Ecole Centrale de Nantes
University of Twente

Human Media Interaction
Engineering report

Designing a social touch skin for a humanoid robot avatar

Antoine Lorentz

Supervisor
Gwenn Englebienne
Camille Sallaberry

October 4, 2022

Contents

1 Thanks

Thanks to Gwenn for all the time he gave me, for all the fascinating discussions that interested me so much and for the ideas that always pushed me forward.

Thanks to Camille for his support during my internship, for all the services he gave me and the nice moments we shared.

Thanks to all the HMI department for their warm welcome and for all the games of foosball (I'm quite proud to have ended up winning a few matches at the end of my internship)

2 Introduction

The project presented is a research and prototyping project whose context is teleoperation with a humanoid robot avatar. Its aim is to design a touch-sensitive sensor that can cover the robot like a "skin".

Touch can be seen from two perspectives, the action of *touching* or the action of *being touched*. And the difference is as written above, the use of the active or passive voice, that is to say the fact of being an actor or not of this touch. And in the context of touching, this distinction makes a big difference. This difference is on the side of the *intention*.

For example, when we touch an object, we try to acquire information about its shape, its temperature, its roughness or its softness. Or when we touch a person, we want to convey information or support an emotion (attracting someone's attention, comforting them, etc.).

But being touched means being touched *by someone*. When we are touched, someone is trying to convey information or emotion to us, and we then interpret that touch according to the context of the interaction.

But what about touch in the field of robotics?

A robot can use touch to locate itself, to characterise a shape. But it becomes harder to talk about touch in a human-robot interaction because it involves the notion of intention for the touching robot and in the other case it requires the robot to have a strong capacity to interpret the touch.

But in teleoperation, the robot avatar becomes only a medium and the interaction is basically between the robot operator and the person in front of the robot. It is then possible to talk about a social touch through a robot.

This project explores this precise part: a social touch through a humanoid robot avatar enabled by a touch-sensitive sensor.

The objective of the project is therefore to make a state of the art of the technology related to touch, to choose one adapted to social touch and to develop it in the perspective of an integration on a robot. The development of this technology involves the physical construction of the sensor and the implementation of a software system from the low-level programming of the microcontroller to the processing and interpretation of the signal on a PC.

The project is part of two dynamics. The first is the academic interest in this topic at the University of Twente, more precisely the Human Media Interaction department, and the second is the ANA Avatar XPRIZE world robotics competition. This competition "aims to create an avatar system that can transport human presence to a remote location in real time"[1]. The University of Twente is working with the Dutch team i-Botics in this competition and social touch is one of the components of the competition.

[1] https://www.xprize.org/prizes/avatar

3 Presentation of the University of Twente

The University of Twente is a public university located in Enschede, the Netherlands. The university is ranked among the top 170 universities in the world in several central rankings. In addition, the technical university is ranked as the best technical university in the Netherlands in the Keuzegids Universiteiten, the main national university ranking. Under the umbrella of 4TU, the University of Twente collaborates with the Technical University of Delft, the Technical University of Eindhoven and Wageningen University and Research Centre, and is a partner in the European Consortium of Innovative Universities (ECIU).

The University of Twente offers a wide range of Bachelor's, Master's and PhD programmes: 22 Bachelor's programmes and 31 Master's programmes are taught in English.

Figure 1: Campus of the University of Twente

The Human-Media Interaction (HMI) group conducts research on multimodal interaction: from brain-computer interfaces to social robots. It is a multidisciplinary group in which computer science meets social science to study, design and evaluate new forms of human-computer interaction. The HMI group is strongly involved in the MSc. programme "Interaction Technology" and the BSc. programme "Creative Technology".

4 Literature review

In order to develop a touch-sensitive sensor, the first step is to turn to our own touch sensor, i.e. our skin. Because the definition of touch comes from the experience we have.

According to [3], the skin has sensory receptors that inform the brain about the characteristics of the object being touched. These characteristics are texture, weight, temperature and whether the overall sensation is pleasant or produces pain. The neurobiological process behind touch is the transformation of physical deformations into electrical signals proportional to pressure. This transformation is carried out by mechanoreceptors. Also, there are different types of mechanoreceptors sensitive to different physical quantities (pressure, movement, vibration, heat, pain and pleasure). A good example to understand the use of motion-sensitive mechanoreceptors is the bottle held in the hand but slipping because the grip is not strong enough. Pressure-sensitive mechanoreceptors can't tell the bottle is slipping since the pressure of the grip doesn't evolve.

With regard to the shape of an object, it is the elasticity of the skin that enables it to form a mirror image of the object and thus to deduce its shape.

Historically, the first touch-sensitive sensors were capacitive and then resistive. For capacitive sensors, it is the proximity and size of the contact surface that will influence a change in capacitance. In the case of resistive sensors, the response of the sensor depends on the contact force. These sensors were mainly a response to a demand for input devices to exchange information more easily with computers. Thus, from one touch to multiple touches [5], these tools have improved in spatial and temporal resolution as well as in touch description [4]. Input devices with a haptic feedback concept have also been proposed, such as the PresenseI I [7] which can vibrate thanks to multi-layer piezo-ceramics or the STIMTAC [1] which can vary its friction coefficient thanks to the squeeze film effect.

The use of these devices has expanded to other uses specifically in robotics. Firstly, these touch sensors were used on robot hands to allow gripping of objects. Then the idea of putting these sensors on the whole body of the robot was considered. The solutions developed consist of a mesh of specific sensitive cells (taxels). [2] features capacitive taxels that can be connected in a chain (up to 16 sensors), [6] inclues proximity sensors (optical), a 3-axis accelerometer (to acquire body knwoledge), a temperature sensor and a normal force sensor.

Finally, Marc Teyssier presents a different approach in his work [9][8], not that of a mesh of taxels but rather of a flexible sensor. His approach is based on biomimicry and he is trying to design a sensor that looks like skin.

As this project focuses on social interactions, the way the robot's skin is perceived is an important element. Moreover, Marc Teyssier proposes a sensor that can be easily and cheaply made, leaving a good margin for experimentation. For these reasons, the technology developed by Marc Teyssier was chosen as the starting point of the project.

5 Overall presentation of the system

The aim is to capture a touch and reproduce it using a haptic device. So the whole system consists of a touch-sensitive sensor transmitting data to a PC, a software that processes this data, transforms it and sends it to a haptic device and finally a haptic device.

The sensor is the Skin-On device (capacitive sensor)[9]. More precisely, it is a grid molded in silicone of electrodes connected to a *Muca Board*. The grid is composed of Tx electrodes parallel to each other and Rx electrodes perpendicular to the Tx (Figure 2). All electrodes are insulated from each other. The Muca Board uses a FocalTech chip (FT5316DME) which scans the electrode grid and calculates the capacitance at each intersection point $(Tx; Rx)$ of the grid. It does this by emitting a high frequency signal at each Tx electrode in turn and measuring the response at all Rx electrodes for each emission. For the sensing principle, each point $(Tx; Rx)$ is a capacitor. A finger approaching such a point disturbs the electric field and reduces the capacitance (C_m) (Figure 3).

