**Design of a touch sensor for acquiring continuous pressure and position data over a defined surface**

**Karon MacLean, Laura Cang, Andrew Strang**

**September 22, 2014**

Table of Contents

1. INTRODUCTION 2

2. BACKGROUND 2

Earlier, simplified fabric sensor design 3

Motivation & Requirements 3

Related Work 3

Present design 4

Objectives 4

3. CONSTRUCTION PHASE 4

Initial construction 4

Hardware Requirements 7

Hardware Challenges 8

Resolution of the Sensor 9

Scaling Issues 9

Processing Hardware and Algorithm 9

Electrical Connections 10

Considerations for Physical Construction 12

Sources of Interference 12

Pre-Loading 13

Cross-talk 13

Noise due to to shorts 14

4. CHARACTERIZATION PHASE 14

Requirements for Gesture Recognition on Non-flat, Compliant Surfaces 14

5. DISCUSSION 15

6. CONCLUSIONS 15

7. APPENDIX 15

# INTRODUCTION

Flexible wearable touch sensors have become affordable and very accessible.  Due to inexpensive electronics and even Do-It-Yourself guides (Perner-Wilson & Satomi, 2009), anyone with a healthy interest in constructing their own touch sensor, can.  However, while the techniques described do a good job of presenting a variety of low cost pressure sensors, we wondered if we could improve on the sensing capabilities and recognize relatively complex gestures as well.  We are interested in leveraging this multidimensional touch data to be able to perform relatively high-resolution gesture recognition on our low-resolution data.  This kind of sensor could allow for a variety of touch interfaces that may afford a range of interesting interactions while keeping materials and construction costs very low.

The SPIN lab has been working toward the design of a touch sensor for use with robot interfacing. The specific application that was targeted was a therapeutic robot pet, which would require continuous coverage over its body to sense gesture in terms of the position and pressure of the user's touch. It was posited that the number of pressure points at any instant, corresponding to the number of fingers used in the gesture, would also be required information.

In order to acquire the relevant gesture information, it was posited that a grid of touch points, each roughly the size of a fingertip, could provide accurate position sensing and be able to accommodate multiple points of contact. A conductive fabric-based sensor was constructed using a combination of fabrics: Eeonyx Corporation’s Eeontex ‘Zebra’ striped fabric and Eeontex SLPA resistive fabric. The ‘Zebra’ fabric consists of electrically conductive strips, each approximately 20-25 mm wide, separated by non-conductive strips of approximately 5 mm width. Layering two sections of this fabric on top of each other, arranged at 90 degrees to each other respective to the direction of the strips, creates a conductive grid pattern. By sandwiching the SLPA resistive layer in between, it becomes possible to detect changes in resistance and location of pressure when the fabric ‘sandwich’ is touched.

# BACKGROUND

## Earlier, simplified fabric sensor design

An earlier version of the sensor, the piezoresistive ‘sandwich’ sensor, consists of two layers of resistive fabric (Eeontex LTT-SLPA-20K was used with good results) with a layer of plastic mesh in between. This allows for a drop in resistance when the sensor is pressed at a particular location, which is determined by comparing the relative resistance of each layer.

As shown by Freed et al., when opposing edges of the two resistive layers are polled in sequence by a processor, the centroid and amplitude of pressure can be located. This sensor detects the centroid of contact; that is, it does not determine individual touch points or the total area of contact, but rather the position and pressure of the centroid of contact. This is not as much information, clearly, as is given by the ‘pixels’ of the zebra-striped conductive fabric, but it allows for a much simpler circuit.

The design which has been developed makes use of a fabric with conductive 'stripes' on top and bottom and a resistive layer in between, with the 'stripes' of the layers arranged at 90 degrees to each other, forming a grid. This allows both pressure sensing through the resistive layer and location sensing through the 'pixels' created by the grid.

## Motivation & Requirements

Previous work related to gesture recognition has employed the interplay between “below surface” pressure sensors and “above surface” touch activation.  Anna Flagg (TEI, 2013) built a zoomorphic creature that had a pressure sensor installed below a conductive fur surface.  In this way, the creature could sense the relatively high pressure forces of the touch using the “below surface” sensing capabilities, as well as determine location based on conductivity activation of the “above surface” conductive fur.

