Toward Accurate Sensing with Knitted Fabric: Technical Considerations and Application Potential

RICHARD VALLETT, Center for Functional Fabrics, Drexel University, USA
DENISA QORI MCDONALD, College of Computing and Informatics, Drexel University, USA
GENEVIÈVE DION, Center for Functional Fabrics, Drexel University, USA
YOUNGMOO KIM, ExCITe Center, Drexel University, USA
ALI SHOKOUFANDEH, College of Computing and Informatics, Drexel University, USA

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

Fig. 1. Examples of the knitted capacitive touch sensor created through our manufacturing process: *(a)* A planar knitted fabric touchpad made from polyester and carbon-suffused nylon connected to two electrodes. *(b)* The sensing circuit integrated into a knitted sleeve. *(c)* Button snap fasteners used to connect the textile to an external sensing controller. *(d)* Oscilloscope visualization of the voltage input and output from the fabric circuit.

Fabric sensors have been introduced to enable flexible touch-based interaction. We advance the technical capabilities of a scalable and low-profile knitted capacitive touch sensing system by introducing methods to improve its touch localization accuracy. The sensor hardware design tends toward minimalism by using a single conductive yarn and two external connections located at each endpoint. Fewer connectors simplify the textile system integration, but this comes at the expense of reduced signal information output from the system. The electrical continuity of the sensing element, essential to the process of knitting, also increases the uncertainty of localizing touch. We propose using Bode analysis to measure changes in signal due to capacitive touch, as well as design a new algorithm, *Mixed-Source Description (MSD)*, which retains the most significant aspects of the signal in terms of touch location identification. We do not classify location of touch, but focus on an invariant signal representation. To evaluate our methods, we introduce *Euclidean Levenshtein Distance (ELD)*, a distance metric to compute the similarity of pairs of key-presses, generalizable to computing

Authors' addresses: Richard Vallett, rjvallett@drexel.edu, Center for Functional Fabrics, Drexel University, 3101 Market St. Philadelphia, Pennsylvania, USA; Denisa Qori McDonald, dq38@drexel.edu, College of Computing and Informatics, Drexel University, 3675 Market St. Philadelphia, Pennsylvania, USA; Geneviève Dion, Center for Functional Fabrics, Drexel University, 3101 Market St. Philadelphia, Pennsylvania, USA; Youngmoo Kim, ExCITe Center, Drexel University, 3401 Market St. Philadelphia, Pennsylvania, USA; Ali Shokoufandeh, College of Computing and Informatics, Drexel University, 3675 Market St. Philadelphia, Pennsylvania, USA.

2573-0142/2020/6-ART79 $15.00
https://doi.org/10.1145/3394981

distances of tensors of varying lengths. Our experiments show that the proposed sensing method results in high-fidelity signals. Furthermore, the sparse representation of key-presses produced by *MSD* significantly increases separability between different touch locations. Possible applications based on these sensors are also illustrated through prototypes and use case descriptions.

CCS Concepts: • **Human-centered computing → Systems and tools for interaction design**; *Interaction devices*; *Ubiquitous and mobile computing*; • **Computing methodologies**;

Keywords: knitted sensor, knitted fabric, capacitive touch sensor, signal analysis, time series, sparse representation, SIFT, carbon fiber, digital weft knitting, e-Textiles, Scale-Invariant Feature Transform, Bag-of-Temporal-SIFT-Words, BoTSW

ACM Reference Format:
Richard Vallett, Denisa Qori McDonald, Geneviève Dion, Youngmoo Kim, and Ali Shokoufandeh. 2020. Toward Accurate Sensing with Knitted Fabric: Technical Considerations and Application Potential. In *Proceedings of the ACM on Human-Computer Interaction*, Vol. 4, EICS, Article 79 (June 2020). ACM, New York, NY. 28 pages. https://doi.org/10.1145/3394981

1 INTRODUCTION

Textiles are among humankind's first inventions and have since evolved beyond uses for comfort and protection. Advances in materials, manufacturing, sensing and computing have given way to novel interactive systems supported by soft, deformable textile touch sensors. Prototypes of fabric-based interfaces have been constructed that emulate well-established hard electronics designs and rely on similar sensing principles. Like commercial touchscreens, many fabric-based touch sensors measure input using a grid-like structure of wires integrated into the textile substrate—specifically, sensors constructed from woven textiles [23]. Though grid-like patterns easily discretize touch across a surface, this structure requires many small and fragile fabric-to-wire connections at points where the conductive yarn affixes to hard electronics. This complexity may limit the sensor's durability when folded or stretched. Knitting has great potential to produce fabric sensors, since it can create complex, flexible and integrated structures using few individual continuous yarns. However, the electrical continuity of conductive yarns in knitted textiles limits applications of discrete distributed sensing. Thus, knitting is currently less-utilized as a smart textile production technique than either weaving or embroidery.

Previous knitted touch sensors utilize layers of knitted electrical traces sewn together as a grid [21]. In this work, we investigate distributed touch sensing using a planar knitted capacitive sensor able to be manufactured at scale using digital weft knitting, first introduced in [30] (Figure 1a). This design relies on one conductive yarn and only two fabric-to-wire connections at yarn endpoints (Figure 1c), simplifying connection to sensing electronics and improving wearability by making the textile more flexible and durable. However, sensing techniques in [30] only approximate general areas of touch.

We introduce the use of Bode analysis towards *differential capacitive sensing* [30] to improve the precision of measured data (Figure 1d). Furthermore, we introduce *Mixed-Source Description*, our algorithm to extract invariant features from time-series signals, which sparsely represents the captured signal in a more meaningful way regarding touch location identification. In this work, we do not classify location of touch. Rather, these contributions improve *touch location representation* while maintaining compatibility with existing knitted textile sensors. We conduct a 13-subject user study to evaluate our methods. In order to determine the level of similarity between different samples of touch signal representation, we introduce the *Euclidean Levenshtein Distance* algorithm. Furthermore, we explore the usability of knitted capacitive sensors as interfaces (Figure 1b) by presenting application prototypes and discussing potential uses.

The contributions of this paper to the state-of-the-art are as follows:

(1) The application of Bode analysis towards *differential capacitive sensing* to localize touch along a single, low-conductivity electrical pathway.
(2) The development of *Mixed-Source Description (MSD)*, an algorithm to detect invariances in the scale-space of multi-channel signals.
(3) The introduction of *Euclidean Levenshtein Distance (ELD)*, a generalizable distance metric which measures the similarity between two tensors of varying lengths.
(4) The introduction of application prototypes created from weft knitted capacitive touch sensors.

After discussing related work in Section 2, we discuss the knitting process and properties of the sensing circuit in Section 3. We then describe our *differential capacitive sensing* method and compare it to conventional methods of capacitive touch sensing in Section 4, followed by a description of the signal acquisition pipeline in Section 5. In Section 6, we introduce *MSD*, an algorithm to represent touch input events in terms of sparse, invariant features. Section 7 focuses on *ELD*, an algorithm to compute similarity between pairs of sequential data describing touch events. Subsequently, we discuss our experimental procedure and user study in Section 8 and its results in Section 9. We then focus on possible applications of this technology and their integration in Sections 10 and 11, followed by limitations in Section 12 and conclusion in Section 13.

2 RELATED WORK

Many modern human-computer interfaces measure input via capacitive sensing [4]. A capacitive sensor sources voltage or current through a conductive wire or trace and measures the charge stored between it and conductive objects in proximity. Capacitive sensors can detect input across an interface at single locations by polling individual wires or conductive areas (Figure 2a). Additionally, wires can be interspersed horizontally and vertically to form a matrix to localize distributed touch (Figure 2b). Capacitive sensing circuits can be formed on printed circuit boards or glass and have an advantage of high sensitivity, low cost and low-profile compared to resistance-based sensors.

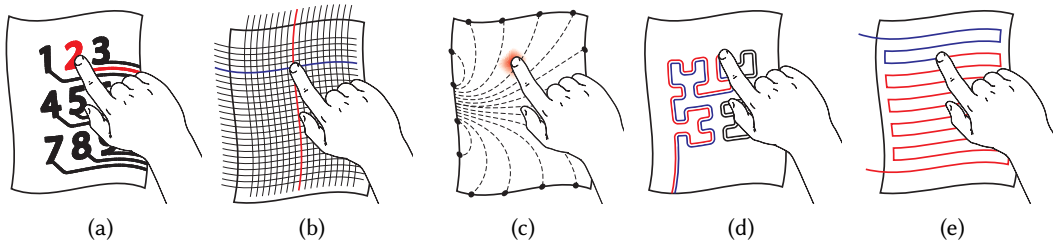


Fig. 2. Illustrations of capacitive sensing techniques used in textile touch sensors: *(a)* Embroidered capacitive buttons. *(b)* Woven or embroidered capacitive sensing matrix. *(c) Electric Field Tomography* measured across a continuous conductive fabric surface. *(d) Reflectometry* measured using two parallel conductors. *(e)* Current difference measured along a single weft knitted conductor using *differential capacitive sensing*.

Capacitive sensing as a strategy for touch detection has been extended to devices that use textiles as a physical interface. Many fabric-based capacitive touch sensors measure input by self-capacitance from either a change in proximity between conductive yarns or through human

skin contact. Textile-based capacitive touch sensors have been constructed using embroidery [7–9, 22] (Figure 2a), woven textiles [1, 10, 23, 27, 29] (Figure 2b), sewn layers of conductive knitted textiles [26] and seamless machine knitted textiles [20].

Many fabric-based distributed touch sensors achieve measurement simplicity at the cost of design complexity—namely in the connection and management of numerous individual wires. As in commercial devices, the wires exiting the physical sensor are multiplexed to a sensing controller which sequentially measures the capacitance of each wire. More sophisticated sensing strategies have been proposed to measure both distributed touch location and pressure across continuous, pattern-less surfaces using a minimum number of fabric-to-wire connections. *Electric Field Tomography (EFT)* [28] is a capacitive sensing method which attempts to localize touch within a bounded area by sourcing and measuring current at the periphery (Figure 2c). EFT estimates touch location by attempting to reconstruct current flow across the conductive surface and infer the number of contact points. EFT has been extended to irregularly-shaped or deformable interfaces with potential applications towards textiles [35].

Single-wire continuous capacitive touch sensors have also been investigated using applied transmission line theory (Figure 2d). These sensors measure the distance to touch locations using either *Time Domain Reflectometry (TDR)* or *Frequency Domain Reflectometry (FDR)*. Capacitive sensors have been constructed that localize touch using TDR along both single-wire and parallel-wire transmission lines. The transmission lines are affixed to flexible substrates to enable interaction on a variety of surfaces [33]. Fabric transmission lines have also been constructed to measure touch via deformation using FDR [11].

In this work, we further the technical capabilities of the knitted capacitive touch sensor, proposed in [30], aiming to localize touch across a single wire (Figure 2e). The sensor is assembled as a complete structure using weft knitting. The conductive yarn is integrated in a serpentine pattern natural to weft knitted textiles (Figure 3a). The capacitive sensing method shares similarities with both tomography and reflectometry by measuring continuous touch across a resistive, linear pathway. However, touch input is measured through a differential in current sourced from either endpoint of the conductive yarn to the touch location. The supplementary microprocessor used to generate and measure voltage can coarsely localize single points of touch across the yarn using a simple heuristic but is susceptible to electromagnetic interference due to the low current sourced and a lack of grounding near the sensing pathway. The capacitive sensing method demonstrated in [30] shares similarities with techniques used to measure self-capacitance. A microprocessor generates an oscillating input voltage and measures the transient response of the output voltage as the circuit charges and discharges. To resolve better touch localization, we use Bode analysis [5] to measure the effects of touch on the output at individual frequencies. Bode analysis is a control systems technique used to characterize systems based on observed input and output. Because we can control the frequency of the input signal, we can selectively reject frequencies where additive interference is present.

