SOFT ROBOTICS
Volume 00, Number 00, 2019
[a] Mary Ann Liebert, Inc.
DOI: 10.1089/soro.2018.0086

SoRo
Soft Robotics

A Deformable Interface for Human Touch Recognition Using Stretchable Carbon Nanotube Dielectric Elastomer Sensors and Deep Neural Networks

Chris Larson,[1] Josef Spjut,[2] Ross Knepper,[3] and Robert Shepherd[1,4]

Abstract

This article presents a machine learning approach to map outputs from an embedded array of sensors distributed throughout a deformable body to continuous and discrete virtual states, and its application to interpret human touch in soft interfaces. We integrate stretchable capacitors into a rubber membrane, and use a passive addressing scheme to probe sensor arrays in real time. To process the signals from this array, we feed capacitor measurements into convolutional neural networks that classify and localize touch events on the interface. We implement this concept with a device called OrbTouch. To modularize the system, we use a supervised learning approach wherein a user deÞnes a set of touch inputs and trains the interface by giving it examples; we demonstrate this by using OrbTouch to play the popular game Tetris. Our regression model localizes touches with mean test error of 0.09 mm, whereas our classiÞer recognizes Þve gestures with a mean test error of 1.2%. In a separate demonstration, we show that OrbTouch can discriminate between 10 different users with a mean test error of 2.4%. At test time, we feed the outputs of these models into a debouncing algorithm to provide a nearly error-free experience.

Keywords: soft robotics, neural networks, humanÐcomputer interaction, stretchable electronics

Introduction

Humans and other animals demonstrate a remarkable ability to map sensory information from their skin onto internal notions of hardness, texture, and temperature in order to reason about their physical environment. This capability is enabled by massively parallelized neural computation within the somatosensory cortex, which is fed by a network of nerve cells distributed throughout the epidermis. Recent advances in stretchable electronics, soft robotics, and nonconvex optimization methods for deep neural networks now offer us building blocks on which we can start to replicate this tactile perception synthetically. Inspired by biological skins, in this study we have leveraged these advances to develop OrbTouch, a device that interprets tactile inputs using deep neural networks trained on examples provided by a user.

Figure 1 illustrates the OrbTouch concept. We monolithically integrate stretchable carbon nanotube (CNT) capacitors into its rubber membrane to create a soft haptic interface. The sensing apparatus is composed of an overlapping mesh of CNT Þlms, in which orthogonal traces are separated by a thin layer of rubber, forming a parallel plate capacitor at each intersection. Our sensing matrix is designed to enable the independent addressing of n^2 sensors using n electrical connections. To localize interactions on the interface, we feed a single sensor output vector (i.e., from one time step) into a two-dimensional (2D) convolutional neural network (CNN) that regresses the coordinates of touch events. To classify these events, which may vary in abstraction from a simple poke (Fig. 1a) to gestures producing complex deformations that evolve over time (Fig. 1b, c), we convolve a three-dimensional (3D) Þlter over several time steps of the incoming data stream to capture the relevant spatiotemporal features. As simple demonstrations of this idea, we use OrbTouch to play the video game Tetris, in real time at a sampling rate of 10 Hz, and also to identify users.

The remainder of this article is organized as follows: in Section 2 we brießy discuss recent advances in shape-

[1]Department of Mechanical Engineering, Cornell University, Ithaca, New York.
[2]NVIDIA Research, NVIDIA Corporation, Santa Clara, California.
Departments of [3]Computer Science and [4]Materials Science and Engineering, Cornell University, Ithaca, New York.

FIG. 1. Illustration of the OrbTouch concept. A dome-shaped balloon is inßated to render a haptic interface, through which a user transmits information by deforming it. Both the syntax and the semantics of the input patterns can be speciÞed by the user. Outputs from an array of capacitors embedded in the membrane are fed through a series of convolutional neural networks trained to localize interactions, such as the Þnger press shown in (a), as well as recognize abstract events, such as pinching (b) and twisting (c), which evolve over time, yet constitute discrete inputs.

changing interfaces, haptics, stretchable sensing, as well as literature from the deep learning and statistical machine learning communities on which our approach is motivated. Section 3 covers the design and fabrication of the OrbTouch device, whereas Section 4 covers the signal processing architecture, training methods, and training results. Section 5 provides an overview of the software implementation and highlights two example applications of OrbTouch. In Section 6 we provide a contextual overview of these results and also provide information theoretic analyses of our training data to better understand the information density in our interface, and its potential to be used for more sophisticated functions. Finally, Section 7 concludes the article by brießy discussing future research directions and associated challenges.

Related work

User interfaces provide an interactive window between physical and virtual environments. In tradition, the tactile interface facilitating this interaction has been capacitive touch screens, keyboard buttons, and the computer mouse. Making physical interaction more rich, both in terms of expanding the type and complexity of inputs that are available to the user, as well as the physical rendering of virtual objects, is of fundamental interest to the Þelds of humanÐcomputer interaction (HCI), humanÐrobot interaction (HRI), and virtual reality (VR).