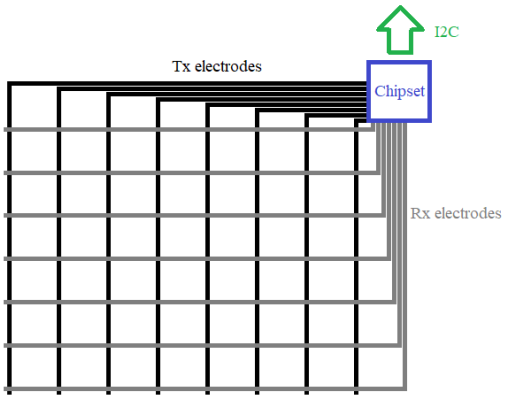

Figure 2: Projected mutual capacitance sensor

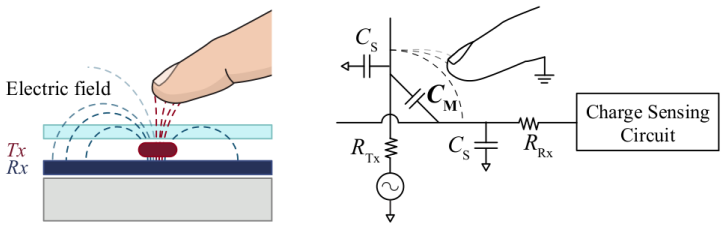

Figure 3: Sensing principle

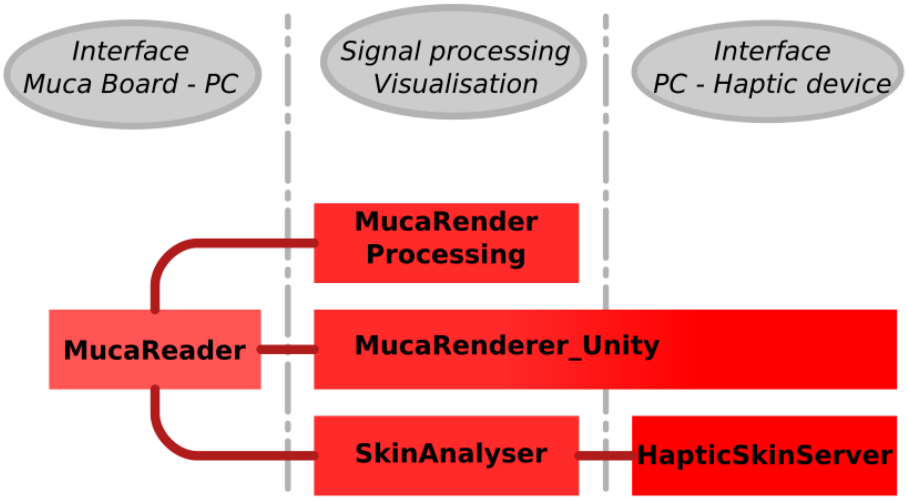

Figure 4: Software system

The Muca Board stores the matrix of measured capacitances in its registers. It is then possible to retrieve these values using a microcontroller communicating with the Muca Board via I2C. This is the *MucaReader* (Figure 4) which reads these data and sends them to a PC by UART link.

This data is then handled by a processing tool. A visualisation allows to check the good functioning of the sensor and to adjust the processing algorithms. 3 tools have been developed:

- *MucaRenderer_Processing* uses the software sketchbook Processing to process and display the data.
- *MucaRenderer_Unity* implements the same algorithms under Unity but also transforms the signal and can send it directly to BHaptics haptic devices. It can also display a visualisation.
- *SkinAnalyser* also implements the same signal processing and transformation algorithms in Java but provides a comprehensive graphical interface for setting all parameters. It also proposes a visualisation of the data. Unlike *MucaRenderer_Unity*, it does not communicate directly with a haptic device but it does offer the functionality of sending by network the signal representing the touch captured to a *HapticSkinServer* (a C# application linking to the haptic hardware). This allows for modularity in the haptic devices without modifying *SkinAnalyser*. Two *HapticSkinServers* have been developed: *BHapticsSkinServer* communicating with BHaptics hardware and *HapticVestSkinServer* communicating with the haptic vest prototyped by the University of Twente.

Haptic devices consist of clothes with motors that can vibrate at different intensities.

Figure 5: BHaptics TactSuit X40

6 Building the sensor prototype

The manufacturing process of the sensor, as described in Marc Teyssier's article [9], is based on an analogy with human skin (Figure 6). The sensor is composed of a thin, textured and colored silicone layer (DragonSkin) that represents the epidermis. Then there is the electrode grid which is sensitive like the dermis. And finally, there is a 3rd layer, in silicone (EcoFlex), used to obtain a haptic feedback and corresponding to the human hypodermis.

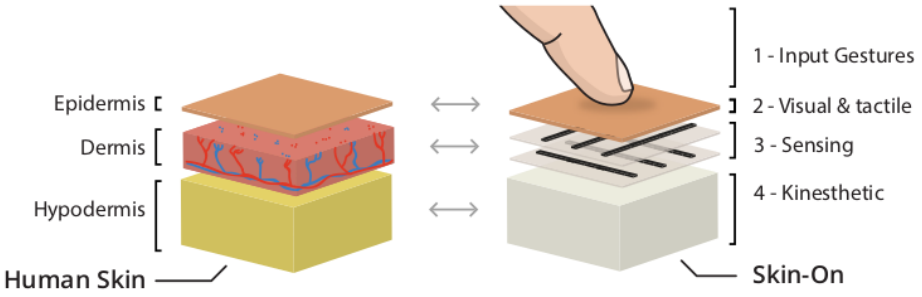

Figure 6: Skin-On analogy

The process consists in making the epidermis first by drying a layer of colored silicone on wood. The layer will then have the texture of wood. The electrodes are then positioned regularly on this layer by means of holes (like sewing, Figure 7). The second layer of silicone (hypodermis) is then poured on and thus the electrode grid is molded in the silicone. Finally, it remains to cut all the electrodes (because for the moment, it is only a long wire) and to weld one end of each electrode to the Muca Board.

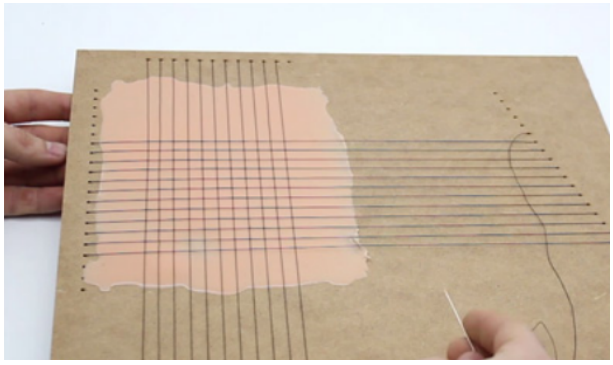

Figure 7: Sewing electrodes

On figure 8, the first prototype is presented. The Muca Board is connected to an Arduino nano board (*MucaReader*) which communicates with the pc.

The construction of a prototype involves several aspects: the choice of the number of electrodes and the placement of these electrodes, the management of the connectivity, the molding and the number of silicon layers.

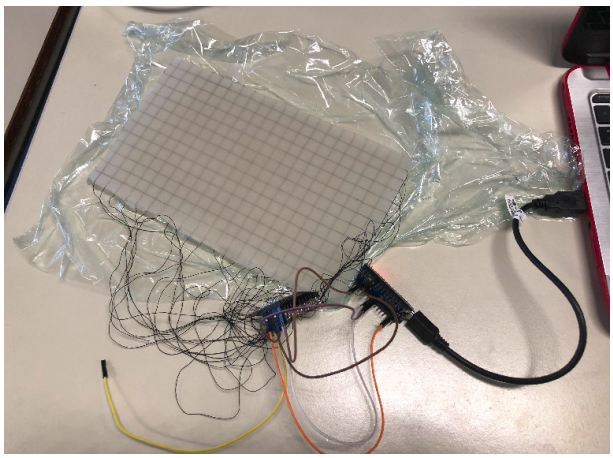

Figure 8: First prototype connected to *MucaReader*

6.1 Molding and silicon layers

The first prototype made strictly followed the instructions of Marc Teyssier's article. This resulted in a sensor that felt very much like skin. On the other hand, the EcoFlex silicone hypodermis layer was quite sticky and thick which was not very suitable for integration on a robot (it is better to have thin and not sticky skins patches). Thus, the Ecoflex was dropped and replaced by DragonSkin silicone in a thinner layer.

The molds are rectangular and are assembled from laser-cut wooden pieces (Figure 9).

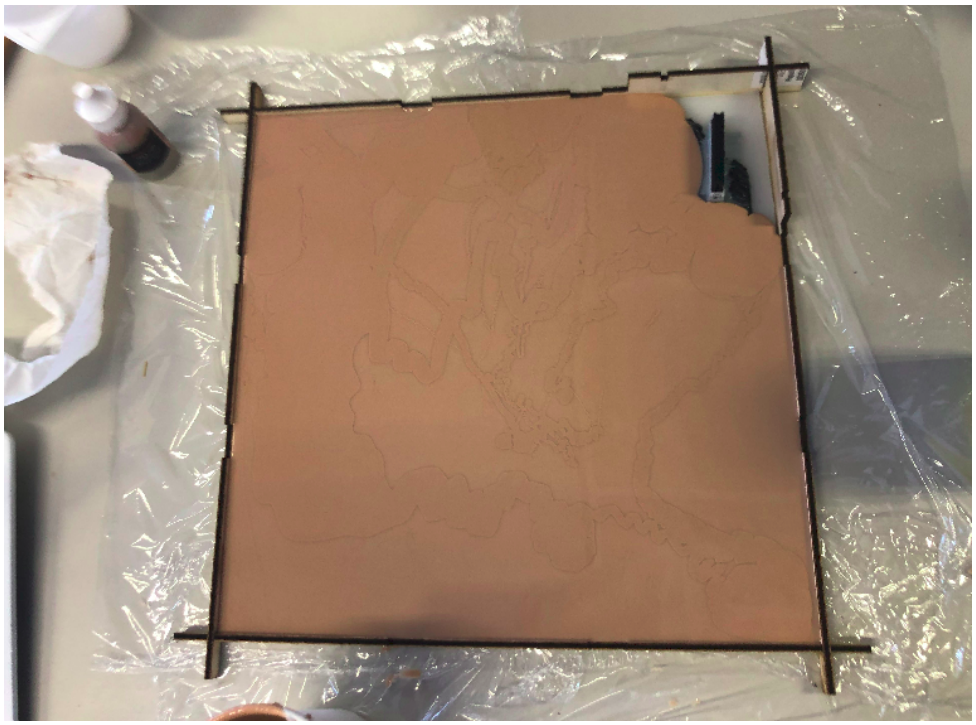

Figure 9: Curing silicone layer

Since the capacitive sensor is sensitive to the distance between the finger and the grid, it cannot directly measure the pressure exerted by the hand on the sensor. On the other hand, when the pressure exerted increases, the epidermis layer flattens which causes a

reduction of the distance between the hand and the sensor. And therefore in a higher response of the sensor.