From Flagg’s work, we learned that the conductive fur did not contribute as greatly to the gesture recognition capabilities as compared to the pressure and location data.  Thus we chose to improve upon the fabric sensor to yield both pressure and positional information, the goal being that it would perform relatively well in on-the-fly gesture recognition tasks on a variety of surfaces while keeping construction and computation costs low.

We also required that our sensor to be able to perform well even if mounted over non-rigid and/or actuated surfaces, requiring that the sensor be both highly flexible and somewhat elastic.

## Related Work

## Present design

Prior to the present prototype's design, the idea was developed that a series of conductive fabric strips could be overlapped in a grid pattern, with a resistive layer between them, which would allow both pressure sensing through the resistive layer and more precise position sensing through the grid. This idea became the focus of the current sensor design.

## Objectives

The sensor was designed to meet the following criteria:

- Have a range of sensitivity from very light to very hard touches

- Have an approximately linear response to pressure

- Be positionally accurate to roughly within a finger width; resolution of the sensor should also be easily adaptable.

# CONSTRUCTION PHASE

## Initial construction

A first pass at the grid sensor was undertaken in early 2013, with inconclusive results. In September 2013, construction began on a new prototype of this sensor, in an effort to characterize it and determine its limitations for position and pressure sensing.

A 5x5 grid was constructed using the Eeontex 'Zebra' fabric (5 strips wide on both the top and bottom layers), with Eeontex SLPA 20,000 Ohm resistive fabric sandwiched in between. This grid size was chosen based on two main factors: the available input/output pins on a basic Arduino processor, and the minimum size needed to identify sensor cross-talk for multiple touch points.



Figure 1 - The 3 layers of the sensor

In order to poll the sensor, a circuit was constructed in which each strip of the conductive fabric was wired to either a digital output or a paralleled pair consisting of a digital output and analog input. As explained in the algorithm section of this document, the digital outputs were designed to sequentially send a ‘high’ voltage signal to each row and column, effectively polling each ‘pixel’ or ‘square’ of the sensor. The resistance value of each ‘pixel’ would then be read by the analog inputs in sequence. As seen in Fig. 2, the digital output of the paralleled pair was connected in series to a resistor; this resistance value acted as a voltage divider and was chosen to optimize the sensitivity and range of the sensor readings. Pictured in Fig. 1 is the circuit chosen for the therapeutic robot application, which was found to have a satisfactory range and sensitivity using a 1,000 Ohm resistance value in a 16x16 grid.



Figure 2 – Sensor connections to the Arduino processor’s pins

Because the initial 5x5 grid required 10 digital and 5 analog pins, an Arduino Uno, with the commonly available configuration of 8 analog and 16 digital I/O pins, was deemed a suitable choice for processing. To accommodate further expansion of the grid size, an Arduino Mega was used, with 16 Analog inputs able to accommodate at least a 16x16 grid. A larger asymmetrical grid would also be possible with this configuration – with 54 digital pins on the Arduino Mega, it would be possible to connect a 16x38 grid before requiring multiplexers.