We further improve the accuracy of localizing gestural touch input by applying scale-space analysis to extract invariances from the time-series differential signal to more succinctly represent the action therein. Scale-Invariant Feature Transform (SIFT) [19] models have improved object recognition due to their ability to extract image features invariant to changes in scale, rotation, noise and limited variation in viewpoint. However, SIFT was designed to process 2-dimensional data and optimized to detect features associated with images. Therefore, it is not directly applicable to 1D time-series sensor data. However, most of its extraction steps are based on mathematically abstract and generalizable concepts, enabling it to serve as a basis for algorithms that process sequential data. Bag-of-Temporal-SIFT-Words (BoTSW) [3] is an algorithm that applies SIFT concepts to 1D time series data. Our algorithm, *MSD*, detects scale-space invariances in sequential signals using a

process similar to that of BoTSW but does not incorporate the Bag-of-Temporal-Words approach. Instead, one time-series signal is represented as a temporally-related sequence of key-points, rather than a single vector as in the case of BoTSW. Furthermore, *MSD* is generalizable to more than one input sequence and can be extended to multiple sources by accounting for the coupling between them when identifying invariances or key-points. Additionally, *MSD* individually normalizes the coupled key-point descriptors to improve representation.

Our distance metric, *ELD*, is adapted from the Levenshtein Distance (LD) [15]—a general dynamic programming metric used for sequence comparison; most frequently for string comparison. Similar to LD, *ELD* compares two variable-length sequences element-by-element. In contrast to LD, which assigns each edit an equal weight, *ELD*'s assigned weight per comparison depends on the Euclidean Distance, which makes each comparison more nuanced. *ELD* can be used to compare any numeric tensors. Dynamic Time Warping (DTW) [14, 25] is another established and generalizable distance measure which finds the optimal match between two sequences. It considers the comparison between two signals as "time-warped", where one is a condensed, stretched, or non-warped version of the other. In DTW, each point in one sequence matches one or several corresponding points in the other, making the algorithm more resilient to varying sampling rates. In *ELD*, gaps are allowed in the sequence which allows it to be more robust to noise.

3 WEFT-KNITTED SENSORS

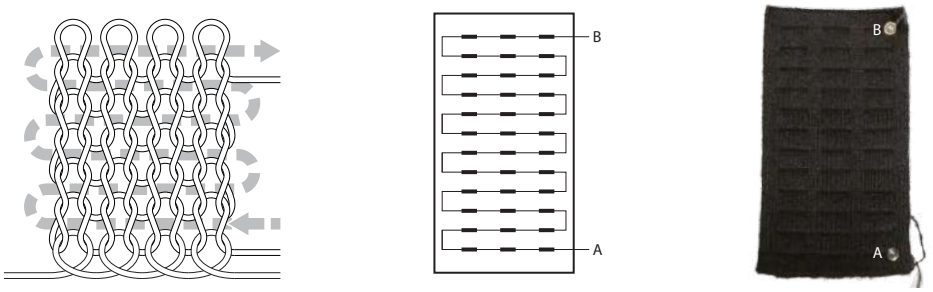


Fig. 3. Weft knitted touch sensor design: *(a)* Illustration of the serpentine pathway formed by yarn during weft knitting. *(b)* Illustration of the conductive yarn pathway knitted during touch sensor construction. *(c)* Image of a complete knitted touch sensor.

Digital weft knitting is a versatile, end-to-end textile manufacturing process that lends itself well to rapid prototyping. Textile designers can create finished pieces without the need for additional assembly using robust programmed instructions standardized across many digital knitting platforms. We leverage the extensibility of weft knitting in the creation of textile interfaces by designing a sensitive yarn layout compatible with the knitting process. Weft knitting, unlike weaving, uses comparatively few yarns that do not separate at the fabric's edge and are continuous throughout the textile (Figure 3a). This layout is incompatible with conventional sensing matrices which require isolated traces. Therefore, in order to sense distributed touch across a knitted fabric, we must measure input across a continuous yarn integrated within the textile.

In this work, we use a knitted sensor in which the pathway of the conductive yarn, illustrated in Figure 3b, follows the natural serpentine of weft knitting shown in Figure 3a. This design constraint simplifies the physical construction of the sensor and improves the durability of the textile, while

shifting the burden of complexity to more refined sensing and computational models. Sensors produced using this process have one conductive yarn and two fabric-to-wire connections located at the two endpoints of the yarn path (Figure 3c). This layout greatly reduces the number of connections otherwise required to measure distributed touch. Furthermore, the structure of the conductive yarn path is diverse and can enable many design forms.

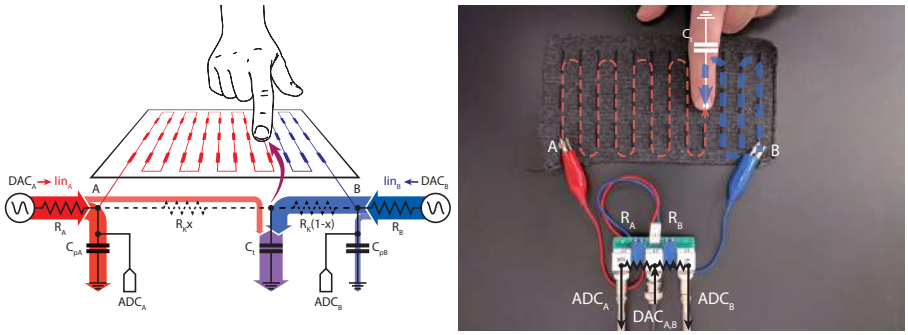
Touch location is inferred based on the resistance of the conductive pathway. Controlling its resistance is crucial to ensuring separable touch data, tuning external current-limiting resistors and predicting signal behavior. The capacitance induced by human touch varies within the range of pico-Farads. Accurately measuring small values of capacitance and fine changes in current across the conductive pathway requires a high yarn resistance. The overall resistance of the pathway is set through modifying the yarn density (denier) or through modifying properties of the knitted pattern. Yarn denier, which is the mass or density of a given length of spun yarn, can be altered to produce an appropriate cumulative resistance. In the case of the carbon fiber trace, the yarn may be represented as a wire of homogeneous composition having a linear resistance, R. The wire resistance is a function of the resistivity, ρ, an intrinsic material property, its length, l, and fiber cross-sectional area, A. The resistivity is set during manufacturing and varies by production lot. The yarn resistance per length increases or decreases by respectively decreasing or increasing the yarn's cross-sectional area. The resistance changes proportionally by length. In practice, significantly decreasing the denier proves difficult to knit due to the fragility of the single-strand monofilament yarn. Combining a lower denier monofilament with a non-conductive yarn improves knitting but sacrifices homogeneity and complicates yarn interconnections.

Specifying the overall sensor resistance through loop formation is not as straightforward as specifying the fiber resistance through length or thickness. The loops formed during the knitting process create a complex, interconnected mesh of resistors [34]. Understanding the interactions between the loops is crucial to better predicting the touch sensor resistance during the design process. Specifying the width and height of the touch points alters the resistance intuitively. Widening the trace decreases the resistance by increasing its cross-sectional area. Lengthening the trace increases the resistance. Knitting techniques like interlocking are used to knit fractional gauges and can significantly decrease the knitted resistance within a small area. Interlock rows deposit yarn on multiple passes, thereby increasing the area of the trace and decreasing the linear resistance. Interposition of non-conductive yarn may also alter the resistance of the conducive trace by separating the electrical connections between yarn loops. The sensor in Figure 3c, used in our experiments, is a 4 inch × 8 inch touchpad with 36 points of contact, or "buttons", and a cross-knitted resistance of 550 kΩ. The buttons in this design are spaced approximately 2/3 inches apart. The inter-button resistance is approximately linear at 15 kΩ between button center-points.

The yarn used to produce the knitted sensors is a 40-strand, 44-denier carbon-suffused nylon monofilament, *Resistat® F901, Merge D044* [12], for use in high-wear applications. The fibers received have an average diameter of approximately 30 microns with a reported surface coating thickness of 1 micron and an average linear resistance of 972 kΩ/cm. The non-conductive base yarns used are a 50/50 natural/synthetic blend *Primaloft®* or a polyester synthetic monofilament. A polyester-based heat-melt yarn is used to steam-seal the finishing loops. A high-bulk nylon is used to emboss the conductive yarn pathway and provide surface texture.

4 SENSING PRINCIPLES

Previous work [30] describes a weft knitted capacitive touch circuit that is sensitive to touch location across a continuous conductive pathway. A current differential is measured using conventional capacitive sensing techniques that are susceptible to distortion, which limits the overall location



(a) (b)

Fig. 4. *(a)* Circuit diagram of the fabric capacitive touch sensor and sensing circuit: The knitted resistor, R_K forms two series-connected resistors at the contact point. *(b)* An overlayed image showing a real knitted touchpad and connections to sensing hardware.

resolution. The contributions of signal processing to this work improve the precision and separability of measured data and provide a solid foundation for invariant feature extraction.

4.1 Differential Capacitive Sensing

The digital weft knitting process can produce complex textiles with seamless integration of sensitive yarns. Though numerous designs can be knit, the interlocking nature of yarn loops and the requirement that yarn remain contiguous throughout the textile pose unique challenges when attempting to route independent conductive yarns through knitted fabrics. A solution to measuring distributed touch across a weft knitted fabric is to route yarn following the natural weft serpentine and measure touch along the length of the contiguous path. The non-branching serpentine maps all touch locations along the surface of the fabric to a linear arrangement of points. Any touch location can be described by the resistances from the touch point to either yarn endpoint.

Figure 4a depicts current flow from either endpoint to a touch location on the fabric. A synchronized voltage waveform is input at either end of the fabric using a *Digital-to-Analog Converter (DAC)* at points DAC_A and DAC_B and passed through two equivalent current limiting resistors, R_A and R_B. The current output at points A and B branch both into the knitted circuit and towards the voltage measurement circuit. The parasitic capacitance of the measurement circuit contributes to the response recorded by two *Analog-to-Digital Converters (ADCs)* at points ADC_A and ADC_B.

Figure 5 shows example input and output voltage waveforms recorded by an oscilloscope. Figure 5a depicts the resting state of the voltage signals when no touch is present. Figure 5b shows the attenuation caused by touch occurring near the left (A) electrode. The capacitance induced by touch decreases the peak-to-peak amplitudes of both outputs and produces a greater phase shift. The electrode closer to the point of touch exhibits a more pronounced attenuation and phase shift. If the touch event occurred closer to the right (B) electrode, we would expect a response analogous to that in Figure 5b, but with the right electrode having greater attenuation. In the case of contact occurring in the center of the circuit, the response would look similar to that shown in Figure 5a, since the attenuation from both electrodes would be approximately equivalent. Figure 5c shows the effects of EMI distortion in addition to contact. The difference in peak-to-peak amplitude is present but obscured by high-frequency distortion.