For this reason, different layer thicknesses were tested. And indeed, a greater sensitivity to pressure was observed in the response of the sensor. However, the practicality and aesthetics of the thin sensor prevailed in the rest of the project.

This choice of thinness has led to tests to have the thinnest sensor possible. The objective was to be about 2 mm. DragonSkin silicone being rather viscous, it is not possible to make a layer homogeneous enough in thickness of less than 2 mm. To reach the objective of 2 mm of thickness, it is thus necessary that the sensor is constituted only of a single layer with inside of this layer the silicone grid.

To ensure that the electrode grid is well embedded in the silicone layer, two approaches were tried. The first one is to simply lay the cables and assume that the buoyancy will be sufficient for the cables to rise a little in the silicone. The second approach is to lift the cables in the air with washers.

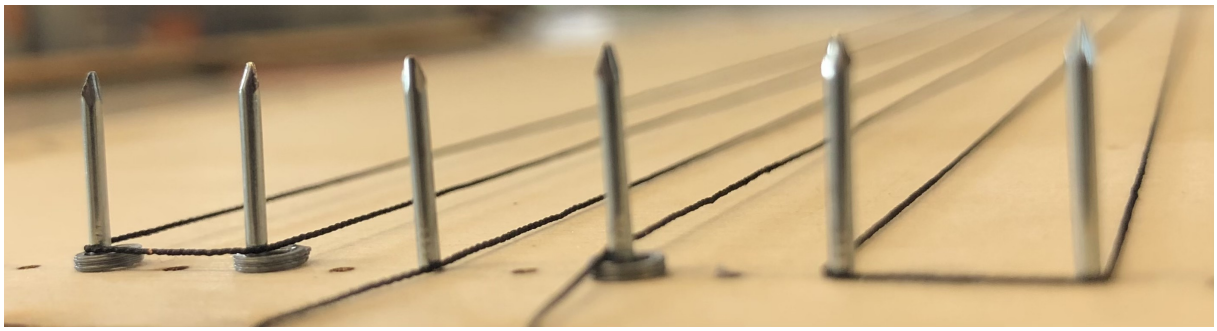

Figure 10: Initial position of the electrodes

Figure 11: Final position of the electrodes in a layer

As shown in Figure 11, the only wires properly embedded in the silicone are the wires that were raised by the washers. The other wires are not sufficiently embedded in the silicone layer and stand out in places (Figure 12).

Another interesting point is that it is no longer necessary to use designed and laser-cut parts to make the mould. Since the layer thickness is 2 mm, boards thicker than 2 mm can be nailed in place of the mould (Figure 13).

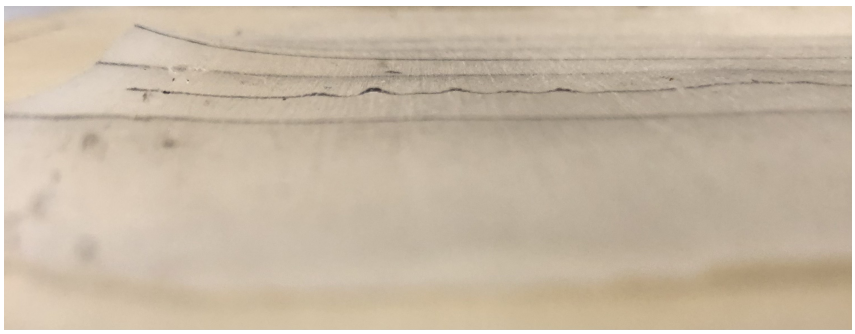

Figure 12: Some wires come out of the layer in places

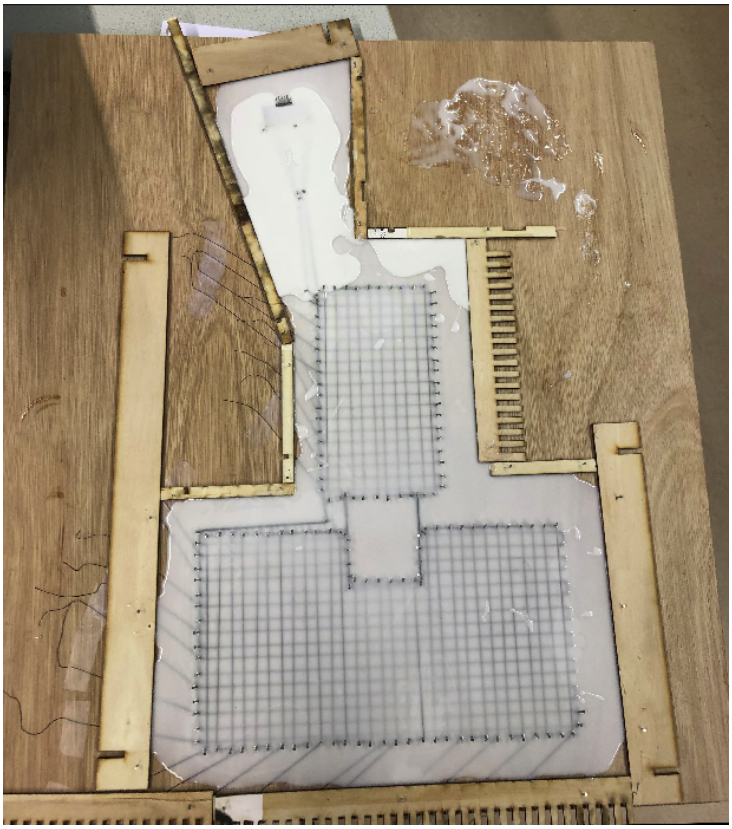

Figure 13: The mould is replaced by a series of boards nailed to the edge

6.2 Management of the connectivity

Connectivity is an important issue in integration. It is a question of robustness, practicality but also aesthetics. The first prototype (Figure 8, page 9) did not focus on connectivity at all. The aim was to get to work with the hardware first.

To improve the connectivity, the first idea was not to solder the electrodes directly to the Muca Board but to connect them to a connector embedded in the silicone. This male connector could be connected to a female connector soldered to the Muca Board. It is also possible to add a flat cable extension between the two connectors if you don't want to have the Muca Board right next to the electrode grid.