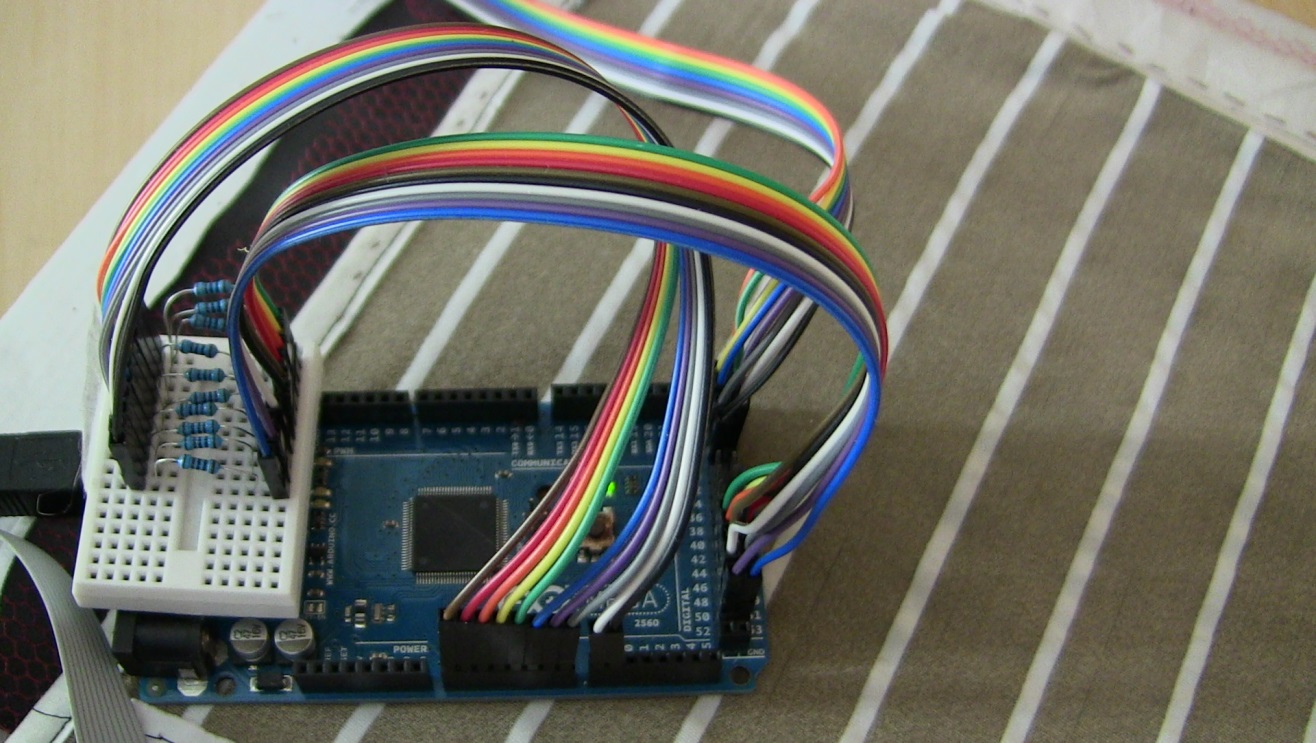


Figure 3 - The breadboarded circuit design, using an Arduino Mega.

## Hardware Requirements

Initially, this sensor was constructed using the following materials:

- Eeonyx 'Zebra' fabric

- Eeonyx resistive fabric SLPA-20K

- Plastic mesh, to create a standoff layer and prevent static noise

- Resistors, chosen to optimize sensitivity and range

- Arduino Mega microprocessor board

- Jumper wires

- USB connection form Arduino to PC

In later stages, the Arduino Mega was replaced with a Teensy 3.1 board running on the Arduino platform, with the sensor inputs and outputs connected to multiplexers. This allowed for much faster processing times and much fewer inputs/outputs on the processing platform. The circuit for this version is shown in the diagram below.