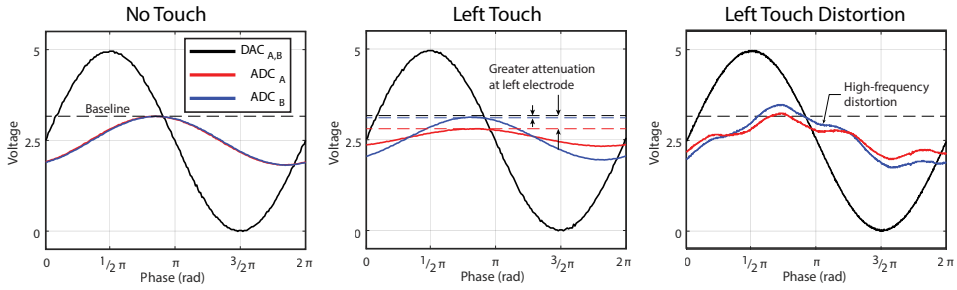


Fig. 5. Example input and output voltage waveforms: *(a)* Input and output without touch. *(b)* Touch occurring near the left (A) electrode. *(c)* Left touch distorted by a nearby EMI source.

Conventional capacitive sensors that measure attenuation or phase shift do so by directly tracking the voltage envelope or measuring the time difference between crossing voltage thresholds. Measuring phase offset with minimal distortion yields measurements that are relatively consistent. However, in the presence of additive interference, the variance of phase increases considerably.

4.2 Measuring Differential Capacitance using Bode Analysis

Elevating the performance of the knitted touch sensor requires resolving fine changes in capacitance in real-time. The measurement strategies used by conventional capacitive sensing controllers do not account for distortion while capturing the voltage waveform. Instead, distortions present within the time-series output are processed using a windowed moving average or convolution to remove high-frequency variations and improve touch threshold detection. Filtering measured data does not improve the overall quality of the data captured and, in the case of measuring the strength of capacitance, distortion within measured data decreases the effective resolution of touch localization.

As an example, we observe the waveforms recorded from input at adjacent touch locations on the yarn separated by approximately 30 kΩ. A 10 kHz sine wave with a peak-to-peak voltage of 5V is input through both current limiting resistors. Voltage is sampled at approximately 8.14 MHz and 815 points per frame for 1000 frames.

Figures 6a and 6b plot the distribution of data measured through phase analysis. Figure 6a shows data measured with negligible interference while Figure 6b shows the effects of measurement distortion from a nearby fluorescent light bulb outputting interference above 40 kHz. The distortion in the time measurements contribute to the larger variance between Figures 6a and 6b. As we can see, the touch location data information overlaps in both cases, especially under higher-interference conditions.

By applying Bode analysis toward differential capacitive sensing, we aim to remove the variance present in data measured by conventional capacitive sensing. Fine changes in capacitance are measured by sampling the continuous voltage waveform over time, performing a Fourier transform and and extracting the gain ratio from the measurements at the input frequency bin. This process allows us to selectively discard all other frequencies that are not present within the input. In this example, a single frequency is input. Measuring one full period of the voltage signal yields information at the first frequency bin at 10 kHz. Figure 6c shows the clustering of processed data without induced noise, while Figure 6d is produced in the presence of the same fluorescent light

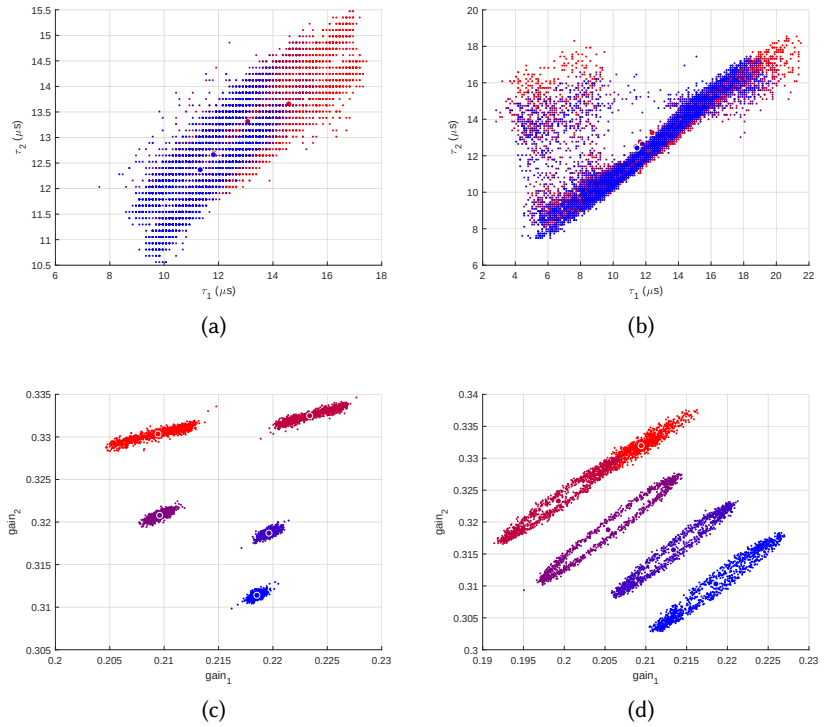


Fig. 6. Comparison of phase and Bode analysis: *(a)* Phase analysis without EMI distortion. *(b)* Phase analysis with EMI distortion. *(c)* Bode analysis without EMI distortion. *(d)* Bode analysis with EMI distortion.

exposure as in the case of Figure 6b. The clusters of points are easily separable in Figure 6c, and that separability is still retained even in the presence of noise, since the clusters do not overlap. However, the data still has some spread and not completely accurate localization ability. We plan to further move toward accurate localization through our sparse, invariant signal representation, described in Section 6. In the section below, we give more details about the capacitive signal acquisition process and its post-processing techniques.

5 SIGNAL ACQUISITION

The signal processing pipeline, illustrated in Figure 7, describes the process of generating an excitation waveform, measuring the direct output from the textile sensor and converting sensed data into amplitude gains for our chosen frequency.

5.1 Waveform Capturing

We record voltage waveform data using a Keysight MSOX-3024A 4-channel mixed-signal oscilloscope. Channel 1 measures the input voltage waveform. Channels 2 and 3 measure the A and B voltage outputs. Channel 4 measures the sample trigger. A Keysight 33622A function generator is used to generate both the voltage input and the sample trigger. We input a 20 KHz sine wave at both fabric endpoints. The measurement window spans 400 μs with 1 ms between sample window triggering. Each window captures 8 periods of the input and output waveforms at approximately

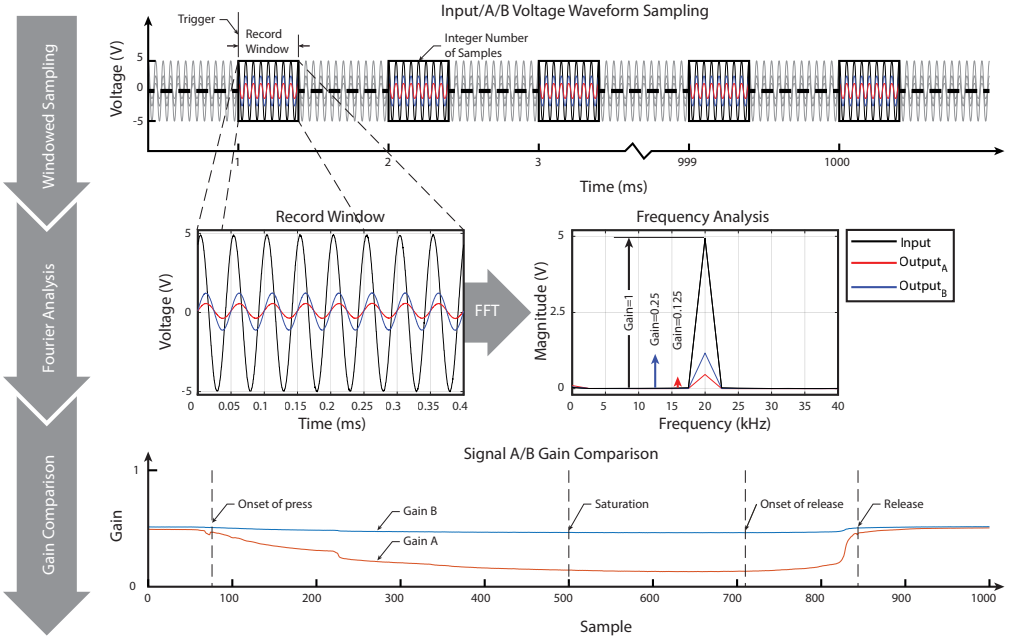


Fig. 7. Pipeline for waveform analysis: The input and output waveforms are recorded in segments which are individually processed. The spectrogram of the record window is computed to find the single-sided amplitude of the input and outputs. The ratio of input to output is computed as the output gain.

4.06 MHz (1627 points per waveform). During each touch event, 250 sample windows are recorded over the duration of 2 seconds.

5.2 Post-Processing

Fourier analysis of each sample window is conducted during the second phase of the pipeline to determine the voltage magnitude present at the given input frequency. Because an exact integer number of periods (8 periods) is sampled within the window, the output of the Fourier transform will the be present at the 8^{th} frequency bin. In embedded applications, this detail simplifies analysis of the waveform by placing all of the relevant data at an exact bin location.

The magnitude values of the input and two outputs are used to compute the normalized voltage gain of outputs A and B in the third phase of the pipeline. $Gain_A$ and $Gain_B$ are computed as $Output_A/Input$ and $Output_B/Input$. The plot in the third phase shows the attenuation during a press-and-release event on the left side of the sensor. The dip in gain is much greater towards the sensing electrode at A than at B.

When $Gain_A$ and $Gain_B$ are plotted orthogonally in Figure 8, it is possible to obtain a better understanding of their relationship. The points plotted by the gain pairs fan out from a point at the top-right of the plot. As the touch pressure increases, the differences in touch position become more pronounced. While it is possible to use analytical heuristics to decouple the touch pressure and touch position, we propose more robust decoupling methods, detailed below.

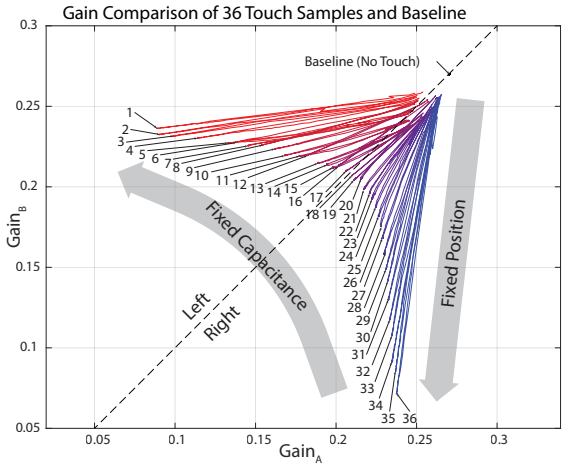


Fig. 8. Comparison of $Gain_A$ and $Gain_B$ of 36 touch points and the baseline (no touch).

6 INVARIANT FEATURE CONSTRUCTION

As described in Sections 4 and 5, two independent waveforms are recorded from the sensor—one at each end of the conductive yarn. We build our features based on the gains of the voltages levels from each signal measured over time. The voltage gains are rendered using Bode analysis and saved as a 250×2 matrix of time-series data. Our approach, *Mixed-Source Description*, produces a sparse representation of binary signal time-series data, aiming to capture the relevant information and discard noise. *MSD* is based on Bag-of-Temporal-SIFT-Words (BoTSW) which detects invariances in the scale-space of each signal and is adapted to incorporate a joint description of the two waveforms. The scale-space is a multi-scale representation of a signal. Scale space enables the detection of structures at different scales, where a new scaled version of the signal is achieved by smoothing the original one. At each scale, its salient features are extracted as key-points and then locally described. The BoTSW adapts SIFT to 1-dimensional time series signals instead of 2-dimensional images, for which SIFT was originally developed, while also incorporating the concept of Bag-of-Words from natural language processing. *MSD* does not use that concept in its representation.