Recently, researchers have started to adopt strategies from the Þeld of soft robotics[1] to augment the touch experience, creating tangible interactions that go beyond tapping, swiping, and clicking. Follmer *et al.*[2] used the concept of particle jamming, developed by Rodenberg and Amend,[3] to create a passive haptic interface that the user can free-form shape and then freeze in place. More recently, Stanley *et al.*[4] developed an active version of this interface, which dynamically renders 3D topologies using a grid of connected rubber cells controlled by pneumatic inputs, particle jamming, and a spring-mass-based kinematic model. Deformable haptic interfaces are a promising area of research with opportunities to leverage microßuidic technologies[5] to enable shape-changing interfaces for teleoperations, VR, and braille displays.

In addition to using soft haptic interfaces for physicalization, there are efforts to understand how we can use the passive dynamics of deformable materials,[6,7] and even the human epidermis,[8,9] as a medium for communication. A signiÞcant challenge in this pursuit pertains to sensing Þnite deformation in the compliant medium, as well as signal processing and software for robust mapping of sensory data to continuous states, for functions such as Þnger tracking, as well as discrete states to recognize user intent or emotion. Patel *et al.*[10] developed a passive haptic ball with embedded accelerometers and an outer enclosure containing ßexible capacitors. They used an extended Kalman Þlter to estimate ball orientation and Þnger positions using their bimodal sensor input. Han and Park[11] created a conceptually similar device and demonstrated the ability to recognize different grips with a classiÞcation accuracy of * 98% using a support vector machine (SVM) classiÞer. Tang and Tang[12] developed a dome-shaped foam interface and used Hall-effect sensors positioned around the base of the interface to capture a set of predeÞned interactions. In perhaps the most simple approach, Nakajima *et al.*[13] placed a microphone and a barometer inside of a balloon and were able to discriminate grasps, hugs, punches, presses, rubs, and slaps with a mean classiÞcation accuracy of 81.4% using an SVM classiÞer. Vision-based sensing has also been explored. Harrison and Hudson[14] used an infrared camera, placed behind the interface to capture a bottom-up view of the deforming membrane, in conjunction with blob detection algorithms to localize touch interactions. Other researchers have used vision with different interface designs.[15] Although vision-based sensing is inherently high dimensional and sensitive to deformation, focal length and camera placement impose two very significant constraints on the system design.

Both the human somatosensory system and capacitive touch displays alike beneÞt from high-dimensional tactile sensory input. It is our view that, by embedding sensors directly into the touch surface, we will similarly enable the widest range of functional soft interface designs. To accomplish this, we can leverage stretchable electronics,[16] which has enabled new capabilities across many applications such as in vivo biosensing,[17] robotics,[18] and soft robotics.[19] Charge conduction in stretchable media can be achieved using many different strategies, such as back Þlling channels embedded in elastomers with low melting point liquid eutectic alloys[20] or ionically conducting hydrogel polymers,[21] depositing silicon thin Þlms with serpentine patterns to

enable them to stretch by uncoiling,[22] and using CNTs. Yamadaet al.[23] and Lipomi et al.[24] recently made transparent electrode Þlms that remain conductive to within one order of magnitude by aerosol spraying a dilute suspension of CNTs in N-methylpyrrolidone onto a polydimethylsiloxane (PDMS) substrate. This combination of high conductivity at high strains, coupled with ease of fabrication, makes CNTs an excellent choice for shape-changing user interfaces.

In additional to improved sensing methods, there is a simultaneous need for robust signal processing architectures that are suited for stretchable electronics. As evidenced by recent trends in computer vision and deep learning,[25] enabling tactile sensing machinery to reason about the physical world in a meaningful way will likely require high-capacity models that learn from data efÞciently. This is important for emerging touch-sensing methods in VR,[26] wearable sensing,[27] HRI,[28] and HCI[29] that are being used for increasingly complex recognition tasks. Systems based on deep neural networks have surpassed, or are approaching, human capabilities in a number of areas including the classiÞcation and segmentation of both natural and medical images,[30] playing Atari games,[31] playing high complexity board games,[32] interpreting natural language,[33] and sequence recognition.[34] ArtiÞcial neural networks are known for their representational power, and convolutional Þltering is particularly suited for inputs that are spatially or temporally correlated. Like pixels in an image, sensors distributed throughout deformable bodies exhibit behaviors (e.g., spatial correlation) that make convolutional Þltering a suitable processing technique for feature extraction; this observation informs the modeling approach taken in this study.

Materials and Methods

Our shape-changing interface, OrbTouch (Fig. 2a), consists of a pressurized silicone orb with an embedded array of stretchable CNT capacitors. Each CNT electrode is bonded to an external copper lead that is routed through an analogÐdigital converter (ADC) to the general purpose input output interface on a Raspberry Pi 3 (RBPI3; Fig. 2b). To train the device, there is a push button adjacent to the interface that the user presses during training to supplement the logged data with ground truth labels. Models are trained offline and then uploaded onto the RBPI3, which computes them directly in the sensor measurement loop in real time. In addition to computing neural networks, we use the RBPI3 to control the sensing peripherals as well as host communication through Bluetooth.