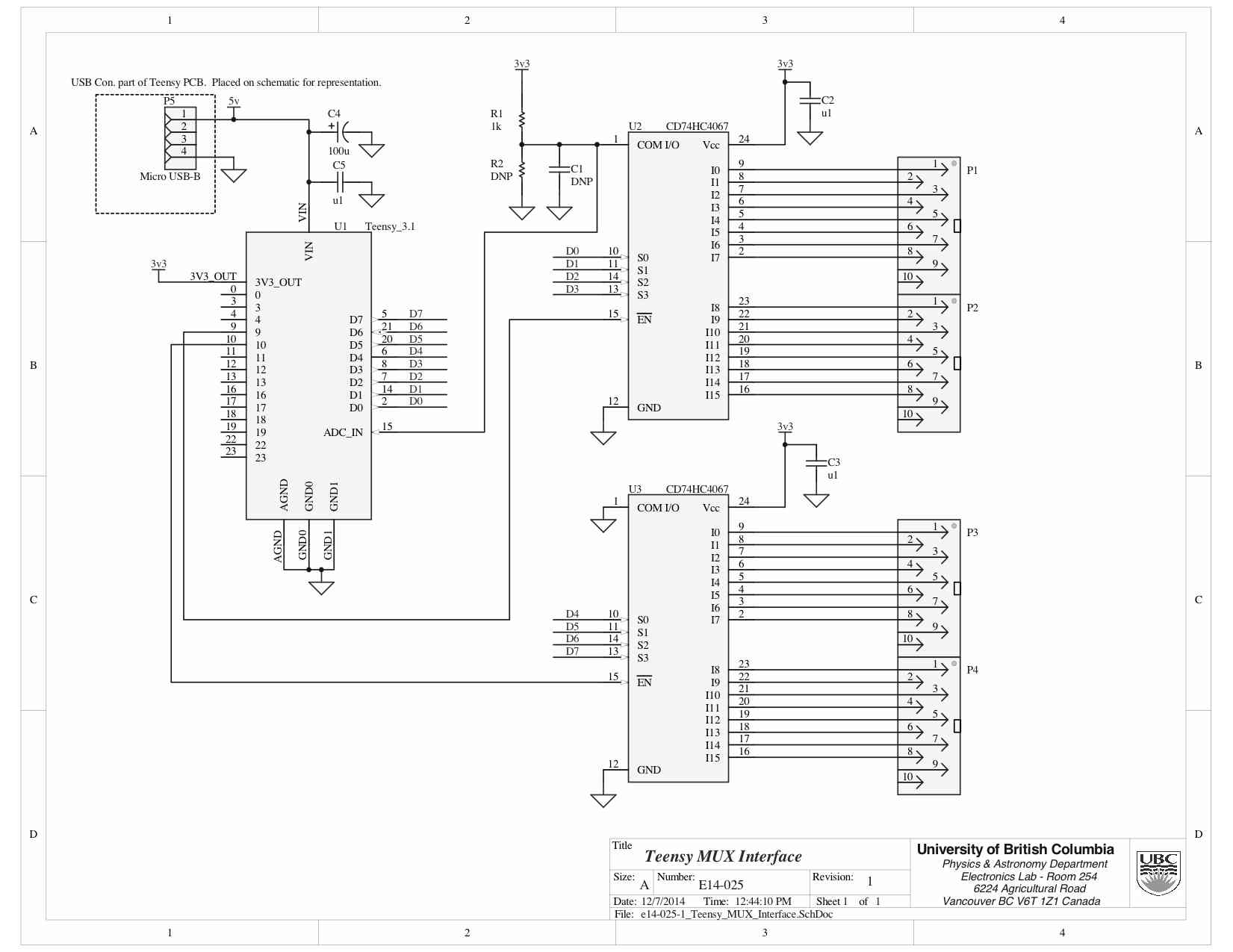


Figure 4 – the Teensy 3.1 multiplexer circuit for the sensor

## Hardware Challenges

While constructing and testing the sensor, several challenges became apparent in the acquisition of consistent and clean data. Mitigating these challenges became an integral part of the design process. They include:

- Pre-loading of the sensor due to adhesion between the layers

- Cross-talk noise occurring with multiple contact points

- Noise due to shorts within the fabric or wiring

All three of these factors were affected by the physical construction of the sensor. The efforts to eliminate these issues are explained later in this document.

## Resolution of the Sensor

It was decided that the Sensor's 'pixels', or conductive grid squares, should be approximately scaled to the width of an average human fingertip. Motion could then be detected by the sensing of one or more neighboring 'pixels' changing their pressure reading as a result of slight movements.

## Scaling Issues

Starting with a 5x5 grid simplified the early processing and seemed a reasonable optimization of sample size and data efficiency. Scaling this up to our desired 16x16 grid introduced one main occurrence: the resistance of the middle layer of the sensor was reduced slightly as the edges of the sensor were approached, whereas the middle section of the sensor saw slightly increased resistance. This meant that the pressure values recorded differed by up to 5% when the centre of the sensor was compared with the edges. This difference can be accounted for in practise by creating location-specific calibrations in software.

## Processing Hardware and Algorithm

The Arduino processor was used as a convenient prototyping platform, but the basic circuit design would be suitable for a variety of processors. In recent applications, we have made use of the ARM STM32 F4 and F3 series microprocessors. These processors have higher bit rates and are able to process the data across a broader range of resistance values.