Figure 14: Soldering electrodes to flat cables

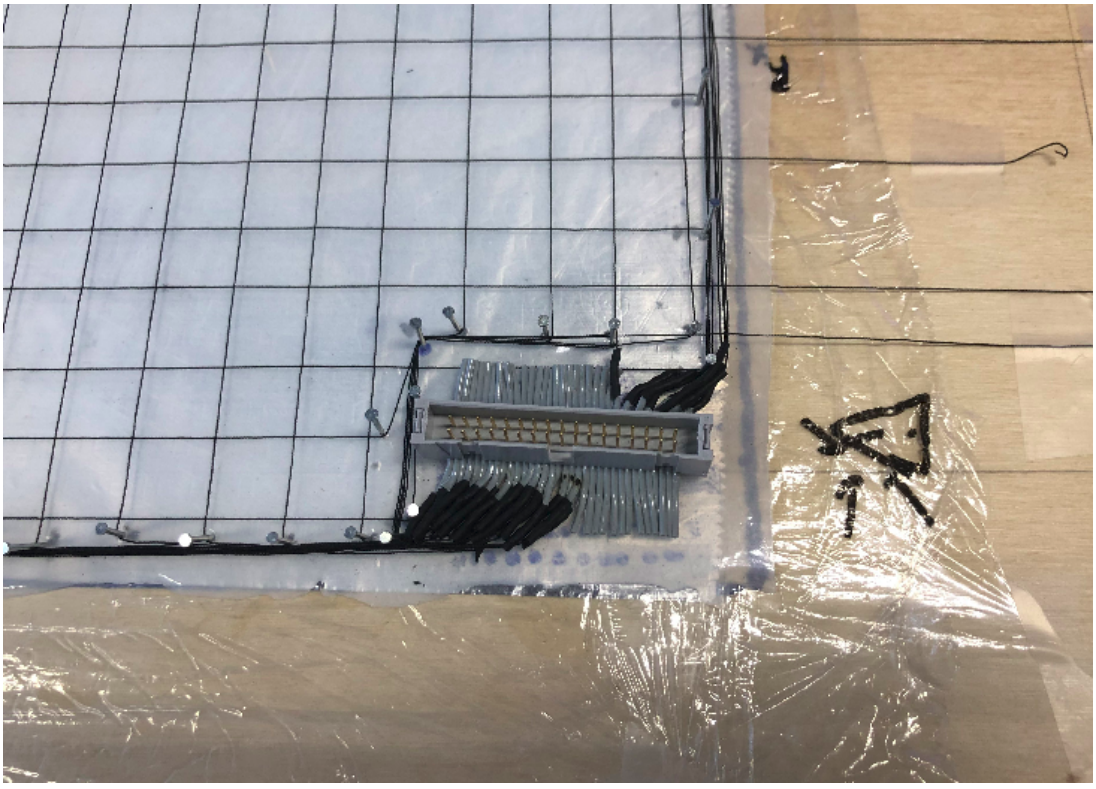

Figure 15: The connector is crimped

Figure 16: The connector is embedded in the layer

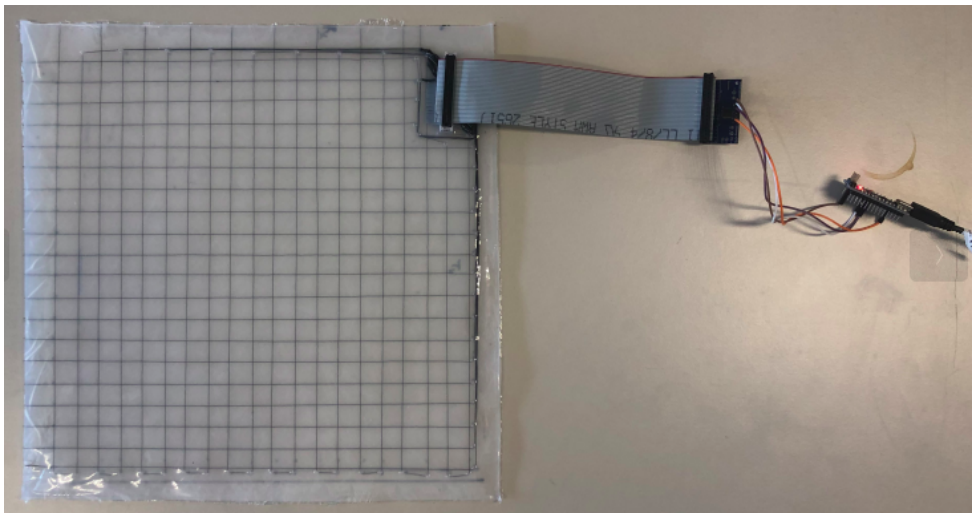

Figure 17: A flat cable extension between the Muca Board and the electrode grid

A box was produced with a 3D printer to manage the connectivity between the Muca Board and the *MucaReader*. This box can be directly connected to the male connector of the electrode grid. All that remains is to connect the *MucaReader* to the PC with a USB cable.

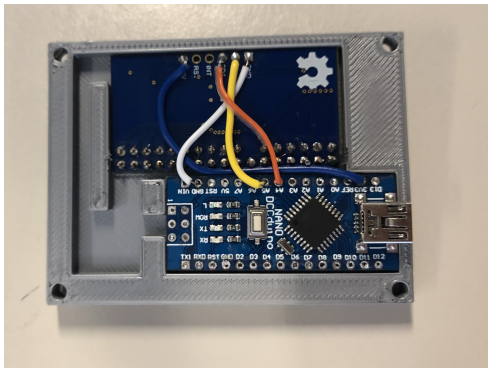

Figure 18: Box with the Muca Board and the *MucaReader*

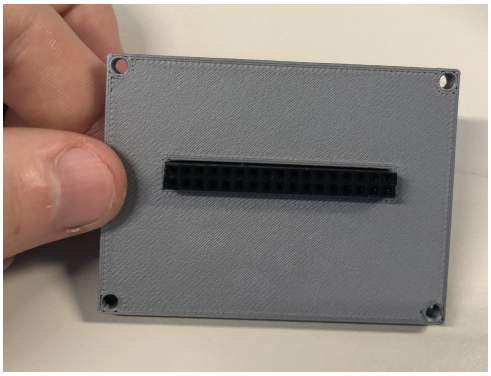

Figure 19: Back of the box

Figure 20: The box plugged with the grid of electrodes

The second idea was to solder the electrodes to the Muca Board (as before) but to embed the Muca Board in the silicone layer. The Muca Board then has a connector (just for the I2C link and power supply) that comes out of the silicone layer. The top of the Muca Board is missing a little silicone (Figure 21).

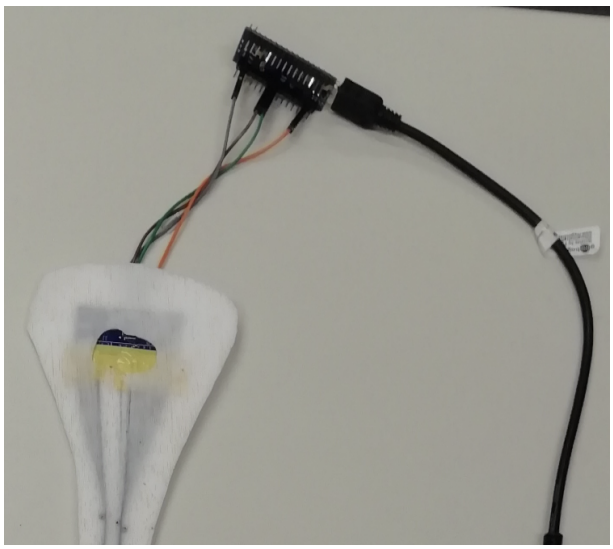

Figure 21: The Muca board is embed in the silicon layer

6.3 Building the grid of electrodes

To build the electrode grid, Marc Teyssier proposes to use holes and to make it like sewing. Unfortunately, the cables get stuck in the holes and after a few turns, it becomes impossible to continue the manipulation. The solution found was to cut the wires and to tape them at the end so that they would not move (Figure 22). Also, the use of nails allows the electrodes to have angles. This allows the electrodes to be brought together to distribute them in a cleaner way. This has enabled the development of much more complex sensor designs and shapes.

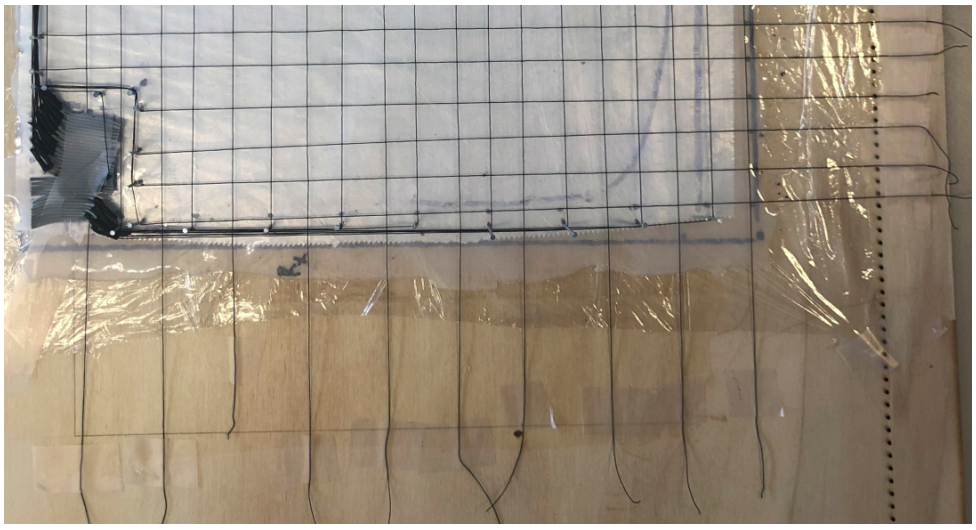

Figure 22: The electrodes are taped and the nails allow the electrodes to have angles

Concerning the spacing between 2 electrodes, after various tests, it turns out that it must be 15 mm maximum in order to avoid having insensitive zones. To increase the sensitivity of each crossing point, it is possible to make loops with any electrode (Figure 24). In this way, the spatial resolution can be reduced to increase the area covered by the sensor.