Sensor fabrication

Figure 3 shows the internal construction and conÞguration of the CNT dielectric elastomer sensors and OrbTouch membrane. Each sensor consists of a parallel plate capacitor with two blended multiwalled carbon nanotube (MWCNT)Ð single-walled carbon nanotube (SWCNT) thin Þlm electrodes separated by a PDMS dielectric layer. The electrodes are patterned by aerosol spraying a dispersion of the CNTs in a solution of 2-propanol and toluene through a stencil on the base PDMS substrate (adapted from previous work[24]). Our process is performed in several steps: (1) in a beaker, a blended mixture of MWCNT (P/N 724769; Sigma Aldrich Corp.) and SWCNT (P/N P3-SWNT; Carbon Solutions, Inc.) is dispersed in a solution of 2-propanol (P/N 278475; Sigma Aldrich Corp.) and toluene (P/N 244511; Sigma Aldrich Corp.) (10 vol.% toluene) at a concentration of 0.05 wt.% using a centrifugal mixer (SR500; Thinky U.S.A., Inc.) in combination with ultrasonic agitation. (2) A 0.5 mm layer of silicone rubber (Ecoßex-0030; Smooth-on Corp.) is cast

FIG. 2. Photographs of the OrbTouch device. (a) Its embedded capacitors capture shape changes caused by human touch. (b) The internal components of OrbTouch consist of an embedded RBPI3 computer, ADC, and air compressor used to control pressure in the orb. ADC, analogÐdigital converter. Color images are available online.

FIG. 3. Membrane and sensor architecture. The interface is composed of upper and lower PDMS encapsulation layers, upper and lower carbon nanotube Þlm electrodes, and a 0.5 mm PDMS dielectric layer, yielding a total thickness of 2 mm. The sensors are conÞgured into a passive matrix, where each electrical lead in the grid measures 5 mm, yielding an overall density of 1 sensor/cm^2. PDMS, polydimethylsiloxan.

onto an acrylic sheet and cured. (3) A layer of polypropylene adhesive tape (S-423; Uline Corp.) is overlaid onto the substrate and a laser cutter (Zing 24; Epilog Laser Corp.) is used to selectively remove portions of it to form the bottom electrode pattern. (4) The CNT dispersion is sprayed through the mask with an airbrush (eco-17 Airbrush Master; Master, Inc.) to form the bottom electrode. Several coats are applied until each trace reaches an end-to-end resistance * 1 kO. (5) The mask is then removed and a thin (* 0.5 mm) dielectric layer (Ecoßex-0030) is cast over the entire substrate and cured. (6) Steps 3–5 are repeated (in reverse order) to form the top half of the membrane (overall thickness * 2 mm). (7) External copper leads are attached to each of the 10 CNT electrodes and connected to the ADC and RBPI3.

Sensing method

The sensing grid is designed as a passive matrix that enables us to position 25 sensors over the surface using only 10 electrical connections. To measure capacitance, we use the digital I/O pins on the RBPI3 and an ADC. To isolate the $(i, j)^{th}$ sensor, where $i, j \in \{0,1,2,3,4\}$, we set the i^{th} electrode to +3.3 VDC (vertical orientation, Fig. 4a), and monitor the corresponding voltage change on the j^{th} electrode (horizontal orientation, Fig. 4a), with the remaining electrodes connected to ground on the RBPI3 chassis to reduce cross-talk and interference. Figure 4b shows the equivalent circuit of the measurement. The capacitance in our sensor grid is 41.2 pF (standard deviation [SD] = 2.9 pF). We use a 50 MO resistor to achieve a nominal resistor-capacitor time constant of * 2 ms. When the i, j^{th} sensor is being measured, the i^{th} column electrode is set to +3.3 VDC, whereas the j^{th} row electrode, which is routed through the ADC, is disconnected from ground. A second capacitor (1 pF) is placed in series with the j^{th} row electrode and the ADC to shift the polarity of V_m into the 0–3.3 V range for the RBPI3.

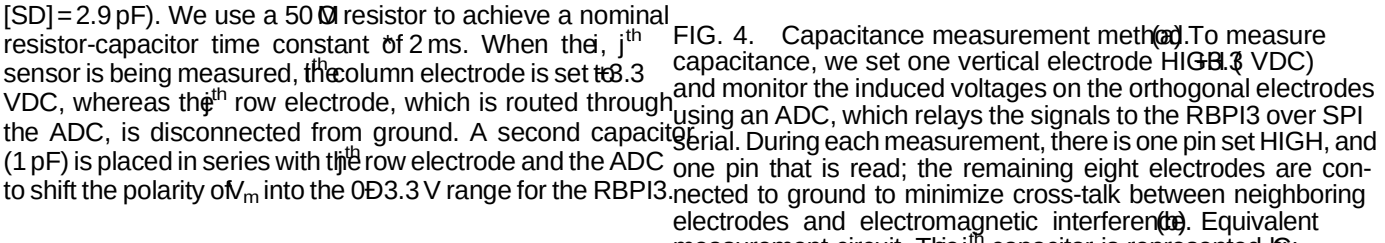

FIG. 4. Capacitance measurement method. (a) To measure capacitance, we set one vertical electrode HIGH (+3.3 VDC) and monitor the induced voltages on the orthogonal electrodes using an ADC, which relays the signals to the RBPI3 over SPI serial. During each measurement, there is one pin set HIGH, and one pin that is read; the remaining eight electrodes are connected to ground to minimize cross-talk between neighboring electrodes and electromagnetic interference. (b) Equivalent measurement circuit. The i, j^{th} capacitor is represented by $C_{i,j}$. The nominal capacitance of our sensors is 41.2 pF (standard deviation = 2.9 pF). We use an R_m = 50 MO inline resister to yield an RC time constant of * 2 ms. We use a second capacitor, C_m = 1 pF, to ßip the polarity of the measured V_m() voltages. RC, resistor-capacitor; SPI, serial peripheral interface; VDC, volts direct current. Color images are available online.