All the steps of our approach are further explained below. The algorithm workflow is illustrated in Figures 9 and 10. The key-point detection step in Figure 9 includes one of the main differences between *MSD* and BoTSW: if a key-point is detected in one of the signals, its associated point in the other signal is assigned as a key-point as well. Figure 10 illustrates another difference with BoTSW, with descriptors for each key-point being normalized, to be subsequently concatenated. Both SIFT and BoTSW, and as a result *MSD*, rely on some relatively well-known concepts such as Gaussian scale-space and Difference of Gaussian. A brief overview of these concepts is provided in Sections 6.1 and 6.2 for clarity and completeness. For more details on the existing algorithms, please refer to [19] and [3].

6.1 Gaussian Blurring

Each of the two 1D signals present in one key-press were blurred separately in five different scales (σ) using a Gaussian kernel $G(t, \sigma)$ [2, 16, 18] of size 250 and standard deviation σ, corresponding to the size of the original signal interval. Blurring refers to the operation of performing convolution of the original time series signal interval $S(t)$ with the Gaussian operator, to produce $L(t, \sigma)$ as

shown in Equation (1).

$$L(t, \sigma) = G(t, \sigma) * S(t) \tag{1}$$

The scales selected for our experiments follow the suggestions of the original SIFT algorithm [19] and differ from each-other by a factor of $k = \sqrt{2}$, starting with $\sigma = 0.5$.

6.2 Difference of Gaussian

In order to find the areas of rapid change in the signal, the Laplacian, which is the second order derivative of the functions, needs to be computed on the blurred signal. Since the Laplacian is sensitive to noise, smoothing the signal through blurring helps stabilize its representation. In order to preserve scale-invariance, the Laplacian of Gaussian needs to be scaled by σ^2. Computing the Laplacian is expensive—however, it can be closely approximated by the Difference of Gaussian (DoG) operation [17, 19]. The DoG at a particular scale can be computed by subtracting blurred signals of subsequent scales from each-other, as shown in (2).

$$D(t, \sigma) = L(t, k_{sc}\sigma) - L(t, \sigma). \tag{2}$$

A signal filtered with a Gaussian of the scale of DoG being computed is subtracted from that same signal filtered with a scale k_{sc} times greater, which is the next scale up, as shown in Figure 9 (b). Since we blur the signal in five different scales, there are four DoG signals produced for each time-series measurement.

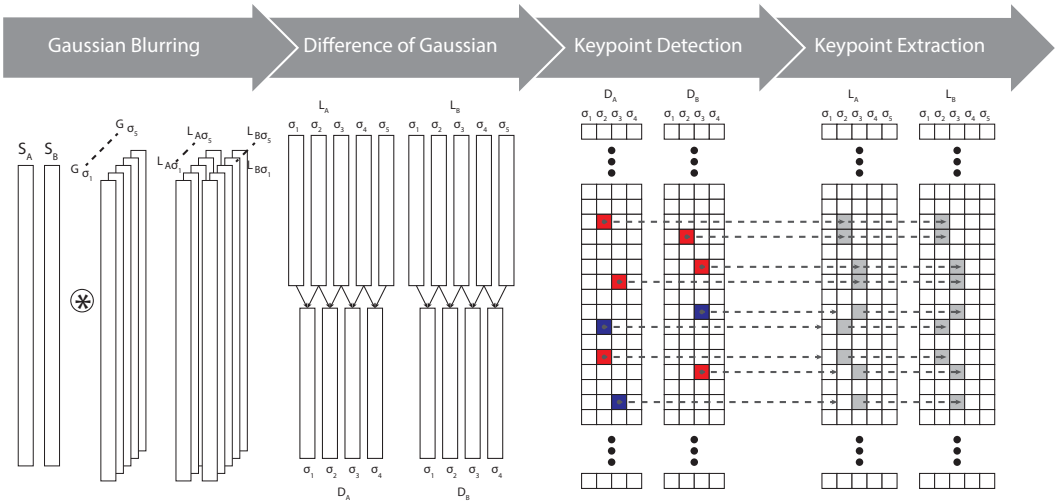


Fig. 9. Invariant feature construction process for each interval of two signals: (a) : Gaussian blurring—individual kernel application to each sensor data to produce filtered signals in different scales (b) : Difference of Gaussian—computed by subtracting pairs of consecutive filtered signal data (c_i) : Key-point detection—checking the DoG signals for points that are local maximums or minimums (c_{ii}): Key-point extraction—Finding the key-point according to its co-ordinates of position t, and scale σ in the filtered signal pair, where the point of the same co-ordinates in the other signal is considered a key-point as well, and extracted.

6.3 Key-point Extraction

Key-points are discovered from the DoG representations at a particular scale σ. Specifically, a key-point is defined as an extremum, which means that that point is the maximum or the minimum

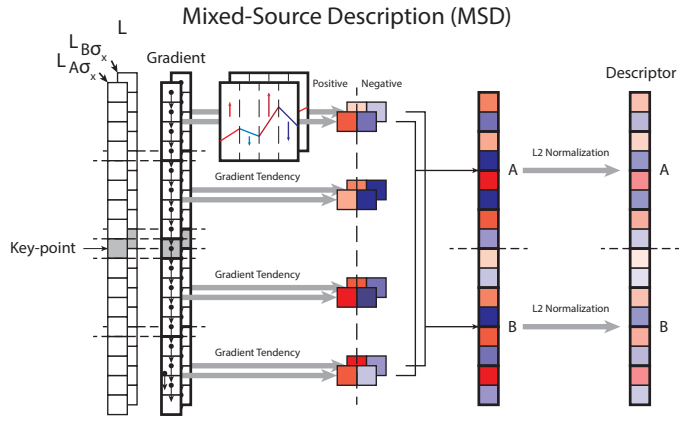


Fig. 10. Construction of *MSD* descriptors.

of the points in its neighbourhood. A point's neighbourhood is comprised of points in its own scaled DoG representation, $D(t, k_{sc}\sigma)$, as well as points in the DoG representations of one scale below $D(t, k_{sc-1}\sigma)$ and one above $D(t, k_{sc+1}\sigma)$.

For each 250-instance interval corresponding to a key-press, every point in the scale-space of each of its 1D signals is compared to its neighbourhood. If the point in the DoG representation is detected to be a key-point, its position t and scale σ are retained. The neighbourhood of each point is composed of one point before and one point after itself in its own scale, as well as the three corresponding points to the previously mentioned ones in the scale above, and the scale below. In total, each point of every DoG is compared to eight other points.

Our dataset is composed of two signals measured simultaneously, and each signal interval's DoG is searched for key-points separately, as in BoTSW. If a key-point is discovered in one, the corresponding position t and scale $k_{sc}\sigma$ are retained to be subsequently described.

6.4 Descriptors

After key-points are detected, each is used as a coordinate of position t and scale $k_{sc}\sigma$ to a point in the filtered signal representation $L(t, k_{sc}\sigma)$. That point located in the filtered signal is then described in terms of a fixed number of other points that surround it in that scale space.

Blocks are defined by grouping some of these points together. For our key-point descriptors, we use a total of 4 blocks of 4 points each, with two blocks sequentially before the key-point and two blocks after. For a 1D signal, the neighbourhood consists of a total of 17 points: 2 blocks of 4 points each before the point, the point itself, and 2 blocks of 4 points after the point. Next, gradients are computed for every point of the 17×1 vector, to produce another vector of the same size. Within each block of the gradient interval, positive gradients are summed together, as are negative ones. These two values, the sum of its positive gradients and the sum of its negative ones, are used to describe one block. A point is represented by the descriptors of all of its blocks, resulting in a feature vector whose size is double the number of blocks: 8 in this case.

6.4.1 MSD Descriptors. BoTSW uses one time series signal over which it extracts key-points and computes descriptors as described above. We apply the same principle and, additionally, use the fact that there are two simultaneous signals for each key-press to capture inter-dependencies between them. If a key-point is discovered in one of the signal's filtered representations, L_σ, the

corresponding point is locally described within that 1D signal. In our algorithm, additionally, the point of same position t and scale σ in the other signal is considered a key-point by association, even if it has not been detected as an extremum. That key-point is also described the same way within its own signal column. Next, the descriptors of these associated key-points of the same t and σ, but different capacitive sensor sources, are each individually normalized using *L2-normalization* to increase the stability and robustness of representation. They are then concatenated to create a 16D feature vector. The computing of *MSD* descriptors is shown in Figure 10. In BoTSW [3], each key-point is described only within its neighbourhood—since there is no other coupled signal from which to define an associated key-point, and it is not normalized.

The reason for joint description in *MSD* is the fact that the two signals are related, since they are produced by the same touch input measured at either end of the conductive yarn. Therefore, if an invariance is detected in one of these signals, the state of the other at that point in time becomes important as well. The produced descriptor in our case contains cumulative information from two different waveforms, but this method is extensible to signals from multiple sources related in time.

As a result of *MSD* computation, a key-press is represented by a sequence of descriptors. The size of each descriptor is fixed for every key-press, and it is greater than the size of any single sample from the original key-press by a factor of 8. The number of descriptors per key-press varies, since it depends on the number of detected key-points for that key-press. In any case, the *MSD* representation of a key-press is sparser than that of the original signal, resulting in a shorter sequence.

6.5 Codebook Generation

BoTSW contains the extra step of *codebook generation* in feature construction, which is omitted in *MSD*, so we are better able to analyze the quality of generated descriptors. Codebook generation, consists of first clustering all key-point descriptors of the data set using the K-Means algorithm, then calculating for each interval the distribution of its key-points over the clusters. This distribution would constitute the feature vector of each interval. Thus, a matrix of dimension 250×2 would be represented by one vector of size equal to the number of clusters selected for clustering. While this approach would still give insight into the data-set, the number of samples to analyze would be considerably reduced. Moreover, as is the case with K-Means clustering, the number of clusters would need to be picked experimentally, without any particular reason applicable to this type of problem. This would obscure the answer to our primary question of whether key-points described in terms of their neighborhood are a good choice for feature construction.

7 DISTANCE METRIC

In order to measure similarity between representations of key-presses, we modify the Levenshtein Distance [15], a similarity metric used to measure the difference between two sequences. We investigate how different key presses that belong to the same button, which is a position along the sensing yarn on the knitted pad, relate to each-other and those of other buttons.

7.1 Levenshtein Distance

The Levenshtein Distance, frequently called the *edit distance*, is a general dynamic programming sequence distance metric typically used for string comparison. In that scenario, it refers to the number of edits necessary to convert one string into another, where an insertion, deletion, or conversion to another character has the same cost of one edit. Two strings a_n and b_m are compared character by character in a matrix. If two characters a_i and b_j are the same, the cost of that action is zero; if an edit is necessary, the cost of that particular change becomes one. The cost of converting the string characters up to and including a_i to b_j is calculated by taking the minimum cost of the

conversion before reaching one character before the current, and adding the cost of the last step, if any. This distance metric has been used in several HCI applications [6, 24]. However, it returns only values of 1 or 0, and it does not compare arbitrary vectors or tensors, only the number of edits as the difference between two strings. In the case of tensors, they would have to be identical for LV to be 0, which is incongruent with the nature of signals from sensors or their representations. Alternatively, one could define a threshold over their difference to output a 0 or a 1. However, we define the Euclidean Levenshtein Distance below, which aims to preserve additional information related to the difference between tensors.