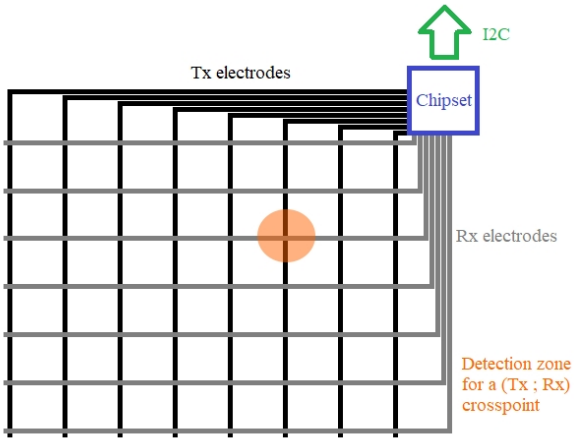

Figure 23: Detection zone for a $(Tx; Rx)$ crosspoint without loop

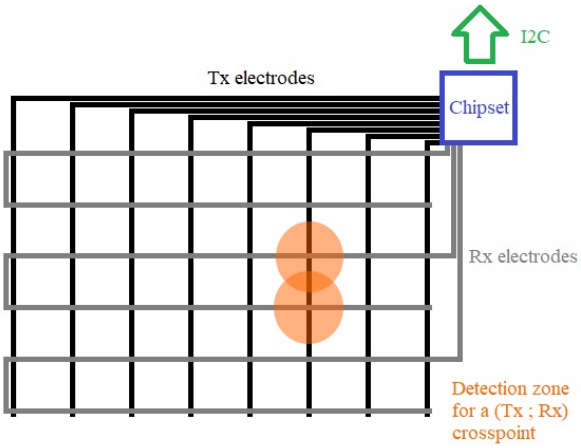

Figure 24: Detection zone for a $(Tx; Rx)$ crosspoint with loops

These two techniques have made it possible to produce a prototype to cover the torso of a humanoid robot (Figure 25). This prototype has a non-rectangular shape and must have a hole in its centre for an air vent. Its spatial resolution is not constant because it integrates loops only on its lowest part in order to cover the whole abdomen of the robot. For its attachment to the robot's torso, this sensor is designed to be worn as an apron. Two strips of fabric will be sewn to the bottom of the sensor to make the equivalent of a back belt.

Figure 25: The last prototype put on EVE Torso

7 MucaReader

7.1 Description and functioning

MucaReader is the solution at the interface of the Muca Board and the PC. It communicates with the Muca Board via I2C and with the computer via UART. The chosen hardware is an Arduino nano board. This choice is mainly explained by the small size of this board and also because this board is quite well dimensioned for the task. The code of the *MucaReader* is mainly based on the use of the Muca library coded by Marc Teyssier.

The point of interest is not in the capacitance values but in the variation of the capacitances. This is because the capacitance values are not comparable for each intersection point $(Tx; Rx)$ of the electrode grid because the length of the electrodes is different, the welds are not identical or the distance between the Tx and Rx electrode varies depending on the intersection point chosen.

The idea is to find a good reference for each $(Tx; Rx)$ and the relevant quantity is the variation between the capacitance at $(Tx; Rx)$ and the reference capacitance at $(Tx; Rx)$. And the correct reference is the value of the capacitance in $(Tx; Rx)$ when there is no touch. These values are calculated when *MucaReader* is run and then stored in an array. It's a calibration step. The capacity values are stored as 16-bit integers, but the capacity change is usually of the order of magnitude of a byte. For this reason and for performance reasons, the data sent to the PC is the capacity change values remapped to one byte per intersection point $(Tx; Rx)$.

Finally, *MucaReader* can receive certain requests from the pc. This may be an explicit request to recalibrate or to change the hardware gain of the Muca Board.

For the deployment of the code of *MucaReader*, the code can be uploaded to an Arduino board from the Arduino IDE. It is necessary to install the modified[2] Muca library and to adapt the *"TxRxPinout.h"* file to the hardware construction of the sensor.

7.2 Potential improvements

A question arose about capacitance variations. If the point $(Tx1; Rx1)$ has a different reference capacitance than the point $(Tx2; Rx2)$, is the capacitance variation at these two points the same for an identical stimulus?

One way of answering this question is to do an experiment. To do this, a sheet of aluminium foil is placed over the sensor and covers it completely. This leads to an increase in the capacitance of the points $(Tx; Rx)$. It is therefore necessary to perform a new calibration. Once *MucaReader* is calibrated, the aluminium foil is touched by a hand. When the hand touches the aluminium foil, the aluminium foil which is conductive is at the same electrical potential as the hand and consequently will cause a decrease in the capacitance of the $(Tx; Rx)$. Due to the symmetry of the experiment, the stimulus on each $(Tx; Rx)$ is strictly identical and it is then possible to compare the results provided by the sensor.

[2]mainly RAM optimisation by storing strings in flash memory and commenting out unused code

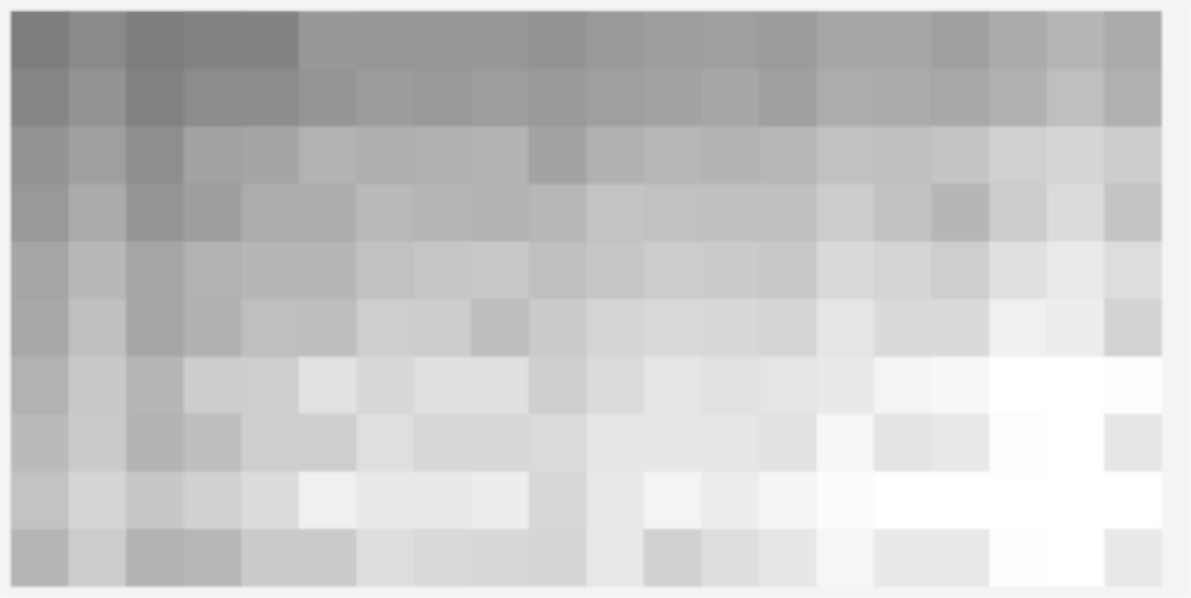

Figure 26: Output of the *MucaReader* with a aluminimu foil touched by a hand over the sensor

The Figure 26 shows a gradient from the bottom right corner to the top left point. Looking closer at the sensor, this corresponds exactly to the thickness profile! This informs that there is no noticeable difference in sensitivity between $(Tx; Rx)$ on this sensor.

But this is not enough to affirm that such a difference in sensitivity between $(Tx; Rx)$ cannot exist. Indeed, depending on the environment of the sensor, it is quite possible that all the points $(Tx; Rx)$ do not necessarily behave the same.

To limit this potential bias as much as possible, it can be interesting to perform a second calibration of the sensor: a scale calibration. The response to an identical stimulus on all $(Tx; Rx)$ can be stored and used to renormalise the sensor response.

A draft of this work (*MucaReader_ScaleCalib*) was started but problems due to the RAM limitations of an Arduino Nano prevented its completion in time.

A second point of improvement in relation to *SkinAnalyser* will be discussed in the section concerning the latter.

8 Processing Tools

These tools take the signal provided by *MucaReader* as input, process it and transform it in order to calculate the intensity with which to activate the motors of a haptic device to faithfully represent the captured touch.