Results

Deformation–capacitance model

The sensors in OrbTouch behave according to the parallel-plate capacitance formula, $C = f A/d_t$, where C is the capacitance of the sensor, A is the surface area of the sensor, and d_t is the dielectric thickness. To validate this experimentally, we develop a simple model of capacitance for incompressible inßating shells, and compare its predictions to measured values that we obtain by inßating the interface.

We Þrst deÞne three principle stretches, k_1, k_2, and k_3, using a Cartesian basis as shown in Figure 5a. In an incompressible (i.e. $k_1 k_2 k_3 = 1$) rubber dielectric under equibiaxial tension (i.e. $k = k_1 = k_2$), the fractional change in capacitance is a function of only its radial stretch,

$$\frac{C}{C_0} = \frac{k_1 k_2}{k_3} = k_1^2 k_2^2 = k^4 \qquad (1)$$

Because it is difÞcult to measure k experimentally, we derive an alternative to Equation (1) that depends on the membrane deßection, d_{def} (Fig. 5a), which we can measure, using the well-known approximation,

$$A_{orb} \approx \frac{r^{16/5} + 2\, r d_{def}^{8/5}}{3}^{5/8}, \qquad (2)$$

which expresses the surface area of the hemispheroidal orb, A_{orb}, in terms of its radius, r, and d_{def}. If we assume that the deformation is homogeneous over the entire membrane as it inßates, we can alternatively express the quartic stretch term as $k^4 = (A_{orb}/A_{orb,\,0})^2$, where the nominal surface area is simply given by $A_{orb,\,0} = \pi r^2$. Combining these expressions with Equation (2) yields the desired relationship between fractional change in capacitance and d_{def}.

$$\frac{C}{C_0} \approx \left[\frac{1}{3} + \frac{2}{3}\left(\frac{d_{def}}{r}\right)^{8/5}\right]^{5/4}. \qquad (3)$$

Figure 5b plots the mean capacitance of our 5 × 5 capacitor grid versus our parameterized function $k^4(d_{def}, r)$, under

FIG. 5. Relationship between deformation and capacitance in the orb (a) Free body diagram of the touch membrane in the undeformed (deßated) and deformed (inßated) states. Under inßation we assume equibiaxial tension, and thus, because the membrane is incompressible, its stretch state is fully described by the radial stretch λ. (b) Plot of C/C_0 versus k^4 ($n = 25$). Color images are available online.

controlled inßation. The observed behavior undershoots our prediction; this has been observed previously,[21] and is commonly attributed to a decrease in dielectric permittivity that occurs in elastomers as they are stretched. We also note two other potential sources of error, the Þrst being our approximation of the orb as a hemispheroid (ref. [35]). Second, we assume that the deformation in the orb is homogeneous, however, sensors near the perimeter of the membrane are closer to the clamped boundary and, therefore, deform differently than sensors near the center. Although we use a simpliÞed model, the general relationship between capacitance and quartic stretch is quasi-linear, as predicted. We also note that each sensor in the grid is well deÞned, varying monotonically with the quartic radial stretch. This behavior suffÞces for our application, as we use these sensors to learn latent representations of deformation with neural networks, not for explicit shape estimation.

Model architecture

Our signal processing architecture is designed for modular touch interaction, enabling one to fully deÞne both the syntax and semantics of a set of inputs for a given application. We build this capability on top of two core functions: gesture recognition and touch localization, both of which are implemented using light weight CNNs. As inputs to our models we use sensor images that are computed as follows: $z = C/C_0 - 1$ ($z \in \mathbb{R}^{5 \cdot 5}$), where C_0 is the mean baseline capacitance taken over a 10 s interval at the beginning of each session. For gesture recognition, we use an inference model based on a 3D-CNN (F_1), to map a queue of n_t sensor images $z_0 : z_9$, to a categorical probability distribution p_c, over n_c gesture classes ($\mathbb{R}^{5 \cdot 5 \cdot 10} \mapsto \mathbb{R}^{n_c}$). We use F_1 to identify gestures, and also to discriminate between users performing the same gesture. For touch localization, we use a regression model (F_2) that uses 2D convolutions, which map sensor readings, z, from one time step to a continuous d-dimensional space ($\mathbb{R}^{5 \cdot 5} \mapsto \mathbb{R}^d$). We use F_2 to estimate touch location on the curvilinear surface (i.e., $d = 2$); however, it could also be used to estimate membrane deßection, touch pressure, or other continuous quantities.