Our visualizations of the sensor data were performed in Matlab. The below image represents a typical visualization – the greyscale squares become brighter and more white with increasing pressure on the sensor (decreasing resistance between layers), and remain black when the sensor is untouched (maximum resistance between layers).

## Electrical Connections

Each strip of the conductive fabric required an electrical connection and processor input/output. The interface between fabric and wiring was an early concern; there are a few known methods of creating a secure interface (see plusea.com), but they require a great deal of time-consuming labour and are not sufficiently repeatable for large numbers of connections. The solution we developed was to thread the strands of the stranded wiring through the fabric, twist it into a single piece, and fold the end of the wire the prevent untangling. A dab of solder paste or conductive adhesive can then be added for greater durability.



Figure 6 - Wiring method



Figure 7 - Mesh standoff layer and wiring sleeve

For many applications, it is desirable for the wiring profile to be kept as inconspicuous as possible. The sensor was therefore prototyped using ribbon cable, which was the most compact wiring system available to us. individual connection on the ribbon was split out and fastened to the fabric, and a main connector and jumpers were used to connect each section of fabric to a prototyping breadboard.

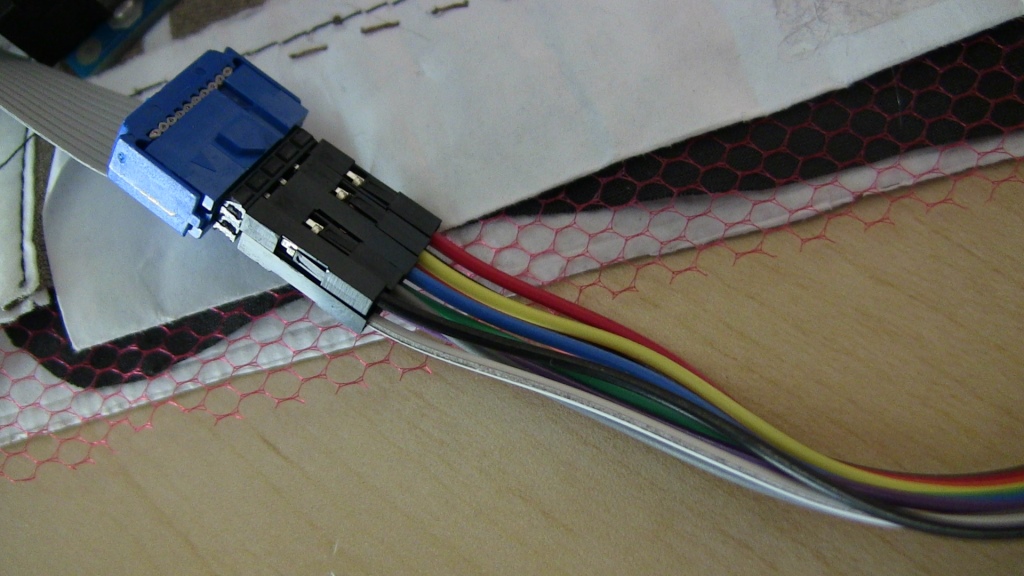


Figure 7 - Ribbon cable and jumper connection

The ends of the wires were threaded through the fabric and the entire cable was secured to the edge of the fabric using fabric stiffening material, which was sewn into place along the edges of the sensor. This created a channel for the ribbon cable and provided strain relief for the individual wires.

## Considerations for Physical Construction

### Sources of Interference

The main sources of interference in the sensor arose from small details in the construction which could result in a short. There are several points of note related to the construction, which will be outlined below.

### Pre-Loading

Pre-loading of the sensor occurs mainly when the fabric is stretched or a static force is otherwise applied to the sensor. In applications where the substrate the sensor rests upon is effectively incompressible and not in motion, pre-loading can be eliminated by careful assembly of the sensor, ensuring that the wiring and sewing of the fabrics do not overlap each other.

In dynamic applications, however, pre-loading is an issue which must be dealt with on a case-by-case basis, in some cases requiring software filtering to eliminate predictable noise.