7.2 Euclidean Levenshtein Distance

Our modified version of Levenshtein Distance (3) uses a similar concept to the original in that it performs a pairwise comparison of two sequences, consisting of key-point descriptors in a key-press. Within a key-press, the discovered key-points, whose order we preserve, are temporally related. We consider it reasonable to treat the list of descriptors on these key-points as a sequence because the voltage discharge during touch is a process with a relatively pre-defined trajectory. This makes these discovered key-points, and by association, their descriptors, a good fit to be matched to each-other, similar to characters in a string.

$$e\text{-}lev_{k_1,k_2}(i,j) = \begin{cases} \max(\|k_{1i}\|_2, \|k_{2j}\|_2) & \text{if } min(i,j) = 0, \\ \min \begin{cases} e\text{-}lev_{k_1,k_2}(i-1,j) + \|k_{1i}\|_2 \\ e\text{-}lev_{k_1,k_2}(i,j-1) + \|k_{2j}\|_2 \\ e\text{-}lev_{k_1,k_2}(i-1,j-1) + \|k_{1i} - k_{2j}\|_2 \end{cases} & \text{otherwise.} \end{cases} \tag{3}$$

Instead of using a binary 0 or 1 cost to denote a match or mismatch, as in the Levenshtein Distance with string characters, we compute the Euclidean distance between two key-point descriptors—each belonging to one of the key-presses in the pairwise comparison. The total cost of converting a key-press k_1 up to its descriptor k_{1i} to the other key-press k_2 up to its descriptor k_{2j} is computed by adding the distance between k_{1i} and k_{2j}, or the vanishing costs of k_{1i} or k_{2j} to the minimum of value of the previous step, according to equation 3. At every step of the algorithm, either one of the two key-points is kept and the other is discarded, or one is transformed to the other—whichever action has the minimum cost. This process is analogous to deletion, insertion, or character conversion when comparing two strings. In the minimum clause, from k_1 to k_2, the first case corresponds to deletion of the current key-point (k_{1i}), the second to insertion, and the third to conversion from k_{1i} to k_{2j}. If two key-presses are compared sample-by-sample, the resulting Euclidean Levenshtein Distance is a measure of similarity between them. A smaller distance indicates higher levels of similarity, since converting one key-press into the other would cost less. While this algorithm was specifically designed to measure the distance of two key-presses, each represented as a sequence of key-points, its uses are not limited to this particular application and can be extended to any problem that aims to measure the minimum cost of converting one tensor to another. For clarity, we illustrate a full example of computing the distance between two tensors in Appendix A.

The Euclidean Levenshtein Distance is a metric, as it complies with the conditions of 1) *positive-definiteness* 2) *symmetry* 3) *triangle inequality*. A full proof is beyond the scope of this work, however it relies on the following observations:

(1) All calculated distance values rely on the sum of L_2-norms, which are non-negative.
(2) The L_2-norm of the difference between two tensors is commutative.
(3) The only time that *ELD* can be zero, besides comparing vectors of length zero, is when the norm of the difference between the two tensors is zero, which in turn happens when the two tensors are the same.

(4) Every branch of calculation of *ELD* relies on L_2-norms, which comply with the triangle inequality condition.

8 USER STUDY

We conduct a study of 13 participants who briefly press discrete touch locations on a knitted capacitive sensor pad. Our experiments aim to test the validity of our capacitive sensing and signal representation methods. The contiguity of touch positions along the conductive yarn pathway poses challenges to accurately localizing discrete touch events. The added difficulty of this task is that time-series signals, especially those that are prone to noise, are difficult to represent in a meaningful way. While neural network-based approaches have emerged as a way to obviate feature construction in a classification task by aiming to automatically handle this step, there is still room to grow in the particular field of sensor time-series classification.

8.1 Experimental Questions

We aim to capture a signal from our capacitive sensor which is representative of touch location. Furthermore, we seek to reduce some of the complexity of the problem by retaining information from that signal which encodes meaningful underlying changes between touch positions and disregards similarities. This approach is not intended to substitute automatic feature extraction, but rather to enhance it as a pre-processing step to a more advanced neural network which can further learn to distinguish between different positions—but from a more informed starting point.

The focus of these experiments is to test whether:

(1) The data obtained from our *differential capacitive sensing* strategy provides statistically significant separability between locations of key-presses.
(2) Our proposed representation, produced by *MSD*, increases intra-class similarity and decreases inter-class similarity, compared to that of the raw signal data.

We use *ELD*, described in Section 7, as a distance metric in both cases, since this algorithm computes distances between tensor sequences of varying lengths.

8.2 Experimental Procedure

For our experiments, we use a 36-button touchpad with the same design as the one illustrated in Figure 3c. A group of 13 participants was selected to conduct data collection. The participants' ages ranged from 19 to 45 with an average age of 26. All participants were of good health and exhibited no skin-related medical conditions. Each of the 13 subjects conducted between 10 to 20 independent trials. A trial consists of one subject individually pressing and releasing each of 36 touch points over a duration of two seconds. The data was collected during one or more sittings per subject in a laboratory environment and processed as described in section 5. Additionally, a baseline (no-touch) measurement was also recorded during the beginning of each trial.

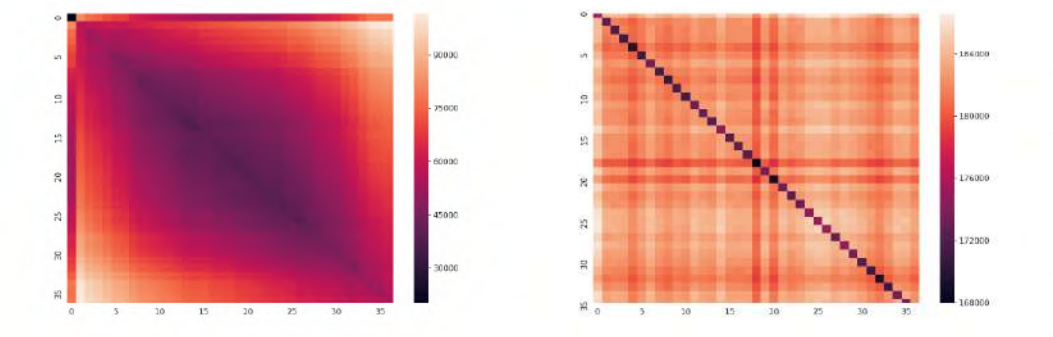
A press on these touch positions, or *buttons*, was similar in duration to a key-press on a keyboard, allowing enough time to capture the onset of press, pressure saturation, release and return to baseline. In order to investigate the inherent behavior of the sensor, the data was collected under ambient environmental conditions without adding further variability to it, such as altering humidity, body poses, etc. Particularly, data collected during trials was not subjected to interference from fluorescent light or other intense EMI sources. Some subjects did use different fingers when pressing touch positions, however that was not a controlled condition.

A total of 210 key-presses per button were collected from this study. Each key-press is expressed as a matrix of 250×2 gain samples, after being processed as described in Section 5. This matrix representation of key-press data contains "raw" signal data, serving as our baseline model. We use

it to investigate our experimental question *(1)*, posed above. In order to explore our experimental question *(2)*, we need to subsequently process the raw signal data using *MSD*, the algorithm introduced in Section 6 and illustrated in Figures 9 and 10. A processed data set of 7560 key-presses is produced, where each key-press contains a variable number of descriptors, depending on the number of key-points discovered in it. The dimensionality of the *MSD* descriptors is 16.

In order to measure the similarity relationship of different key-presses belonging to the same position, as compared to key-presses that belong to different positions, we compute the pairwise *Euclidean Levenshtein Distance* for every key-press pair in the data set, as described in Section 7.2. We generate a 36×36 matrix, where both rows and columns indicate the number of the touch location on the pad, which is the label of each key-press. Each key-press is associated with its label, and when two key-presses are compared to each other, a scalar value that represents their *ELD* is produced. That value is added to the matrix in position (m,n), where m and n denote the label of the first and second key-presses being compared, respectively.

9 RESULTS AND DISCUSSION



(a) Raw signal data. (b) *MSD* signal representation.

Fig. 11. Heatmaps indicating touch position distances, computed using the sum of *Euclidean Levenshtein Distance* of all key-presses' pairwise comparisons. Each heatmap has a size of 36×36, where each cell in either direction corresponds to a touch position. The key-press representations upon which these values were computed was obtained from the raw signal data in (a) and the *MSD* algorithm in (b). Each element of the heatmaps' diagonals composed of the sum of such distances of key-presses of the same class, while the rest of the elements result from sums of distances of key-presses of different classes.

We represent the key-press data set in two different ways: using the raw signal data and the *MSD* algorithm. Heatmaps, shown in Figure 11, were created over the two 36×36 matrices resulting from each of these methods to visualize the distances of key-press comparisons from all classes. The first heatmap, in Figure 11a is the result of the raw signal representation, while Figure 11b depicts the results of applying the *MSD* algorithm to the raw signal data to compute key-point descriptors. As can be seen from both heatmaps, the diagonal, which reflects distances of key-presses of the same class, is composed of lower-valued results compared to distances of key-presses from different classes. Intuitively, this demonstrates that both representations show invariances in their signal description. There is less similarity among different-class key-presses and more for key-presses belonging to the same class. The *MSD* representation visibly improves separability between classes.

9.1 Statistical Measures on Class Distances

For each method, we compute some statistical measures on the resulting distance matrix to gain more insight into what the data means. We investigate two groups:

(1) The distances between same-class key-presses, encoded in the heatmap diagonal.
(2) The distances between key-presses of different classes, encoded in the rest of the heatmap.

We record the *f-statistic* and its associated *p-value*, p_f, resulting from a *one-way ANOVA*, as well as the *z-score*, and its associated *p-value*, p_z. These inferential statistical tests measure separability of groups. The lower each *p-value* is, the more significant are the group differences considered. A $p\text{-}value \leq 0.05$ is generally accepted to mean that the difference between the groups in the data is statistically significant.

Table 1. Statistical significance of heatmap results.

	$f\text{-}stat$	p_f	$z\text{-}score$	p_z
Raw Data	58.26	0.00	-7.63	0.00
MSD Representation	2279.20	0.00	-47.74	0.00

Our first experimental question was: *Does the data obtained from our sensing strategy indicate some separability between location of key-presses?* Results in Table 1 show that both p_f and p_z are equal to 0.00, which means that there is an estimated 0% probability of the differences between our two groups having occurred by chance. Inspecting the heatmap in Figure 11a, we can see that the values along the diagonal are low compared to the rest of its values. These observations suggest that the construction of our sensing strategy is viable for touch location identification. However, we can see from Figure 11a that the raw signal representation, in addition to having low values for distances between same-class key-presses, also reports low values for other nearby locations. Such results seem reasonable when we consider the design of the sensor relying on one conductive yarn, where *buttons*, or defined key-press locations, are determined along its length. *Buttons* with label numbers that are numerically-close, are located one after the other along the yarn, explaining the low distances among those key-presses as seen in the heatmap. These findings are also compatible with those from [30], where general areas of touch were able to be localized, but not precise locations of key-presses.

In order to determine greater precision in touch location identification, it seems more complex and refined signal representation strategies are necessary. We look at our second experimental question: *Does our proposed representation, produced by MSD, increase intra-class similarity and decrease inter-class similarity, compared to the raw signal data?* Results in Table 1, again, show that both p_f and p_z are equal to 0.00, indicating high statistical significance of differences between same-class key-presses and key-presses of different classes. Furthermore, the reported $f\text{-}stat$ and $z\text{-}score$ values for the *MSD* signal representation are considerably larger than those measures for the raw signal data representation, suggesting greater group differences. The corresponding heatmap in Figure 11b as well, shows how distances between same-class key-presses are much lower than differences between key-presses of different classes. Compared to the heatmap in Figure 11a, the differences between classes that are numerically-close have also been significantly reduced. It can be inferred from these results that the *MSD* representation of the signal does indeed increase the potential for fine location identification. While classification of *button* labels, or key-press locations, is beyond the scope of this study, these results strongly suggest that it is possible to implement touch location classification and construct interactive systems that rely on it.