Figure 6 shows the architectural features of F_1 and F_2 models. F_2 convolves its kernels over the spatial dimensions

FIG. 6. Computational graph of the inference (F_1) and regression (F_2) models. Both networks have two hidden convolutional layers and two hidden fully connected layer. The kernel size, k, and stride, s, of each convolutional operation are provided. Network F_1 accepts as input a sliding window of $k = 10$ discrete sensor readings ($5 \cdot 5 \cdot 10$; bottom) and outputs a probability distribution over n_c classes using a softmax activation on the output. Because the information in a gesture is spatiotemporal, we convolve a 3D kernel over both the spatial and temporal dimensions of the input to capture relevant features. Network F_2 accepts a $5 \cdot 5$ sensor matrix and outputs a continuous valued vector using a tanh activation on the output layer. 3D, three-dimensional. Color images are available online.

of the input, whereas F_1 convolves 3D kernels over the spatial and temporal dimensions to capture the dynamics of the touch gesture. Equation (4) provides an algebraic representation of the convolutions in these networks,

$$a^l_{mijk} = r\left(w^l_m * a^{l-1} + b^l_m\right)_{ijk} = r\left(b^l_m + \sum_n \sum_q \sum_r a^{l-1}_{n,i-q,j-k} w^l_{mnqr}\right), \quad (4)$$

where a^l_{mijk} refers to the $(i, j)^{th}$ node in the m^{th} feature map in layer l, w^l_m and b^l_m are the convolutional kernel and bias terms corresponding to the m^{th} feature map in layer l, respectively, the operator * denotes the convolution between the kernel and its input, and a^{l-1} represents the zero-padded input to layer l (we employ same padding). Equation (4) is valid for both the 2D and 3D convolutions (the time dimension, indexed by k, in the F_2 is singleton). The dense layers after the convolutional layers are mapped using the inner product of the weight matrices with the nodes from the preceding layer. F_1 uses a softmax activation in its output to produce a probability over gesture classes, where F_2 uses a tanh activation to regress continuous valued coordinates of touch. Both networks use interior rectified linear unit (relu) activations.

To run these models on the RBPI3 in real time, we had to consider trade-offs between model depth, number of time steps in the input, t, and sampling rate, x. Ideally we would use deep models in combination with a high-bandwidth input; however, we cannot simultaneously maximize model depth, t, and x in our compute- and time-constrained system. Through observing different users, we noticed that touch gestures are typically ~ 1 s in duration. Using $x^{-1} = 1$ s as a constraint, we found that a window of $t = 10$ and a sampling rate of $x = 10\ s^{-1}$ allow us to capture the relevant features from gestures. To enable the system to run safely at a latency of <100 ms, we use relatively shallow neural networks each with two convolutional layers and two fully connected layers.

Optimization methods and training results

We teach OrbTouch new inputs by pressing the label button, located adjacent to the orb (Fig. 2a), in unison with the imparted gesture. The label button is connected to the I/O interface on the RBPI3 computer, and its state is logged at every time step. We optimize models F_1 and F_2 stochastically on the logged data using an external computer, and then upload the trained parameters back onto the RBPI3 to use the device as a touch controller. To demonstrate this process, we define a set of five simple inputs: a finger press, a clockwise twisting motion, a counterclockwise twisting motion, a pinching motion, and a null input. We collected ~ 5 min of labeled training data for each of the mentioned input classes, yielding $n = 1.75 \cdot 10^4$ total examples. The parameters in F_1 are optimized using the categorical cross-entropy loss, ℓ_{CE} [Equation (5)], with two-norm regularization applied to its weights, where l indexes the layers in the network and m indexes the feature maps in layer l. We used mini-batches of $n = 150$ and regularization constants $k_{CE1} = 5 \cdot 10^{-4}$, $k_{CE2} = 1 \cdot 10^{-5}$. Optimization was implemented using the adaptive momentum estimation algorithm from Kingma and Ba.[36]

$$\ell_{CE}(h(z)) = \frac{1}{n}\sum_{i=0}^{n} (y \ln(h(z)) + (1-y)\ln(1-h(z))) + k_{CE1}\sum_{l=1}^{2}\sum_{m=0}^{M} w^l_m k_2^2 + k_{CE2}\sum_{l=3}^{5} w^l k_2^2. \quad (5)$$

We performed all training offline on a single graphics processing unit (GPU) (GeForce GTX 1080 Ti, NVIDIA Corp.) using the Tensorflow framework.[37] Figure 7a plots the training and validation accuracy of F_1 versus training epoch. F_1 reaches a test accuracy of 98.8% after ~ 500 epochs. Figure 7b plots the learning curve between this model and data set, indicating that the model achieves >95%

FIG. 7. CNN training results. (a) Plot of binary classification accuracy versus training epoch for F_1 on the gesture recognition data set. We measure a test accuracy of 98.8% after $5 \cdot 10^2$ epochs ($n = 1.75 \cdot 10^4$). (b) Learning curve of F_1 on the gesture recognition data set. (c) Plot of binary classification accuracy versus training epoch for CNN-3D on the user identification data set. We measure a test accuracy of 97.6% after $6\ 10^2$ epochs ($n = 5 \cdot 10^3$). (d) Learning curve of CNN-3D on the user identification data set. (e) Plot of the mean absolute error of CNN-2D on the touch location data set, measured in millimeters, for $\sim 2 \cdot 10^3$ epochs ($n = 2.85 \cdot 10^4$). (f) Learning curve of CNN-2D on the touch location data set. 2D, two-dimensional; CNN, convolutional neural network. Color images are available online.

classification accuracy using $5 \cdot 10^3$ examples, which is the equivalent of $\sim$ 10 min of training.