8.1 Description of the filtering and transformation algorithms

There are three main points:

- Getting the most accurate and clean signal possible
- From sensor space to motor space
- Optimising the feeling of smoothness

8.1.1 Getting the most accurate and clean signal possible

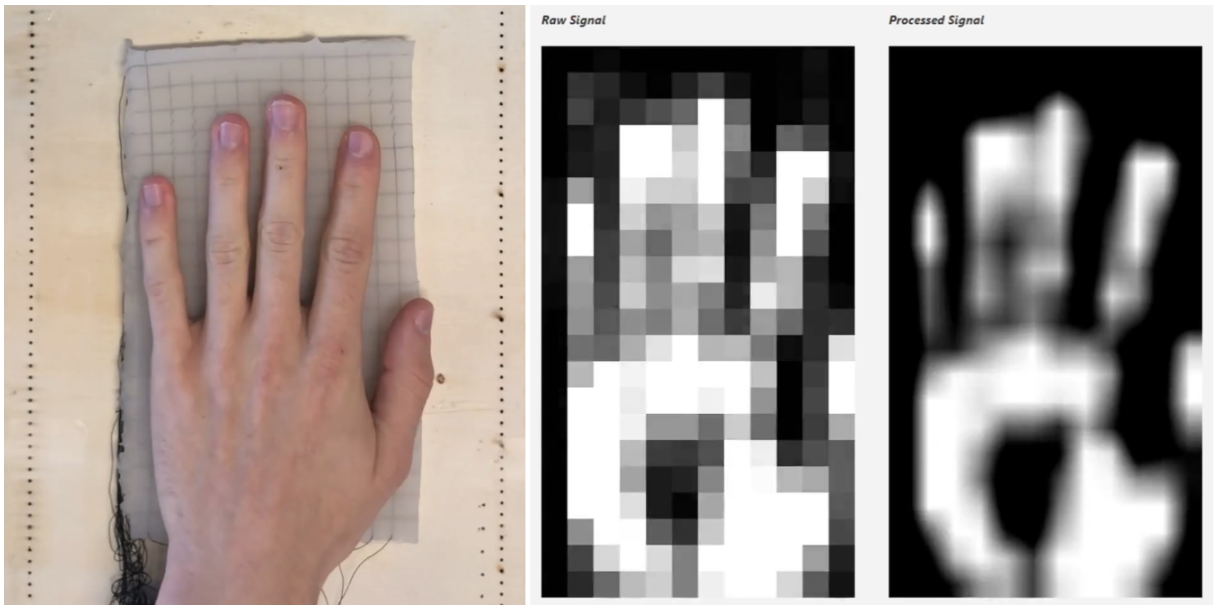

Figure 27: Signal processing

The first thing is to smooth the signal by filtering out the high frequencies which are the high frequencies of noise. The second is to increase the resolution of the signal by interpolating (image upscaling). The third is to threshold the signal to eliminate the noise. As these last two operations are not commutative, it is important to interpolate first and then threshold in order to obtain a precise signal. In contrast, low-pass filtering and interpolation (which is a convolution) are both linear operations and are therefore commutative. Thus it is better to perform the low-pass filter on the signal with the lowest resolution in order to minimise the computation for the computer. The need for thresholding comes from the fact that the DC component of the noise is non-zero (here the signal is even positive) and must therefore be cut off.

On a practical level and order of magnitude, the low-pass filter used is a rolling average over the last 3 frames (i.e. over the last 100 ms) to keep a good reactivity, the value of the thresholding depends on the environment (but around 20% of the max value) and I used a bilinear interpolation (faster and easier to code than bicubic or Lanczos) with a resize factor of 20 (the distance between two sensors (i. e. a $(Tx; Rx)$ point) was 10 mm, so the spatial accuracy of the interpolation output was about 0.5 mm). The interpolation is relevant if the number of interior points (sensors that have at least on neighbouring sensor in each direction or in other words that are not on the edge) is sufficiently large. In the case of the protypes I worked on, 75% of the sensors were interior points.

8.1.2 From sensor space to motor space

The processed signal has a very high resolution compared to the spatial resolution of the motor grid. The idea is to take advantage of this high resolution to activate the motors as intelligently as possible. To determine the intensity of a motor, I chose to perform a spatial convolution of the signal at each instant with a Gaussian kernel centred at the motor position. The choice of the standard deviation of the kernel allows to decide the size of the motor activation zone.

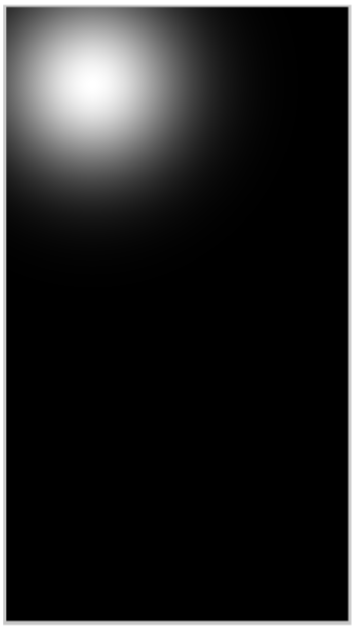

Figure 28: Gaussian Kernel

In Figure 28, the kernel showed is used for the calculation of the motor intensity which must reproduce the touch measured in the top left corner of the skin.

Concretely, doing a spatial convolution just means multiplying term by term 2 images (the kernel image with the signal image) and summing.

8.1.3 Optimising the feeling of smoothness

The difficulty with the motor feel is to get a precise but also smooth rendering in the case of a moving touch.

If for a localized touch, only one motor is activated at the time of the touch, then the rendering is accurate, but for a moving touch, there is a succession of distinct motor activations. This is unpleasant and does not correctly represent the notion of movement which is continuous.

On the other hand, if for a localized touch, we activate several motors (with different intensities for example) then indeed, the rendering of the movement is better but there is a loss of precision for a fixed touch.

One idea is to make the activation of the motor last a little while, even after there is no more touch.

In terms of implementation, this would partly correspond to introducing a delay by translating the signal in time. But it could be interesting to have also a damping effect so that the activation of the motor does not suddenly stop. This corresponds exactly to a low-pass filter with delay (a rolling average with the previous frames!). But the delay is not desired at the beginning of the touch (otherwise we just made the experience worse). So it is sufficient to take the maximum between the signal without delay and the signal with delay. Thus, the rising edges of the touch will be represented by an immediate activation of the motors but the falling edges of the touch will be represented by a progressive deactivation of the motors, which is the desired effect.

The goal of the game to optimize the rendering is therefore to find good parameters for the standard deviation of the Gaussian kernel (thus the size of the activation zone of a motor) and the delay (if it is too big, we feel a too slow rendering).

I obtained convincing results for a touch motion of 0.3 m/s when the introduced delay was of the order of 300 ms and the standard deviation of the Gaussian kernel was of the order of the intersensor distance (10 mm). And a fixed touched was rendered with accuracy.

These algorithms have been implemented to a greater or lesser extent in the tools presented.

8.2 MucaRenderer_Processing

The starting point was once again the work of Marc Teyssier. He proposes a Processing sketch performing signal processing algorithms using OpenCV. I started by using this work and adapting it to the *MucaReader* protocol (protocol with 5 times less data). I also modified the graphical interface to display not only the processed signal but also the raw signal for comparison.

This sketch uses Processing version 3.5.4 (because there are problems with OpenCV and Processing version 4) with the *ControlP5* and *OpenCV for Processing* libraries. However, the same code does not work under Ubuntu although it works perfectly under Windows...

On the Figure 29, we can see the raw signal on the left, the processed signal on the right and finally on the right side the filtering parameters.