9.2 Resources and Performance

The computation whose speed we will be discussing here is that of the heatmap matrix using the data of only one subject, a subset of the whole dataset. It relies on computing the *ELD* of every pair of 180 key-presses, resulting into 32220 total comparisons. The complexity of the *ELD* algorithm, as currently implemented is quadratic. Computing the distance matrix is not a task that would necessarily need to be repeated for key-press classification, or any application that relies on it. However, since this algorithm has relatively high complexity, it can serve as an indicator of the time each representation of the key-presses might need to be processed depending on the application.

The heatmap matrix for each representation, implemented in Python, was computed on an Ubuntu 18.04 machine with 128 GB of RAM, and an Intel® Xeon® CPU E5-2650 v2 @ 2.60GHz. The CPU has 32 cores, with 2 threads/core, and each heatmap was computed in parallel using 64 processes. The time necessary to compute the heatmap in Figure 11a, where key-presses were represented by raw signal data, was approximately 17 hours, while the time to compute the heatmap in Figure 11b, where key-press representation were produced using the *MSD* algorithm, was approximately 20 minutes. The considerable improvement in processing time is a direct result of the sparse representation of the signal through *MSD*, which is another benefit of using it, in addition to better class separability.

10 FROM TEXTILE DESIGN TO APPLICATION INTEGRATION

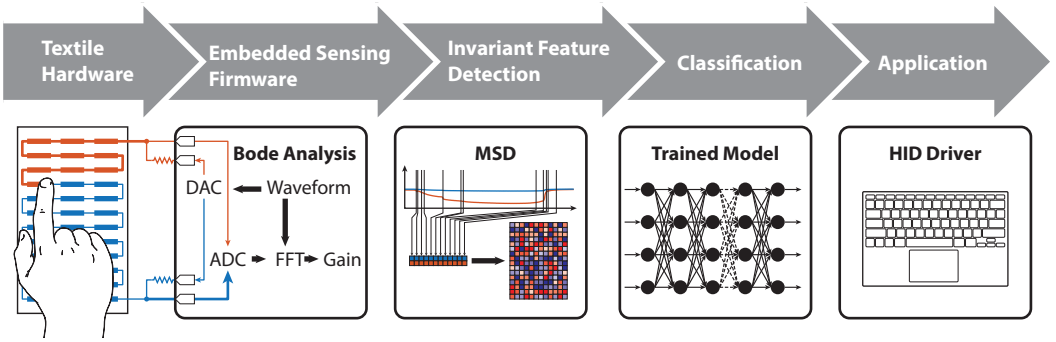


Fig. 12. Integration of Bode analysis and invariant feature detection into the application development pipeline. Textile hardware is connected to an embedded sensing microprocessor that performs signal generation, acquisition and processing. The controller passes the data to the MSD algorithm which extracts key invariant features. The features are passed to a trained classifier which outputs a position or gesture used by an application or device driver.

The process to create a knitted interface accommodates the workflow of both textile designers and UX programmers. Digital weft knitting workstations like the *Shima Seiki SDS APEX ONE* contain software to program and visualize the structure of the knitted textile and create assembly programs for use with digitally-controlled knitting machines. The serpentine route of the conductive yarn is fully compatible with the weft knitting process and is intuitive for textile designers to understand and adapt into designs. The yarn can be exposed on the surface of a textile by knitting loops on the front or back needle beds, forming visible touch locations for interface elements like buttons or sliders. Alternatively, the yarn can be hidden within the textile by "floating" un-knitted yarn between needle beds through the middle of the textile. At the beginning and end of the knitted path, the yarn endpoints are exposed and affixed with fabric-to-wire connectors.

An advantage of requiring only two connections to any textile sensor allows for the use of a uniform sensing controller across multiple textile designs. The sensing controller utilizes a low-power microprocessor for signal generation, acquisition and processing. The current sourced across the fabric circuit should not exceed 10 μA. The power consumption of the microprocessor depends on computations performed or the power requirements of peripherals like wireless transceivers but should not exceed 300 mW. Algorithms like *MSD* can be embedded into the on-board processor for use with internal or external heuristics or classifiers.

In a complete system, the sensing controller would return instantaneous estimated touch location and pressure and may also output information relating to high-level events like tapping or swiping. A mapping of user input to actions may be embedded within the controller or the unprocessed signal may be sent downstream to be interpreted by remote applications. Other algorithmic implementations can be used instead of, or in conjunction with, the ones presented here. Application development can rely on input received from the sensing controller through a wired or wireless connection. The sensing controller can connect to a computer or smart device and stream input as a Human Interface Device like a keyboard or mouse and directly receive touch or key input.

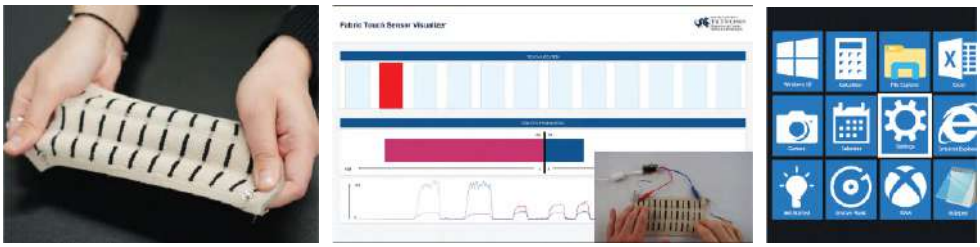
Figure 12 shows the main stages of the process, starting with the knitted sensor, which is connected to a microprocessor. The excitation signal is generated by the microprocessor at a rate at or above 1 MHz. The signal is processed using the ARM CMSIS DSP complex FFT [13]. Subsequently, the signal is represented according to the *MSD* algorithm. The implementation of *MSD* is currently in Python, and relies on NumPy [31] and SciPy [32]. In order to return location of touch, supervised learning would be needed. A trained model could be deployed and the input data, after signal processing and feature extraction, would be assigned a location. This information would then be used by a higher level application. In the section below, we describe some application prototypes using capacitive knitted sensors.

11 APPLICATIONS OF TOUCH-SENSITIVE KNITTED FABRIC

The physical design possibilities of the knitted capacitive touch sensor are diverse, given the extensibility of knitting as a manufacturing process. Sensitive knitted interfaces could be produced as standalone devices or integrated into existing products. We developed textile analogues of human-computer interfaces like trackpads or keyboards to appeal to users' familiarity with such devices and allow comparisons to be formed from an inherent understanding. Applications like roll-out keyboards or flexible trackpads showcase devices that are more comfortable to touch than hard electronics and are more robust to bending than flexible electronics. Below, we detail several application prototypes we have created to illustrate use cases of knitted capacitive sensors. These prototypes demonstrate functionality with 8-bit microprocessors that do not perform Bode analysis or *MSD* but instead are based on the hardware and methods proposed in [30]. Even though the localization accuracy is not very high in these prototypes, they illustrate the usability potential of this technology.

11.1 Knitted Touchpad

Figure 13 demonstrates a standalone knitted touch sensor, similar in design to the knitted device we used for our experiments. The button-like touch points are formed by exposing yarn on the fabric surface during knitting and are spaced approximately 1/2 inches apart. The touch points are knit along a serpentine path. Two alligator clips connect the yarn endpoints to the sensing controller. We use this as a general form of the sensor, since it resembles typical trackpads and keypads. This design can be scaled by increasing the length or width of the pattern within the knit program. Additional conductive yarns can be knit together to match the resistance of larger



(a) (b) (c)

Fig. 13. The weft-knitted touchpad: *(a)* Stretching the sensor. *(b)* Signal visualization upon touch. *(c)* A mock-up button interface.

and smaller pads. Figure 13a shows how the knitted sensor can be stretched without damage to its structural integrity.

In Figure 13b, we connect the sensor to a visualization application, which shows the approximate location of touch, as a function of the voltage differential from both connections. A demonstration of this application can be found in the associated video of this work. As mentioned above, sensing hardware and processing methods from previous work [30] are used for demonstration purposes. In practice, the sensor measures the difference between columns more accurately than within the same column. This is due to the larger resistance difference between columns than between buttons in the same column. The bars on the left and the right indicate the voltage gain detected from each yarn endpoint connection, and they change with location of touch along the conductive yarn in the sensor. The row of rectangles above shows the detected column of touch along the sensor. As we can see, the pink horizontal bar on the left side is longer than the blue bar for that key-press, and the key-press is estimated to come from the second column—as shown from the vertical red bar on top.

The UI mock-up shown in Figure 13c illustrates a potential use of the sensor as a button interface. The user would press a touchpad button to select the corresponding functionality.

11.2 Upholstery-Integrated Touchpad

(a) (b) (c)

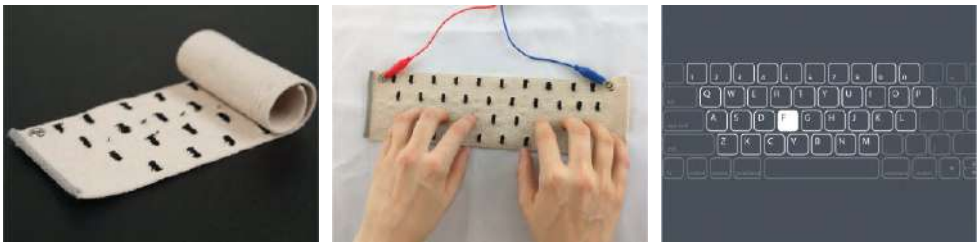
Fig. 14. Upholstery-Integrated Touchpad: *(a)* Knitted touchpad integrated into a slip cover *(b)* Using the integrated knitted sensing system *(c)* A mock-up media player prototype

In this use case, we demonstrate how the knitted touchpad can be integrated into other textiles. Figure 14a shows the sensor sewn into a slip-cover, with wiring hidden inside the upholstery. A

user seated in the chair (Figure 14b) could easily access the sensing pad and use it as a control system. This same knitted structure can be integrated into furniture, car seats, pillows, blankets and other textile-based surfaces.

Its functionality can vary depending on user needs. Figure 14c shows the touchpad's possible use as a media controller. The touchpad is similar to that of 13a, however, rather than relying on fine touch locations, the application interface divides input into three regions: left, center and right. Each region recognizes tap and hold gestures. This is an example of an application which relies on resolving areas of touch that are relatively broadly defined. Compared to the fine touch position identification explored in this work, it is easier to accurately detect the region of touch in this case. The level of accuracy necessary in resolving location of touch is application-dependent.

11.3 Knitted Keyboard



(a) (b) (c)

Fig. 15. Knitted computer keyboard: *(a)* Unrolling the sensor *(b)* Knitted yarn structure *(c)* A keyboard UI

Knitted sensors can be constructed with the conductive yarn structured in different patterns, according to the purpose of each application. The carbon fiber yarn of the sensor in Figure 15 is knitted to form an alphanumeric computer keyboard. The touch points are knitted in the same serpentine structure as those in the touchpad, though the placement is staggered to mimic a standard keyboard. The buttons are again spaced approximately 1/2 inches apart to maintain the spacing of a full-sized computer keyboard.