### Cross-talk

Initial efforts at constructing a 5x5 grid sensor resulted in cross-talk, wherein contact with more than 1 point on the sensor would result in untouched pixels in a different region of the sensor registering pressure. If two or more fingers were touching the sensor, then, it would frequently occur that 1 or more pixels diagonally across from the additional points of contact would register a signal. This was hypothesized to be due to:

- Overlap of sensor connections in one grid direction with sensor fabric in the other direction

- Mid-layer (resistive) perhaps not wide enough ranging in ohmage

- Mid-layer perhaps too thin

- Spacing of bus lines (“zebra stripes”) too wide to get one-touch resolution - at the overlap of two stripes, two readings will appear.

Careful assembly and preventing the fabric stripes from contacting each other can reduce cross-talk occurrences. Early builds were sloppy for prototyping purposes, but the build process has gradually improved.

This ‘Ghosting’, or artefacts in the sensor matrix, are not able to be completely eliminated in our experience. They are the consequence of induced currents flowing as the result of Lenz’s law – for instance, two touched points on one conductive stripe in combination with a single touched point on another conductive stripe will result in an induced current to match the one-point stripe with the two-point stripe (since the stripes are roughly parallel to each other, the current flow is induced to “even out” the flow of current through the matrix).

### Noise due to shorts

Again, this type of sensor noise is preventable in static applications through careful construction of the sensor. However, in dynamic applications the sensor may be prone to folding, wrinkling, stretching or other effects which can induce overlap of electrical connections, causing shorts. Careful examination of the sensor's trajectory in these cases can allow the sensor to be constructed and mounted in such a way as to prevent these effects.

# CHARACTERIZATION PHASE

## Requirements for Gesture Recognition on Non-flat, Compliant Surfaces

In order to simulate the accuracy of much higher resolution touch data on our low-resolution sensor, we need our sensor to be able to recognize a range of touch pressures.  Based on a preliminary survey of touch pressures, our sensor must register 5 gram touches on the light end up to as heavy as 1 kilogram.

Preliminary testing of our sensor showed that under ideal conditions, we were easily able to achieve that range using 1000 ohm resistors.  Under the most severe conditions (for this study, that was the curved-foam substrate + thick fur cover), the lighter touches were lost in the dense fur cover and the heavier end equalized by the yielding foam substrate

We also found that using resistors much greater than 1000 ohms allowed our sensor to register much greater forces but would lose a lot of the finer variations as one increment of our 1024 grade scale represented a much greater change in pressure.  Conversely, while lower valued resistors gave us greater granularity in recognizing very fine touches, they caused the sensors to be overloaded too quickly to recognize the more forceful pressures.

In order to evaluate how our sensor would work in a real world context, we came up with 3 different masking variables involving deforming the sensor over a curve, covering the sensor with fur and fabric skins, mounting the sensor on a foam substrate. Finally, we asked human participants to perform 2 different tasks on all 12 combinations of the aforementioned conditions.

The first task involved projecting 6 simple geometric shapes in succession and asking participants to trace over the images.  In order to evaluate the integrity of the pressure data, we analyzed the mean square differences between sensor pixels that were directly activated (i.e. intentionally pressed) and the ambient reactions of adjacent pixels.

For the second task, participants performed 7 gestures over the same set of conditions.  We then ran the gesture data through 3 machine learning algorithms and analyzed how well we were able to classify the gestures.

# DISCUSSION

# CONCLUSIONS

# APPENDIX

**Sensor Sampling Algorithm**

* **Define variables for row and column size**
* **Select digital pin numbers for rows and columns**
* **Create a buffer constant for the data stream**
* **Create a struct to encapsulate the data going into the buffer – header (4 bytes), time stamp (4 bytes), row and column array (2 bytes for each ‘pixel’ of data – one each for upper and lower bound), checksum (1 byte)**
* **Initialize timer for desired sampling rate**
* **Initialize serial communication**
* **For loop – for each row, set the digital pin mode to output and the output to high impedance**
* **End for loop**
* **For loop – repeat as above for each column**
* **End for loop**
* **Initialize analog pins, assigning one for each column (no rows) – set pin mode to input**
* **Begin continuous loop**
  + **Access the buffer and read data offset**
  + **Write any existing data to the buffer**
  + **Write data to file or other output**
  + **Flush the buffer**
  + **Record the timestamp for the next set of data**
  + **Sample the sensor:**
  + **Main for loop – for each row, enable the pin and send a ‘high’ signal**
    - **Nested for loop – while a row is ‘high’, for each column, enable the pin and send a ‘high’ signal**
    - **While a column is ‘high’, delay 1 microsecond to prevent data artefacts**
    - **Read analog signal to associated analog pin**
    - **Save each ‘pixel’ of data into the buffer according to its row and column index**
    - **Add data to checksum**
    - **Set column pin to ‘low’ and disable**
    - **End nested for loop**
  + **Set row pin to ‘low’ and disable**
  + **End main for loop**
* **Repeat main loop**