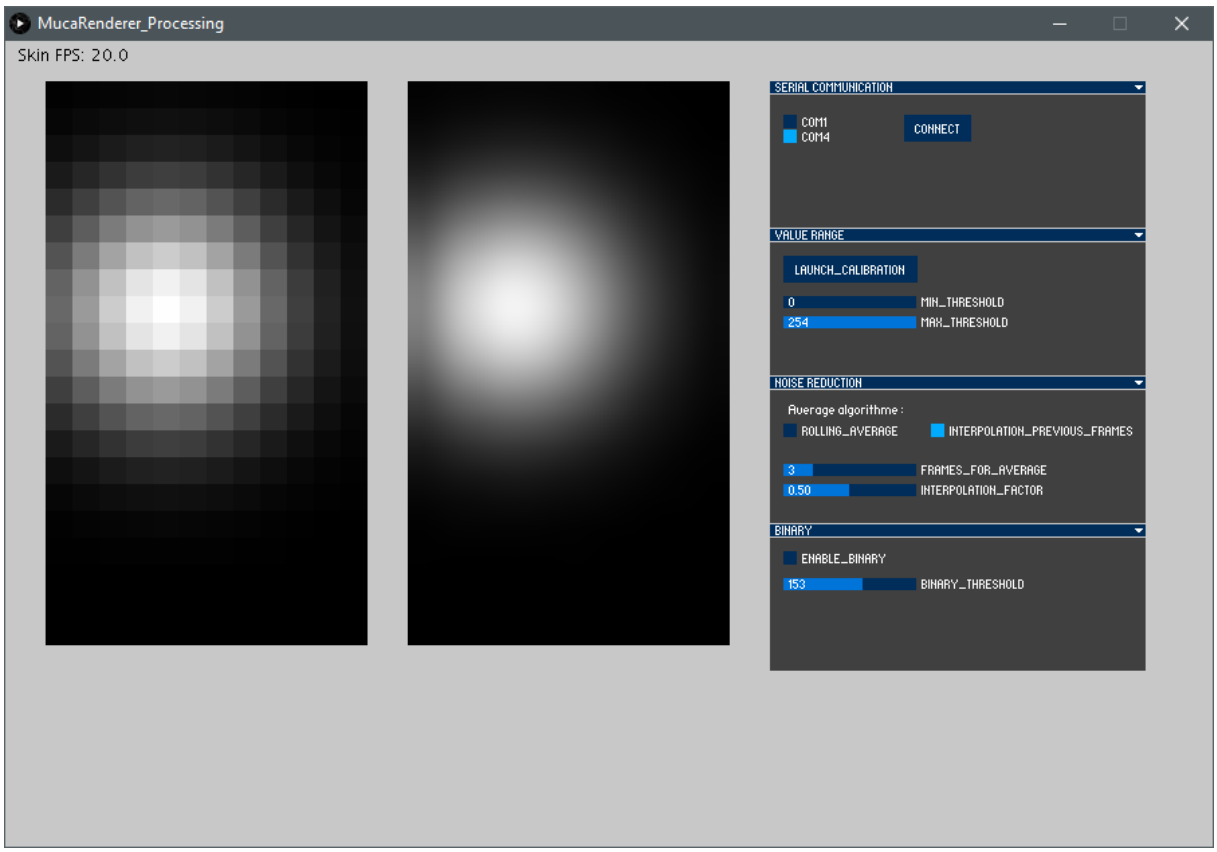

Figure 29: *MucaRenderer_Processing* interface

The convolutional signal transformation algorithms (motor intensity calculations) started to be coded in this sketch, however, the need for a more complete IDE became apparent and the maintenance of the project became difficult. For this reason, the calculation of the motor intensities was not integrated. And another project *SkinAnalyser* was started instead. It integrates all the algorithms described above.

8.3 SkinAnalyser

8.3.1 Presentation

SkinAnalyser is in the continuity of *MucaRenderer_Processing*. It is also a Java project. The choice of the IDE was Eclipse. I chose JavaFX as the graphical framework.

The software architecture deployed in this project proposes an encapsulation of all the parameters of the algorithms in what is called a *configuration*. *SkinAnalyser* is primarily composed of a graphical system for managing and modifying configurations (Figure 30). This configuration management system allows to instantiate a *Renderer* (Figure 31) associated with a specific configuration and a serial port. The Renderer then connects via serial port to the *MucaReader* and processes the data collected according to the parameters of its configuration.

It is possible to edit a configuration while it is being rendered and to see the real-time effects of configuration changes. Also, multiple Renderers can be instantiated (but

associated with different serial ports). In addition, a configuration can be serialized and exported and loaded again. This allows to create a configuration file for each developed prototype.

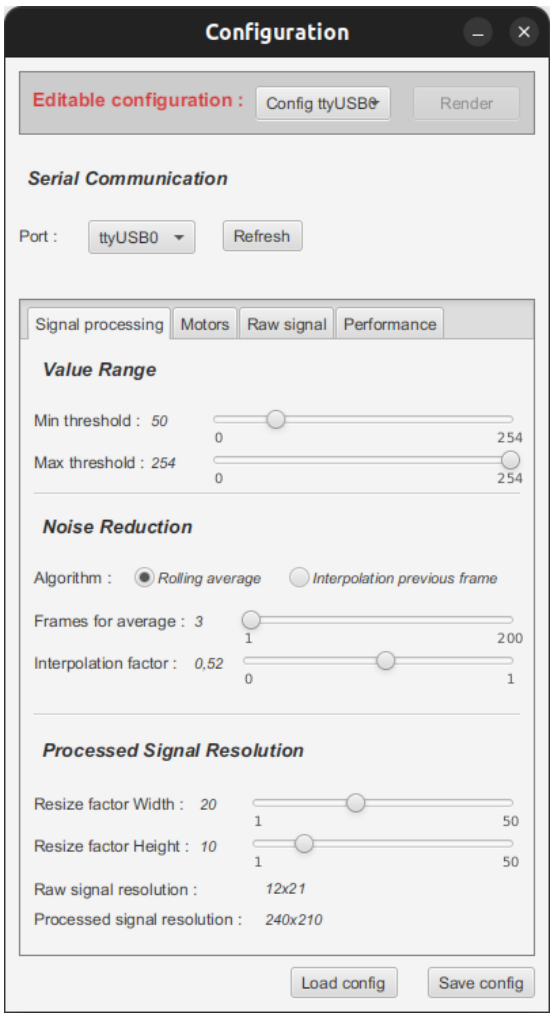

Figure 30: *SkinAnalyser* configurations manager

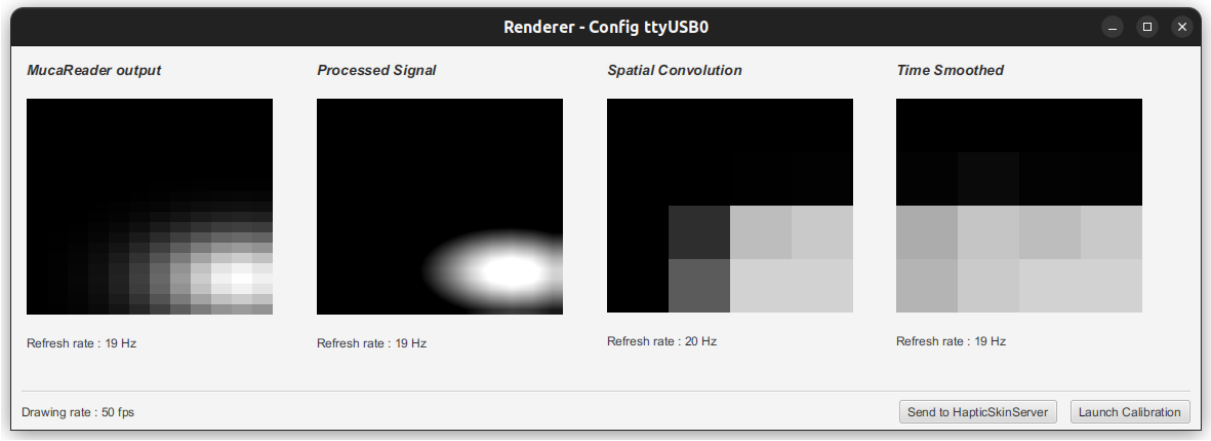

Figure 31: A *SkinAnalyser* Renderer

8.3.2 Deployement

For the deployment of this application, a .jar file is available. As this project uses JavaFX, it runs on Java 8 without adding anything. It has been tested on Windows with the latest version of Java 8 available (version 8.0.341-b10) and with version 8.0.202. It has also been tested on Linux with version 8.0.202 distributed by **Oracle** (not openjdk which does not integrate javaFX). The choice of version 8.0.202[3] comes from the fact that this version is the last one under the Binary Code License and not under the OTN License Agreement. This does not allow to benefit from the latest security patches but it still allows a free and open use and a simplified deployment and development of the application.

Listing 1: Launching *SkinAnalyser*

```
#!/bin/bash
java -jar SkinAnalyser.jar
```

8.3.3 How to developp SkinAnalyser

First, make sure you have **Oracle JDK 8.0.202** installed on your machine. It can be downloaded for free from the Oracle website (but you need an Oracle account). Second, make sure you have **Apache ANT** also installed (version 1.7 or higher) and properly configured (ANT_HOME and JAVA_HOME environment variables)

Listing 2: Downloading sources

```
#!/bin/bash
git clone https://github.com/antNLocks/UT_skin.git
cd UT_skin/ProcessingTools/SkinAnalyser/
```

Then you can edit the source code using your favourite IDE and build and export the project:

Listing 3: Building and exporting *SkinAnalyser*

```
#!/bin/bash
ant
```

Alternatively, it is possible to use Eclipse or VSCode to build and export the project. There is nothing to do except, under Eclispe, to import the project (and make sure to use the right version of Java to compile and that all folders in the Build Path are selected) and, under VSCode, to have some Java extensions (I tested the ones from Microsoft).

8.3.4 Potential improvements

The improvement that seems to me the most practical is related to *MucaReader*. Instead of exporting a configuration linked to a prototype (and thus to a *MucaReader*) to a file

[3] released on 01/15/2019

stored on a PC, the best would be to send it to the *MucaReader* and have it store the data in its EEPROM. Thus, it would be sufficient to connect the *MucaReader* to the PC and simply choose the correct serial port in the configuration manager of *SkinAnalyser*. The configuration would then be automatically collected and loaded.

There may also be a lot to do on the performance side. Using OpenCV for image resizing calculations as well as GPU programming to parallelize convolution calculations could greatly improve performance.

It might be interesting to add a recording function to the Renderers so that it is possible to replay a signal without having the sensor.

Finally, a port to an updated and free version of Java seems inevitable. For that, it is good to check out the notion of module in Java and the jpackage tool allowing to create independent executables for a specific platform (the executable embeds what it needs from the JRE).

8.4 MucaRenderer_Unity

This is a port of some SkinAnalyser features to Unity.

This project receives the data transmitted by *MucaReader* and processes it. It integrates the Unity SDK of BHaptics and can therefore control the motors of the BHaptics vest. The reason for this port was the ease of communicating with the BHaptics haptic device from Unity, whereas this seemed more difficult in Java (no code was provided). And also because a lot of the tools around the robot control were done in Unity.