Given the apparent uses for a computer keyboard, a knitted keyboard could serve as a useful interface for mobile computing, as illustrated in Figures 15b and 15c. A textile keyboard can be made lighter, thinner and more durable than a conventional keyboard. The keyboard could be integrated into a laptop sleeve or tablet cover to be used as a wireless keyboard when the device is in use. This device could function similarly to low-profile keyboards like the Apple *Smart Keyboard Folio* for iPad or Microsoft *Surface Type Cover* which attach to and fold over their respective tablet. Unlike current flexible electronics, a fully-knitted keyboard could stretch, and fold without damaging flexible connections. Figure 15a shows the knitted keyboard unrolling.

11.4 Sensing Sleeve

This example of the knitted touch sensor demonstrates its potential towards wearable interfaces for use with smart phones, tablets or computers. The buttons integrated into the sleeve could act as "quick action" functions to make calls, play music, report or save the wearer's location and report the date and time, as illustrated in Figure 16c. The buttons can be used as a "hands-free" method of interacting with a smart device. Additionally, the textile may be used in tandem with voice commands to control functionality—for instance, pressing "Phone" and speaking the name of the recipient.

(a) (b) (c)

Fig. 16. Knitted sensing sleeve: *(a)* Hidden hardware pocket *(b)* Seamless integration for high usability *(c)* A mock-up button interface

The sleeve is created with the conductive yarn fully-integrated, without needing to sew knitted components separately. There is a pocket knitted on the inside of the sleeve, shown in Figure 16a, which can host any low-profile electronics necessary for interactivity. The sensing hardware attaches to the sleeve by fabric snaps and can easily detach to allow the shirt to be laundered. The sleeve maintains flexibility and can fold and roll without damage to the fabric or wiring. The polyester can be color-matched to the conductive yarn to create a seamless appearance. As depicted in Figure 16b, this easy-to-use and lightweight design offers opportunities for increased functionality of clothes in a variety of circumstances.

11.5 Further Use Cases

We envision many more possible applications of the sensor both as a touch interface, and for human activity recognition. Some uses are mentioned below, even though this list is not comprehensive of this technology's potential.

(1) Automotive upholstery could be produced with sensitive yarns that measure input within the cabin, like adjusting the seat position, opening and closing windows or controlling temperature as would conventional controls.
(2) A distributed touch sensor could detect the physical state of the driver in the event of loss of consciousness, or to alert the driver of the movement of unrestrained cargo.
(3) Smart homes and workplaces could be outfitted with touch-sensitive carpeting on floors or walls to monitor motion to adjust lighting or environmental controls automatically as occupants move through the space; or use touch on a wall to directly control lighting or room access.
(4) Such sensors could also detect intruders outside of the range of cameras or distinguish the movement of pets from those of people.
(5) Clothing could contain sensitive yarns that detect input and relay commands to smartphones or media devices.
(6) Clothing integration could also act as a sensitive skin for prosthetic limbs or a sensitive outer surface for robotic arms.

12 LIMITATIONS AND FUTURE WORK

The prospect of human-centered designs using a knitted capacitive touch sensor is fascinating, however there are still some technical challenges to overcome. The durability of the knitted textile is an open question and requires extended testing to evaluate the electrical properties of the fabric due to abrasion, heating and material aging. Methods of preservation, like lamination, should

also be evaluated as this may affect touch detection performance. The effects of electromagnetic interference (EMI) on touch localization accuracy and resolution must also be studied further. EMI sources like fluorescent lighting and power supplies may be in close proximity to the touch sensor and induce noise in measurements. While Bode analysis can be used to reject interference at specific frequencies, more severe white noise may pose a nontrivial issue.

Another factor worth examining is the ability of the sensor to localize soft touch. The accuracy of the sensor depends on the strength of the applied capacitance. In the case of applying a weak capacitance, such as during a soft touch, it is difficult for the sensor to accurately localize the input. This limitation should be examined further when evaluating user actions like swipes, which do not exert heavy pressure on the sensor. There is also presently a limitation on the types of input that can be measured. The sensor can only detect contact from conductive objects. Furthermore, the sensor cannot presently measure more than one touch point simultaneously. Multiple touch locations are averaged to resolve a single location. Future work will involve disambiguation of the number of touch points contacting the sensor.

Larger scale user studies are also needed to ensure proper response under varying conditions such as finger placements, body poses, weather conditions, and more. Moreover, qualitative user studies should also be conducted to understand how much if at all, and in what way would users engage with this technology. We also plan to address both these aspects in our future work.

In this work, we detailed the engineering of a system which can produce knitted capacitive sensors and demonstrates the viability of class separation, in order to enable textile interactive systems. The larger goal is to build a a classification system, based on our proposed system, and train it under such noise-inducing environments to increase its robustness to outliers produced by them. We find the prospect of enabling many of the above-mentioned applications, exciting, especially since the technical ground was demonstrated to be mature and robust enough to support their implementation.

13 CONCLUSION

In this work, we advance the technical aspects of knitted capacitive touch sensing. The sensor design aims to be human-centered and usable, relying on low-profile electrical components. There are only two wires coming out of the knitted sensor, to make the experience as close to that of touching or wearing regular fabric as possible. Furthermore, manufacturing was a main consideration during the sensor design process, offering scalability and easy producibility.

The differential capacitive sensing method used with Bode analysis increases separability of touch location along a conductive yarn and reduces noise. Our feature representation algorithm further moves toward accurate sensing by capturing invariant aspects of the signal behaviour that indicate position of touch. Experiments show that the distances between key-presses of the same position are lower than distances between key-presses of different positions. Moreover, the signal representation produced by the *MSD* algorithm outperforms the baseline raw signal data. Supervised learning with respect to positions of touch along the sensing yarn is the next step toward building a variety of accurate interactive touch applications, including ones based on prototypes introduced in this work.

ACKNOWLEDGMENTS

We thank Amy Stoltzfus and Keith Taylor from Drexel University's Pennsylvania Fabric Discovery Center at the Center for Functional Fabrics and Robert Lehrich from SHIMA SEIKI USA, for their invaluable digital knitting expertise. This research is supported by the Pennsylvania Fabric Discovery Center, the Drexel University ExCITe Center and the Advanced Functional Fabrics of America grant W15QKN-16-3-0001.

REFERENCES

[1] Talha Agcayazi, Michael McKnight, Hannah Kausche, Tushar Ghosh, and Alper Bozkurt. 2016. A finger touch force detection method for textile based capacitive tactile sensor arrays. In *2016 IEEE SENSORS*. 1–3. https://doi.org/10.1109/ICSENS.2016.7808528

[2] Jean Babaud, Andrew P. Witkin, Michel Baudin, and Richard O. Duda. 1986. Uniqueness of the Gaussian kernel for scale-space filtering. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 1 (1986), 26–33.

[3] Adeline Bailly, Simon Malinowski, Romain Tavenard, Laetitia Chapel, and Thomas Guyet. 2015. Dense bag-of-temporal-sift-words for time series classification. In *International Workshop on Advanced Analysis and Learning on Temporal Data*. Springer, 17–30.

[4] Gary Barrett and Ryomei Omote. 2010. Projected-Capacitive Touch Technology. *Information Display* 26 (03 2010), 16–21. https://doi.org/10.1002/j.2637-496x.2010.tb00229.x

[5] Hendrik Wade Bode. 1940. Relations between attenuation and phase in feedback amplifier design. *The Bell System Technical Journal* 19, 3 (July 1940), 421–454. https://doi.org/10.1002/j.1538-7305.1940.tb00839.x

[6] Adrien Coyette, Sascha Schimke, Jean Vanderdonckt, and Claus Vielhauer. 2007. Trainable sketch recognizer for graphical user interface design. In *IFIP Conference on Human-Computer Interaction*. Springer, 124–135.

[7] Scott Gilliland, Nicholas Komor, Thad Starner, and Clint Zeagler. 2010. The Textile Interface Swatchbook: Creating graphical user interface-like widgets with conductive embroidery. In *Proceedings of the 2010 International Symposium on Wearable Computers (ISWC '10')*. 1–8. https://doi.org/10.1109/ISWC.2010.5665876

[8] Nur Al-huda Hamdan, Jeffrey R. Blum, Florian Heller, Ravi Kanth Kosuru, and Jan Borchers. 2016. Grabbing at an Angle: Menu Selection for Fabric Interfaces. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers (ISWC '16)*. ACM, New York, NY, USA, 1–7. https://doi.org/10.1145/2971763.2971786

[9] Nur Al-huda Hamdan, Simon Voelker, and Jan Borchers. 2018. Sketch&Stitch: Interactive Embroidery for E-textiles. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 82, 13 pages. https://doi.org/10.1145/3173574.3173656

[10] Yuji Hasegawa, M. Shikida, D. Ogura, and Kazuo Sato. 2007. Novel type of fabric tactile sensor made from artificial hollow fiber. In *2007 IEEE 20th International Conference on Micro Electro Mechanical Systems (MEMS)*. 603–606. https://doi.org/10.1109/MEMSYS.2007.4433079

[11] Dana Hughes, Halley Profita, Sarah Radzihovsky, and Nikolaus Correll. 2017. Intelligent RF-Based Gesture Input Devices Implemented Using e-Textiles. (2017). https://doi.org/10.3390/s17020219

[12] Jarden Applied Materials 2014. *RESISTAT® TYPE F901, MERGE D044*. Jarden Applied Materials. http://www.resistat.com/pdf/f901d044color.pdf

[13] Keil. [n.d.]. ARM CMSIS-DSP. https://arm-software.github.io/CMSIS_5/DSP/html/modules.html.

[14] Joseph B. Kruskall. 1983. The symmetric time warping algorithm: From continuous to discrete. *Time warps, string edits and macromolecules* (1983).

[15] Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.

[16] Tony Lindeberg. 1990. Scale-space for discrete signals. *IEEE transactions on pattern analysis and machine intelligence* 12, 3 (1990), 234–254.

[17] Tony Lindeberg. 1994. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of applied statistics* 21, 1-2 (1994), 225–270.

[18] Tony Lindeberg. 2013. *Scale-space theory in computer vision*. Vol. 256. Springer Science & Business Media.

[19] David G. Lowe. 2004. Distinctive image features from scale-invariant keypoints. *International journal of computer vision* 60, 2 (2004), 91–110.

[20] Jifei Ou, Daniel Oran, Don D. Haddad, Joseph Paradiso, and Hiroshi Ishii. 2019. SensorKnit: Architecting Textile Sensors with Machine Knitting. *3D Printing and Additive Manufacturing* 6 (March 2019), 1–11. https://doi.org/10.1089/3dp.2018.0122

[21] Patrick Parzer, Adwait Sharma, Anita Vogl, Jürgen Steimle, Alex Olwal, and Michael Haller. 2017. SmartSleeve: Real-time Sensing of Surface and Deformation Gestures on Flexible, Interactive Textiles, Using a Hybrid Gesture Detection Pipeline. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology (UIST '17)*. ACM, New York, NY, USA, 565–577. https://doi.org/10.1145/3126594.3126652

[22] E. Rehmi Post, Maggie Orth, Peter R. Russo, and Neil Gershenfeld. 2000. E-broidery: Design and Fabrication of Textile-based Computing. *IBM Syst. J.* 39, 3-4 (July 2000), 840–860. https://doi.org/10.1147/sj.393.0840

[23] Ivan Poupyrev, Nan-Wei Gong, Shiho Fukuhara, Mustafa Emre Karagozler, Carsten Schwesig, and Karen E. Robinson. 2016. Project Jacquard: Interactive Digital Textiles at Scale. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16)*. ACM, New York, NY, USA, 4216–4227. https://doi.org/10.1145/2858036.2858176

[24] Kari-Jouko Räihä. 2010. Some applications of string algorithms in human-computer interaction. In *Algorithms and applications*. Springer, 196–209.