**Sensor bill of materials**

- Eeonyx 'Zebra' fabric

- Eeonyx resistive fabric SLPA-20K

- Plastic mesh

- Resistors – 1K used

- Teensy 3.1 processing board

- Jumper wires

- Multiplexers: Texas Instruments CD74HC4067

**-** Ribbon connectors to circuit:

Male: TE Connectivity 215083-8 (http://www.digikey.ca/product-detail/en/215083-8/A111108CT-ND/4142492)

Female: TE Connectivity 188275-8 (http://www.digikey.ca/product-detail/en/188275-8/A107024CT-ND/3488585)

**Sampling speed with Arduino processor**

Some information on sampling speed of the grid can be deduced from the following:

<http://forum.arduino.cc/index.php/topic,6549.0.html>

http://www.marulaberry.co.za/index.php/tutorials/code/arduino-adc/

Here is a method of changing the Arduino’s prescale taken from that first link (prescale attempts to optimize (not necessarily maximize – resolution starts to degrade at higher freq) clock speed and sampling rate):

#define FASTADC 1  
  
// defines for setting and clearing register bits  
#ifndef cbi  
#define cbi(sfr, bit) (\_SFR\_BYTE(sfr) &= ~\_BV(bit))  
#endif  
#ifndef sbi  
#define sbi(sfr, bit) (\_SFR\_BYTE(sfr) |= \_BV(bit))  
#endif  
  
void setup() {  
  int start ;  
  int i ;  
    
#if FASTADC  
  // set prescale to 16  
  sbi(ADCSRA,ADPS2) ;  
  cbi(ADCSRA,ADPS1) ;  
  cbi(ADCSRA,ADPS0) ;  
#endif  
  
  Serial.begin(9600) ;  
  Serial.print("ADCTEST: ") ;  
  start = millis() ;  
  for (i = 0 ; i < 1000 ; i++)  
    analogRead(0) ;  
  Serial.print(millis() - start) ;  
  Serial.println(" msec (1000 calls)") ;  
}  
  
void loop() {  
}

**Associated documentation and links**

Previous graduate papers

<http://yohanan.org/steve/projects/haptic-creature/>

Plusea/Kobikant (Hannah Perner-Wilson)

<http://www.kobakant.at/DIY/>

<http://www.plusea.at/>

<http://www.instructables.com/member/Plusea/>

Existing sensors

<http://www.peratech.com/qtc-touch-processing-unit.html>

<http://vista-medical.com/subsite/stretch.php>

Fabric suppliers

<http://www.eeonyx.com/eeontex.php>

<http://www.adafruit.com/products/1361>

<http://www.lessemf.com/fabric.html#1209>

<http://www.openmaterials.org>

**Ongoing questions for sensor research**

What are the limits of curvature for optimizing the sensor?

What is the optimum pixel size?

What is the optimal size for sensor patches, and how many?

Does patch size impact accuracy of detection? How?

How much total surface area will be covered?

Is it necessary to cover the entire surface?

Can the sensor patches overlap?

Will there be several patches of different sizes?

How will the patch edges connect to the body?

How will patch edges connect to each other?

How can the sensor edges maintain their shape?

How much total I/O is needed?

How can sliding of fabric layers be prevented?

Can sliding of the fabric be accounted for/compensated for?

How can uneven preloading of the sensor be prevented?

Can uneven preloading due to stretch, internal pressure etc. be accounted for in code?

How should the data be read?

How should data be visualized?

What algorithms should be used for gesture sensing?

How will each user’s data be interpreted and distinguished?