In addition to gesture recognition, we also trained F_1 to identify, from a set of $n_c = 10$ users, the person interacting with the device. In this experiment, each participant performed the clockwise twisting motion, as defined previously, for $\sim$ 5 min. We then trained F_1 using hyperparameters similar to those used for the gesture recognition data, achieving a test accuracy of 97.6% (Fig. 7c). Figure 7d plots the learning curve for this data set. We observe only a marginal decrease in test accuracy on the user recognition data set despite its larger number of output classes ($n_{c,user} = 10$ vs. $n_{c,gesture} = 5$) and much more nuanced differences between the $n_{c,user}$ classes. In both cases, we believe our model capacity is limited primarily by our manual labeling method, which introduces noise into our response variable due to nonuniform shifts between ground truth labels and the imparted gestures.

To train the F_2 model, we had a user visually locate the sensors on the membrane and press them (on, off) for a total of $\sim$ 30 min ($n = 1 \cdot 10^4$). We use ridge regression [Equation (6)] to optimize the parameters of F_2 using the Nesterov accelerated gradient algorithm from Nesterov.[38] Figure 7e plots mean absolute error (MAE) versus training epoch; we achieve a test error of MAE $\approx$ 0.09 mm, whereas Figure 7f plots the learning curve for this data set. Our best convergence and training performance were achieved using mini-batches of $n = 128$, gradient clipping ($\|V_{global}\|_2 \leq 10.0$), regularization constants $k_{MSE1} = 1 \cdot 10^{-5}$, $k_{MSE2} = 5 \cdot 10^{-6}$, and by adding zero-mean Gaussian noise ($\sigma = 0.05$ mm) to each ground truth label. For simplicity, we report distances with respect to the undeformed membrane that lies in two dimensions (i.e., its circular state), where the touch surface spans the x-y interval [(0, 0), (4,4)] mm. Thus, for a membrane deflection of $d_{def} = r$, a multiplicative factor of $\pi/2$ provides an approximation of the true error along the curvilinear surface of the orb.

$$\ell_{MSE}(h(z)) = \frac{1}{n}\sum_{i=0}^{n}(y - h(z))^2 + k_{MSE1}\sum_{l=1}^{2}\sum_{m=0}^{M} w_m^l k_2^2 + k_{MSE2}\sum_{l=3}^{5} w^l k_2^2. \quad (6)$$

To demonstrate how these models can be integrated into software applications, we use OrbTouch to play the popular video game Tetris (Fig. 8a). The objective of Tetris is to place a random cascade of falling pieces, or Tetrominos, into a bounding rectangle without filling it up; filling a row causes the Tetrominos in that row to disappear, allowing the pieces above it to drop and thus preventing the game board from filling. During game play, we use OrbTouch to translate (Fig. 8b, e) and rotate (Fig. 8c, d) the Tetrominos as they fall using the gestures that we defined in Section 4. We implement this with a C++ program running on the RBPI3, which executes sensor measurements, neural network computation, and Bluetooth communication with the host (Fig. 8f). We enqueue sensor measurements into a 1 s memory buffer, which gets passed to F_1 and F_2 at each time step. The user's gestures are recognized by computing $\text{argmax}(p_g)$. When a finger press is predicted, F_2 is used to estimate the location of touch, from which an appropriate translation is generated. Because the output from F_1 is noisy (error rate $\approx$ 1.2%), during game play we pass it

FIG. 8. Application of OrbTouch to the popular game Tetris. (a) Photograph of OrbTouch being used to control an adaptation of the game Tetris. (b) Finger pressing or poking is used to translate the Tetromino left, right, and down (L,R,D). (c) Pinching is used to drop the Tetromino directly to the bottom of the grid. (d) Clockwise rotation, or twisting, is used to rotate the Tetromino 90 in the clockwise direction. (e) Counterclockwise rotation is used to rotate the Tetromino 90 in the counterclockwise direction. (f) OrbTouch software diagram. The first processing step executes capacitance measurements, filters the signal, and F_2 and F_1), whereas the second step generates a command and updates the model inputs for the next time step. Each of these steps is multithreaded. We use debouncing filter before sending commands to the host (through Bluetooth). Each cycle of compute takes $\sim$ 86 ms, which fits within our 100 ms target. Color images are available online.

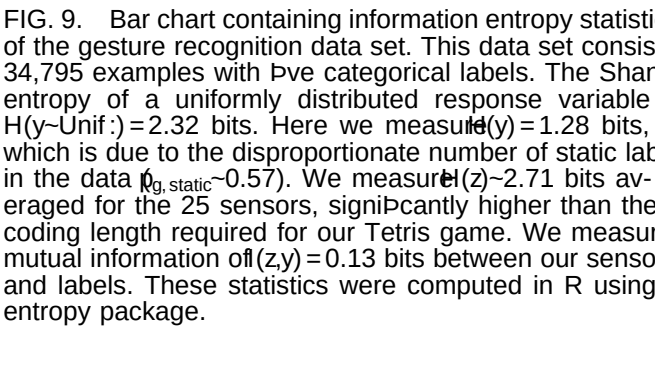

FIG. 9. Bar chart containing information entropy statistics of the gesture recognition data set. This data set consists of 34,795 examples with Þve categorical labels. The Shannon entropy of a uniformly distributed response variable is $H(y \sim Unif:) = 2.32$ bits. Here we measure $H(y) = 1.28$ bits, which is due to the disproportionate number of static labels in the data ($p_{g,static} \sim 0.57$). We measure $H(z) \sim 2.71$ bits averaged for the 25 sensors, signiÞcantly higher than the encoding length required for our Tetris game. We measure a mutual information of $I(z,y) = 0.13$ bits between our sensors and labels. These statistics were computed in R using the entropy package.

through a secondary debouncing Þlter, which in turn relays commands asynchronously to the host.