This project uses **Unity version 2020.3.7f1** and the **bHapticsPlayer** software is required.

9 HapticSkinServer

9.1 Presentation

HapticSkinServer are server applications receiving as input the motor intensities to represent the touch captured by *SkinAnalyser* and sending them to a haptic device. This data is received over the network (TCP connection via port 51470).

The C# script *HapticSkinServer.cs* provides an implementation for this connection.

All that remains is to transmit intensities to the hardware that controls the motors. Two examples are proposed:

- *BHapticsSkinServer* to control hardware provided by BHaptics. The **bHapticsPlayer** software is required
- *HapticVestSkinServer* to control the vest prototyped by the UT

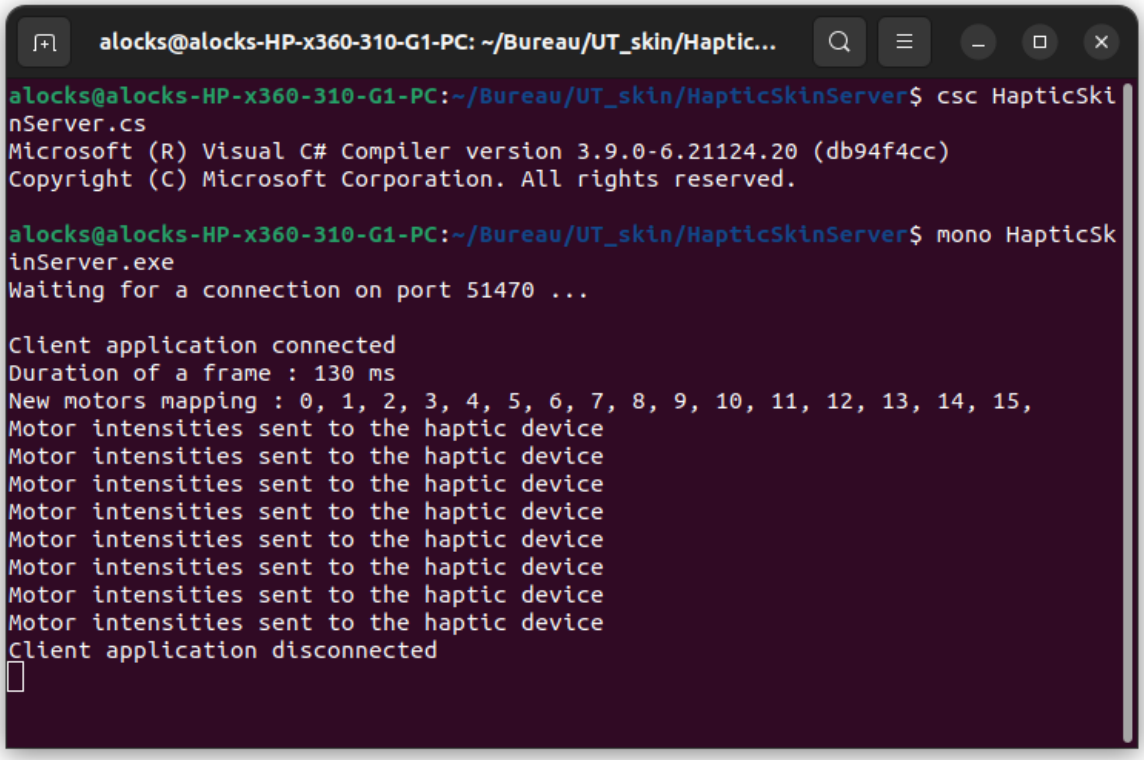

Figure 32: *HapticSkinServer*

9.2 Deployement

As shown in Figure 32, it is possible to use **mono** to compile the script and run it.

Listing 4: Building a C# script

```
#!/bin/bash
csc HapticSkinServer.cs
```

Listing 5: Executing a C# script

```
#!/bin/bash
mono HapticSkinServer.exe
```

The project *BHapticsSkinServer* is not a simple script but uses a Visual Studio solution.

10 Conclusion

This internship was an opportunity for me to discover the world of university research. I was particularly pleased to be able to work on the whole system from the design of the sensor to the interfacing with the haptic vest developed by the University of Twente.

I learned a lot in many areas, but if I had to mention only one, it would be without any surprise the IT. From the fine management of RAM in a microcontroller to concurrent programming in Java and the appropriation of a graphical framework, I, like any programmer, have lost count of the hours I have spent reading documentation and debugging. And I enjoyed it (especially when it ended up working the way I wanted).

I would have liked to stay a few more weeks to finish working on the integration of the torso covering prototype and to experiment with the preparation and processing of a social study.

More generally, I feel very lucky that this internship has given me a better understanding of my professional desires. It has also been a very interesting experience abroad and a nice change from what I have been used to until now.

I hope that this work that I have been entrusted with will continue to make its way and that I will see its outcome.

References

[1] Michel Amberg et al. "STIMTAC: a tactile input device with programmable friction". en. In: *Proceedings of the 24th annual ACM symposium adjunct on User interface software and technology - UIST '11 Adjunct*. Santa Barbara, California, USA: ACM Press, 2011, p. 7. ISBN: 978-1-4503-1014-7. DOI: 10.1145/2046396.2046401. URL: http://dl.acm.org/citation.cfm?doid=2046396.2046401 (visited on 10/02/2022).

[2] Giorgio Cannata et al. "An embedded artificial skin for humanoid robots". In: *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. Seoul: IEEE, Aug. 2008, pp. 434–438. ISBN: 978-1-4244-2143-5. DOI: 10.1109/MFI.2008.4648033. URL: http://ieeexplore.ieee.org/document/4648033/ (visited on 10/02/2022).

[3] Esther P Gardner. "Touch". en. In: *eLS*. Ed. by John Wiley & Sons, Ltd. 1st ed. Wiley, May 2010. ISBN: 978-0-470-01617-6 978-0-470-01590-2. DOI: 10.1002/9780470015902.a0000219.pub2. URL: https://onlinelibrary.wiley.com/doi/10.1002/9780470015902.a0000219.pub2 (visited on 10/02/2022).

[4] Ken Hinckley and Mike Sinclair. "Touch-sensing input devices". en. In: *Proceedings of the SIGCHI conference on Human factors in computing systems the CHI is the limit - CHI '99*. Pittsburgh, Pennsylvania, United States: ACM Press, 1999, pp. 223–230. ISBN: 978-0-201-48559-2. DOI: 10.1145/302979.303045. URL: http://portal.acm.org/citation.cfm?doid=302979.303045 (visited on 10/02/2022).

[5] Sk Lee, William Buxton, and K. C. Smith. "A multi-touch three dimensional touch-sensitive tablet". en. In: *ACM SIGCHI Bulletin* 16.4 (Apr. 1985), pp. 21–25. ISSN: 0736-6906. DOI: 10.1145/1165385.317461. URL: https://dl.acm.org/doi/10.1145/1165385.317461 (visited on 10/02/2022).

[6] P. Mittendorfer, E. Yoshida, and G. Cheng. "Realizing whole-body tactile interactions with a self-organizing, multi-modal artificial skin on a humanoid robot". en. In: *Advanced Robotics* 29.1 (Jan. 2015), pp. 51–67. ISSN: 0169-1864, 1568-5535. DOI: 10.1080/01691864.2014.952493. URL: http://www.tandfonline.com/doi/abs/10.1080/01691864.2014.952493 (visited on 10/02/2022).

[7] Jun Rekimoto and Carsten Schwesig. "PreSenseII: bi-directional touch and pressure sensing interactions with tactile feedback". en. In: *CHI '06 Extended Abstracts on Human Factors in Computing Systems*. Montréal Québec Canada: ACM, Apr. 2006, pp. 1253–1258. ISBN: 978-1-59593-298-3. DOI: 10.1145/1125451.1125685. URL: https://dl.acm.org/doi/10.1145/1125451.1125685 (visited on 10/02/2022).

[8] Marc Teyssier et al. "Human-Like Artificial Skin Sensor for Physical Human-Robot Interaction". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. Xi'an, China: IEEE, May 2021, pp. 3626–3633. ISBN: 978-1-72819-077-8. DOI: 10.1109/ICRA48506.2021.9561152. URL: https://ieeexplore.ieee.org/document/9561152/ (visited on 10/02/2022).

[9] Marc Teyssier et al. “Skin-On Interfaces: A Bio-Driven Approach for Artificial Skin Design to Cover Interactive Devices”. en. In: *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. New Orleans LA USA: ACM, Oct. 2019, pp. 307–322. ISBN: 978-1-4503-6816-2. DOI: 10.1145/3332165.3347943. URL: https://dl.acm.org/doi/10.1145/3332165.3347943 (visited on 10/02/2022).