[25] Stan Salvador and Philip Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis* 11, 5 (2007), 561–580.
[26] Stefan Schneegass and Alexandra Voit. 2016. GestureSleeve: Using Touch Sensitive Fabrics for Gestural Input on the Forearm for Controlling Smartwatches. In *Proceedings of the 2016 ACM International Symposium on Wearable Computers (ISWC '16)*. ACM, New York, NY, USA, 108–115. https://doi.org/10.1145/2971763.2971797
[27] Maximilian Sergio, Nicolò Manaresi, Marco Tartagni, Roberto Guerrieri, and Roberto Canegallo. 2002. A textile based capacitive pressure sensor. In *SENSORS, 2002 IEEE*, Vol. 2. 1625–1630. https://doi.org/10.1109/ICSENS.2002.1037367
[28] Joshua Smith, Tom White, Christopher Dodge, Joseph Paradiso, Neil Gershenfeld, and David Allport. 1998. Electric Field Sensing For Graphical Interfaces. *IEEE Comput. Graph. Appl.* 18, 3 (May 1998), 54–60. https://doi.org/10.1109/38.674972
[29] Seiichi Takamatsu, Takeshi Kobayashi, Nobuhisa Shibayama, Koji Miyake, and Toshihiro Itoh. 2011. Meter-scale surface capacitive type of touch sensors fabricated by weaving conductive-polymer-coated fibers. In *2011 Symposium on Design, Test, Integration Packaging of MEMS/MOEMS (DTIP)*. 142–147.
[30] Richard Vallett, Ryan Young, Chelsea Knittel, Youngmoo Kim, and Geneviève Dion. 2016. Development of a Carbon Fiber Knitted Capacitive Touch Sensor. *MRS Advances* 1, 38 (2016), 2641–2651. https://doi.org/10.1557/adv.2016.498
[31] Stefan Van Der Walt, S. Chris Colbert, and Gael Varoquaux. 2011. The NumPy array: a structure for efficient numerical computation. *Computing in Science & Engineering* 13, 2 (2011), 22.
[32] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2019. SciPy 1.0–Fundamental Algorithms for Scientific Computing in Python. *arXiv preprint arXiv:1907.10121* (2019).
[33] Raphael Wimmer and Patrick Baudisch. 2011. Modular and Deformable Touch-sensitive Surfaces Based on Time Domain Reflectometry. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology (UIST '11)*. ACM, New York, NY, USA, 517–526. https://doi.org/10.1145/2047196.2047264
[34] Hui Zhang, Xiaoming Tao, Shanyuan Wang, and Tongxi Yu. 2005. Electro-Mechanical Properties of Knitted Fabric Made From Conductive Multi-Filament Yarn Under Unidirectional Extension. *Textile Research Journal* 75, 8 (August 2005), 598–606. https://doi.org/doi:10.1177/0040517505056870
[35] Yang Zhang, Gierad Laput, and Chris Harrison. 2017. Electrick: Low-Cost Touch Sensing Using Electric Field Tomography. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/3025453.3025842

A EUCLIDEAN LEVENSHTEIN DISTANCE EXAMPLE

Let us assume we have two tensors, **A** and **B**, represented as matrices, and we want to calculate their *ELD*. We can consider each as sequences of their rows, where each row has the same number of elements—in this case 4. We will refer to each element of these sequences as $\mathbf{a}_x$ and $\mathbf{b}_z$.

$$\mathbf{A}_{x,y} = \begin{pmatrix} 1 & 3 & 0 & 2 \\ 4 & 11 & 2 & 3 \\ 7 & 1 & 10 & 0 \\ 5 & 2 & 6 & 8 \end{pmatrix} \qquad \mathbf{B}_{z,w} = \begin{pmatrix} 2 & 1 & 5 & 12 \\ 3 & 4 & 9 & 1 \\ 19 & 7 & 2 & 6 \end{pmatrix}$$

We then define matrix **M**, which is originally an all zero matrix, and will progressively be populated based on the values of the sequences'elements according to equation 3. The size of this matrix depends on the number of elements of the two sequences being compared, and in this example is *4 x 3*.

$$\mathbf{M}_{i,j} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

In order to compute **M**, we need to rely on the L_2 norms of $\mathbf{a}_x$ and $\mathbf{b}_z$ and in certain instances, the norm of these two vectors'difference. Starting step-by-step:

When $i = 0, j = 0$:

$$\mathbf{M}_{0,0} = max(\|\mathbf{a}_0\|, \|\mathbf{b}_0\|) = max(3.74, 13.19) = 13.19$$

When $i = 0, j = 1$:

$$\mathbf{M}_{0,1} = max(\|\mathbf{a}_0\|, \|\mathbf{b}_1\|) = max(3.74, 10.34) = 10.34$$

When $i = 0, j = 2$:

$$\mathbf{M}_{0,2} = max(\|\mathbf{a}_0\|, \|\mathbf{b}_2\|) = max(3.74, 13.75) = 13.75$$

At this point, the state of $\mathbf{M}$ is the following:

$$\mathbf{M}_{i,j} = \begin{pmatrix} 13.19 & 10.34 & 13.75 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

When $i = 1, j = 0$:

$$\mathbf{M}_{1,0} = max(\|\mathbf{a}_1\|, \|\mathbf{b}_0\|) = max(12.25, 13.19) = 13.19$$

When $i = 1, j = 1$:

$$up = \mathbf{M}_{i-1,j} + \|\mathbf{a}_i\| = \mathbf{M}_{0,1} + \|\mathbf{a}_1\| = 10.34 + 12.25 = 22.59$$
$$left = \mathbf{M}_{i,j-1} + \|\mathbf{b}_j\| = \mathbf{M}_{1,0} + \|\mathbf{b}_1\| = 13.19 + 10.34 = 23.53$$
$$diag = \mathbf{M}_{i-1,j-1} + \|\mathbf{a}_i - \mathbf{b}_j\| = \mathbf{M}_{0,0} + \|\mathbf{a}_1 - \mathbf{b}_1\| = 13.19 + 10.15 = 23.34$$
$$\mathbf{M}_{1,1} = min(left, up, diag) = 22.59$$

When $i = 1, j = 2$:

$$up = \mathbf{M}_{i-1,j} + \|\mathbf{a}_i\| = \mathbf{M}_{0,2} + \|\mathbf{a}_1\| = 13.75 + 12.25 = 26.00$$
$$left = \mathbf{M}_{i,j-1} + \|\mathbf{b}_j\| = \mathbf{M}_{1,1} + \|\mathbf{b}_2\| = 22.59 + 13.75 = 36.34$$
$$diag = \mathbf{M}_{i-1,j-1} + \|\mathbf{a}_i - \mathbf{b}_j\| = \mathbf{M}_{0,1} + \|\mathbf{a}_1 - \mathbf{b}_2\| = 10.34 + 7.81 = 18.15$$
$$\mathbf{M}_{1,2} = min(left, up, diag) = 18.15$$

At this point, the state of $\mathbf{M}$ is the following:

$$\mathbf{M}_{i,j} = \begin{pmatrix} 13.19 & 10.34 & 13.75 \\ 13.19 & 22.59 & 18.15 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

When $i = 2, j = 0$:

$$\mathbf{M}_{2,0} = max(\|\mathbf{a}_2\|, \|\mathbf{b}_0\|) = max(12.25, 13.19) = 13.19$$

When $i = 2, j = 1$:

$$up = \mathbf{M}_{i-1,j} + \|\mathbf{a}_i\| = \mathbf{M}_{1,1} + \|\mathbf{a}_2\| = 22.59 + 12.25 = 34.84$$
$$left = \mathbf{M}_{i,j-1} + \|\mathbf{b}_j\| = \mathbf{M}_{2,0} + \|\mathbf{b}_1\| = 13.19 + 10.34 = 23.53$$
$$diag = \mathbf{M}_{i-1,j-1} + \|\mathbf{a}_i - \mathbf{b}_j\| = \mathbf{M}_{1,0} + \|\mathbf{a}_2 - \mathbf{b}_1\| = 13.19 + 5.20 = 18.39$$
$$\mathbf{M}_{2,1} = min(left, up, diag) = 18.39$$

When $i = 2, j = 2$:

$$up = \mathbf{M}_{i-1,j} + \|\mathbf{a}_i\| = \mathbf{M}_{1,2} + \|\mathbf{a}_2\| = 18.15 + 12.25 = 30.40$$
$$left = \mathbf{M}_{i,j-1} + \|\mathbf{b}_j\| = \mathbf{M}_{2,1} + \|\mathbf{b}_2\| = 18.39 + 13.75 = 32.14$$
$$diag = \mathbf{M}_{i-1,j-1} + \|\mathbf{a}_i - \mathbf{b}_j\| = \mathbf{M}_{1,1} + \|\mathbf{a}_2 - \mathbf{b}_2\| = 22.59 + 11.18 = 33.77$$
$$\mathbf{M}_{2,2} = min(left, up, diag) = 30.40$$

At this point, the state of $\mathbf{M}$ is the following:

$$\mathbf{M}_{i,j} = \begin{pmatrix} 13.19 & 10.34 & 13.75 \\ 13.19 & 22.59 & 18.15 \\ 13.19 & 18.39 & 30.40 \\ 0 & 0 & 0 \end{pmatrix}$$

When $i = 3, j = 0$:

$$\mathbf{M}_{3,0} = max(\|\mathbf{a}_3\|, \|\mathbf{b}_0\|) = max(11.36, 13.19) = 13.19$$

When $i = 3, j = 1$:

$$up = \mathbf{M}_{i-1,j} + \|\mathbf{a}_i\| = \mathbf{M}_{2,1} + \|\mathbf{a}_3\| = 18.39 + 11.36 = 29.75$$
$$left = \mathbf{M}_{i,j-1} + \|\mathbf{b}_j\| = \mathbf{M}_{3,0} + \|\mathbf{b}_1\| = 13.19 + 10.34 = 23.53$$
$$diag = \mathbf{M}_{i-1,j-1} + \|\mathbf{a}_i - \mathbf{b}_j\| = \mathbf{M}_{2,0} + \|\mathbf{a}_3 - \mathbf{b}_1\| = 13.19 + 8.12 = 21.31$$
$$\mathbf{M}_{3,1} = min(left, up, diag) = 21.31$$

When $i = 3, j = 2$:

$$up = \mathbf{M}_{i-1,j} + \|\mathbf{a}_i\| = \mathbf{M}_{2,2} + \|\mathbf{a}_3\| = 30.40 + 11.36 = 41.76$$
$$left = \mathbf{M}_{i,j-1} + \|\mathbf{b}_j\| = \mathbf{M}_{3,1} + \|\mathbf{b}_2\| = 21.31 + 13.75 = 35.06$$
$$diag = \mathbf{M}_{i-1,j-1} + \|\mathbf{a}_i - \mathbf{b}_j\| = \mathbf{M}_{2,1} + \|\mathbf{a}_3 - \mathbf{b}_2\| = 18.39 + 8.37 = 26.76$$
$$\mathbf{M}_{3,2} = min(left, up, diag) = 26.76$$

Finally, the state of $\mathbf{M}$ is:

$$\mathbf{M}_{i,j} = \begin{pmatrix} 13.19 & 10.34 & 13.75 \\ 13.19 & 22.59 & 18.15 \\ 13.19 & 18.39 & 30.40 \\ 13.19 & 21.31 & 26.76 \end{pmatrix}$$

As a result, $ELD(\mathbf{A}, \mathbf{B}) = 26.76$.

Received July 2019; revised September 2019; revised November 2019; accepted December 2019