Movie 1* shows a person performing a random sequence of the Tetris gestures, along with the real-time output of F_1 (trained on the gesture recognition data set). We achieve nearly error-free gesture recognition with OrbTouch using F_1 in combination with the debouncing Þlter. This system runs at a controlled latency of 100 ms, which could be decreased significantly through the use of a GPU.

Movie 2{ shows a recording of a Tetris game, in which both F_1 and F_2 are used to generate game commands. The game is controlled using Þnger presses (Fig. 8b) to translate the Tetromino (left, down, right), pinching (Fig. 8e) to drop the Tetromino directly to the bottom of the board, clockwise twisting (Fig. 8d) to rotate the Tetromino 90 in the clockwise direction, and counterclockwise twisting (Fig. 8c) to rotate the Tetromino 90 in the counterclockwise direction. The OrbTouch controller runs as a standalone device, and wirelessly communicates with our Tetris application (written in Python) that runs externally on a laptop computer.

Information theoretic analysis of sensor signals

Out Tetris commands only require $\log_2(5) = 2$ bits of information to encode (including the null input), which raises the question of whether OrbTouch is capable of encoding more interesting vocabularies of higher perplexity. The performance of F_1 on the user identiÞcation data set ostensibly indicates a lower bound of $\log_2(10) = 3.32$ bits of information in our multivariate sensor signal; however, to gain a more complete understanding of its theoretical limits we consider the complexity of the sensor signals. We evaluate the information content by computing the Shannon entropy, $H(z)$,

$$H(z) = \sum_{i=1}^{n} p(z_i)\log_2(p(z_i)) \tag{7}$$

and mutual information, $I(z,y)$,

$$I(z,y) = \sum_{i=1}^{n}\sum_{j=1}^{n} p(z_i,y_j)\log_2 \frac{p(z_i,y_j)}{p(z_i)p(y_j)} \tag{8}$$

of the capacitance data, z, and labels, y, in the gesture recognition data set ($n = 34{,}795$), where $p(z)$ and $p(z,y)$ represent the marginal and joint probability masses, respectively. To compute $p(z)$ and $p(z,y)$, we Þrst project the data and labels onto the interval [0,1] using minÐmax normalization, $z = (z - z_{min})/(z_{max} - z_{min})$, for each sensorÐgesture combination in the data set, and then concatenate the data for each sensor into a vector of length 34,795. The data and labels are then quantized into 25-bin histograms.

Figure 9 shows a bar chart of the $H(y)$, $H(z)$, and $I(z,y)$ statistics. The complexity of our response variable can be interpreted as follows. Relative to the maximum entropy case in which all Þve of our gesture classes occur in equal proportion, that is, $H(y \sim Unif:) = \log_2(5) = 2.32$ bits, the complexity of our response variable is significantly lower, $H(y) = 1.28$ bits. We expect this given the disproportionate number of static labels in the gesture identiÞcation data set ($p_{g,static} = 0.57$). In contrast, we compute a mean signal entropy of $H(z) = 2.71$ bits, averaged for the 25 sensor channels, indicating that each sensor in OrbTouch contains a surplus of information relative to y. Thus, given near optimal encoding of our signal, we theoretically could play Tetris using only one of our sensors. We also use these data to compute the relative entropy between the response variables and covariates, which is a measure of the decrease in uncertainty (in bits) of our response when it is conditioned on the input. We observe a relatively low mutual information, $I(z, y) = 0.13$ bits, which tells us that although our per-sensor signal entropy is high relative to our response, not all of that information is predictive of the response.

Although these statistics are computed on time series from individual sensors, the multivariate entropy and mutual information, taken over the 250 dimensional input of F_1, would provide a better estimate of the information that is available to our classiÞer. Owing to the curse of dimensionality, however, estimating the multivariate probability masses is computationally intractable using our quantization method. The effects of spatial and temporal correlation in these data also make it difÞcult to estimate the true information content in the multivariate signal using these univariate and bivariate statistical measures. In future work, we intend to explore more advanced estimation methods, such as Markov chain Monte Carlo sampling, to better understand the information in our system, and also to inform better sensor and signal processing design. The high per-sensor entropy in our gesture recognition data (2.71 bits), though, is a promising step toward being able to encode large interesting vocabularies using deformable interfaces with high-density sensor arrays.

