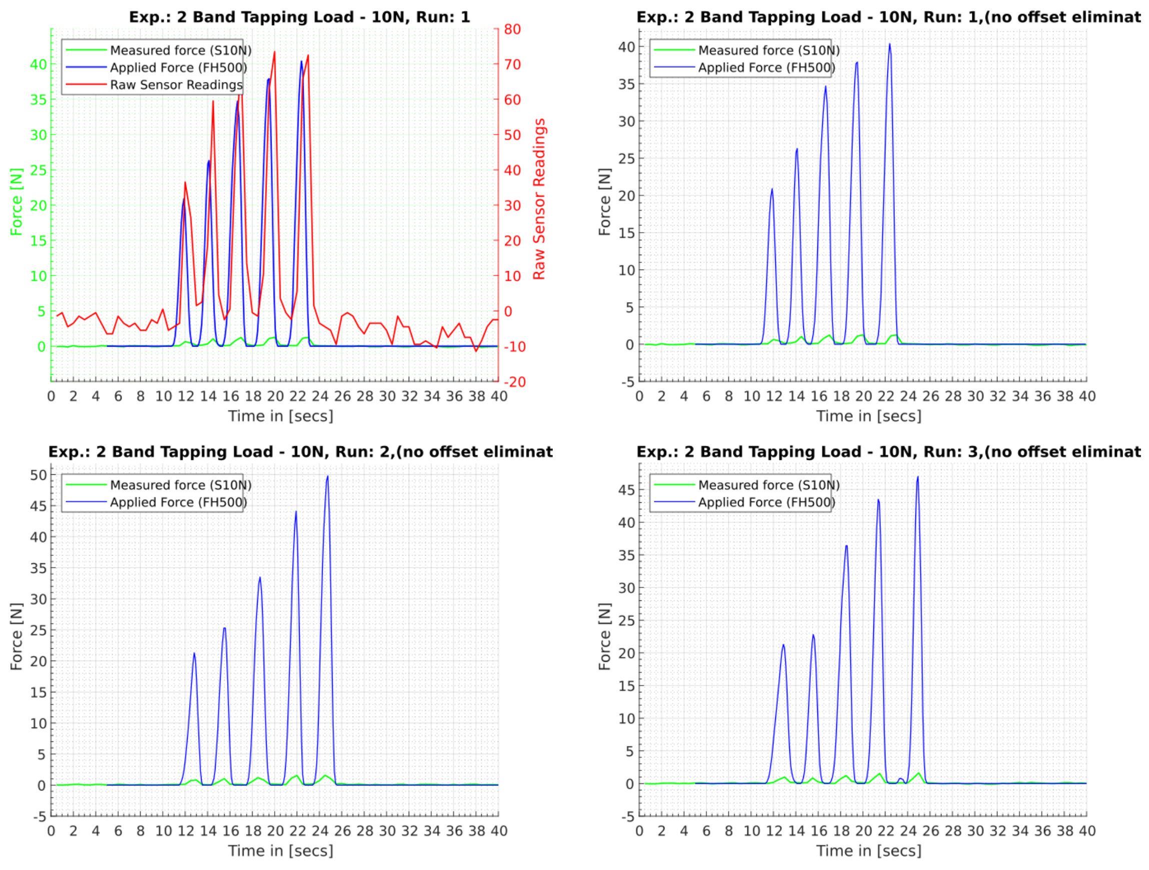


Figure 12: Real Sensor readings - Provided by L. Dahl

References

[1] CliniMed, "Structure and function of the skin." `https://www.clinimed.co.uk/wound-care/wound-essentials/structure-and-function-of-the-skin`. Accessed: 2021-03-10.

[2] W. F. J. Larrabee, "A finite element model of skin deformation. i. biomechanics of skin and soft tissue: a review.," *The Laryngoscope*, pp. 399–405, 1986.

[3] J. Wu, R. Dong, S. Rakheja, A. Schopper, and W. Smutz, "A structural fingertip model for simulating of the biomechanics of tactile sensation," *Medical Engineering & Physics*, vol. 26, no. 2, pp. 165–175, 2004.

[4] S. Lee, M. Park, K. Lee, and J. Lee, "Scalable muscle-actuated human simulation and control," *ACM Trans. Graph.*, vol. 38, July 2019.

[5] B. Angles, D. Rebain, M. Macklin, B. Wyvill, L. Barthe, J. Lewis, J. von der Pahlen, S. Izadi, J. Valentin, S. Bouaziz, and A. Tagliasacchi, "Viper: Volume invariant position-based elastic rods," 2019.

[6] D. Silvera-Tawil, D. Rye, and M. Velonaki, "Artificial skin and tactile sensing for socially interactive robots: A review," *Robotics and Autonomous Systems*, vol. 63, pp. 230–243, 2015.

[7] S. Hirai *et al.*, "A novel model for assessing sliding mechanics and tactile sensation of human-like fingertips during slip action," *Robotics and Autonomous Systems*, vol. 63, pp. 253–267, 2015.

[8] B. S. Zapata-Impata, P. Gil, and F. Torres, "Non-matrix tactile sensors: How can be exploited their local connectivity for predicting grasp stability?," *arXiv preprint arXiv:1809.05551*, 2018.

[9] A. Albini and G. Cannata, "Tactile images generation from contacts involving adjacent robot links," in *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pp. 306–312, IEEE, 2018.

[10] A. Albini and G. Cannata, "Pressure distribution classification and segmentation of human hands in contact with the robot body," *The International Journal of Robotics Research*, vol. 39, no. 6, pp. 668–687, 2020.

[11] P. Su and R. L. S. Drysdale, "A comparison of sequential delaunay triangulation algorithms," *Computational Geometry*, vol. 7, no. 5-6, pp. 361–385, 1997.

[12] A. Garcia-Garcia, B. S. Zapata-Impata, S. Orts-Escolano, P. Gil, and J. Garcia-Rodriguez, "Tactilegcn: A graph convolutional network for predicting grasp stability with tactile sensors," in *2019 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, IEEE, 2019.

[13] P. Tuffield and H. Elias, "The shadow robot mimics human actions," *Industrial Robot: An International Journal*, 2003.

[14] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[15] A. Jespersen, K. Amris, T. Graven-Nielsen, L. Arendt-Nielsen, E. M. Bartels, S. Torp-Pedersen, H. Bliddal, and B. Danneskiold-Samsoe, "Assessment of Pressure-Pain Thresholds and Central Sensitization of Pain in Lateral Epicondylalgia," *Pain Medicine*, vol. 14, pp. 297–304, 03 2013.

[16] 'JWock82', "Pynite: A 3d structural engineering finite element library for python.."

[17] Z. D. Jastrzebski, "Nature and properties of engineering materials," 1 1976.

[18] D. Rees, *Basic engineering plasticity: an introduction with engineering and manufacturing applications*. Elsevier, 2012.

[19] Y. Fouillet, C. Parent, G. Gropplero, L. Davoust, J. L. Achard, F. Revol-Cavalier, and N. Verplanck, "Stretchable material for microfluidic applications," in *Multidisciplinary Digital Publishing Institute Proceedings*, vol. 1, p. 501, 2017.

[20] S. An, D. J. Kang, and A. L. Yarin, "A blister-like soft nano-textured thermo-pneumatic actuator as an artificial muscle," *Nanoscale*, vol. 10, no. 35, pp. 16591–16600, 2018.

[21] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1725–1732, 2014.

[22] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici, "Beyond short snippets: Deep networks for video classification," *CoRR*, vol. abs/1503.08909, 2015.

[23] A. Géron, *Hands-On Machine Learning with Scikit-Learn and TensorFlow - Concepts, Tools, and Techniques to Build Intelligent Systems.* Sebastopol: "O'Reilly Media, Inc.", 2017.

[24] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[25] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR, 09–15 Jun 2019.

[26] M. Jung, "Corpus of Social Touch (CoST)," 6 2016.

[27] "Deep graph library tutorials and documentation." `https://docs.dgl.ai/en/0.6.x/index.html`. Accessed: 2021-06-18.

[28] P. Ramachandran, B. Zoph, and Q. Le V., "Swish: A self-gatered activation function," *arXiv preprint arXiv:1710.05941*, 2017.

[29] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal, "graph2vec: Learning distributed representations of graphs," *arXiv preprint arXiv:1707.05005*, 2017.

[30] A. Galland and M. Lelarge, "Invariant embedding for graph classification," in *ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data*, 2019.

[31] H. Chen and H. Koga, "Gl2vec: Graph embedding enriched by line graphs with edge features," in *Neural Information Processing* (T. Gedeon, K. W. Wong, and M. Lee, eds.), (Cham), pp. 3–14, Springer International Publishing, 2019.

[32] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[33] O. Vinyals, S. Bengio, and M. Kudlur, "Order matters: Sequence to sequence for sets," *arXiv preprint arXiv:1511.06391*, 2015.