*https://youtu.be/lStMXNRmRqU
{https://youtu.be/82p35vj6M2A

Conclusions

This article explores the use of deformation in a compliant touch surface as a medium for communication. To demonstrate this concept, we present OrbTouch, a device that can learn multitouch inputs and localize Þnger presses, akin to a capacitive touch screen, but one that interprets shape change rather than Þnger movements. This is enabled by stretchable CNT-based capacitors that we embed inside of the touch surface to provide real-time shape feedback. Rather than use physical models to map sensor data to explicit representations of shape, we leverage deep neural networks, which learn latent representations of deformation, to directly map sensor signals to virtual states that a user can deÞne for their application.

The core of our approach lies in our use of 3D convolutions to capture spatiotemporal features in the gestural inputs. We initially considered other approaches to capture temporal information, such as using recurrent models with and without CNN-based feature extractors;[39] however, we found that gestures, and even short sequences of gestures, occur over relatively short time horizons. Our approach, therefore, is to expand the dimension of the input to encompass the relevant time horizon while retaining its spatial and temporal structure, and to use Þnite impulse response Þlters to capture the relevant spatial and temporal features. In the future, though, we are interested in expanding the gestural vocabulary to include longer sequences of inputs, which will require the use of recurrent models to capture contextual information.

OrbTouch highlights the utility of statistical approaches and learning algorithms in the rapidly expanding Þelds of stretchable electronics and soft robotics, and shows how they can be applied to HCI. Previous research in shape-changing interfaces, as well as stretchable electronics, has explored the use of machine learning for sensory mapping. To our knowledge, however, we have demonstrated for the Þrst time the use of stretchable sensors to control a software application in real time. We emphasize the distinction between achieving high performance metrics on in-sample data, for which it is very easy to overÞt, and demonstrating that the model generalizes to a real-time data feed such that it can be used to accomplish tasks. This is immensely important in this research area because many of the commonly used stretchable sensors exhibit hysteresis, nonstationarity, and high failure rates.

Although we focus on touch control for humanÐcomputer interfaces, we believe this approach can also be applied more generally in robotics. OrbTouchÕs skin could, for example, be overlaid onto a robot and integrated into its perception system, a step toward the level of sensor fusion that we observe in biological systems. A nearer term ambition would be incorporating the skin into robotic end effectors, such as a jamming gripper,[40] for robust identiÞcation and characterization of grasped objects. Furthermore, in robotics it is generally desirable to have higher dimensional sensing. We designed OrbTouch with 25 sensors, at a density of 1 cm^{-2}; however, this choice was motivated by our application and fabrication method. Decreasing the CNT electrode width to 500l m using commercially available inkjet printers,[41] for example, would yield 100 sensors/cm^{2}. With a mean per sensor entropy of 2.71 bits, skins that can sense at this resolution will be an important step toward improving physical perception in robots that use compliant materials.

The code and model parameters used in OrbTouch are available on Github.[42]

Acknowledgments

We thank K. OÕBrien, B. Peele, K. Petersen, and C.W. Larson for their comments, discussions, and insight. This study was supported by the Army Research OfÞce (Grant No. W911NF-15-1-0464) and the Air Force OfÞce of ScientiÞc Research (Awards No. FA9550-15-1-0160 and FA9550-18-1-0243).

Author Disclosure Statement

No competing Þnancial interests exist.

References

1. Rus D, Tolley M. Design, fabrication and control of soft robots. Nature 2015;521:467Ð475.
2. Follmer S, Leithinger D, Olwal A, et al. Jamming user interfaces: Programmable particle stiffness and sensing for malleable and shape-changing devices. In Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology. Cambridge, MA: ACM, 2012, pp. 519Ð528.
3. Rodenberg EBN, Amend J. Universal robotic gripper based on the jamming of granular material. Proc Natl Acad Sci U S A 2010;107:18809Ð18814.
4. Stanley AA. Deformable model-based methods for shape control of a haptic jamming surface. IEEE Trans Vis Comput Graph 2017;23:1029Ð1041.
5. Russomanno A, OÕModhrain S, Gillespie B, et al. Refreshing refreshable braille displays. IEEE Trans Haptics 2015;8:287Ð297.
6. Lee S-S, Kim S, Jin B, et al. How users manipulate deformable displays as input devices. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Cambridge, MA: ACM, 2010, pp. 1647Ð1656.
7. Rasmussen MK, Pedersen EW, Petersen MG, et al. Shape-changing interfaces: A review of the design space and open research questions. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Atlanta, GA: ACM, 2012, pp. 735Ð744.
8. Ogata M, Sugiura Y, Makino Y, et al. Senskin: Adapting skin as a soft interface. In Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology. St. Andrews, Scotland: ACM, 2013, pp. 539Ð544.
9. Weigel M, Mehta V, Steimle J. More than touch: Understanding how people use skin as an input surface for mobile computing. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Toronto, Canada: ACM, 2014, pp. 179Ð188.
10. Pai DK, VanDerLoo EW, Sadhukhan S, et al. The tango: A tangible tangoreceptive whole-hand human interface. In Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint, IEEE, Pisa, Italy, 2005, pp. 141Ð147.
11. Han S, Park J. Grip-ball: A spherical multi-touch interface for interacting with virtual worlds. In Consumer Electronics (ICCE), 2013 IEEE International Conference. Las Vegas, NV: IEEE, 2013, pp. 600Ð601.
12. Tang SK, Tang WY. Adaptive mouse: A deformable computer mouse achieving form-function synchronization. In CHIÕ10 Extended Abstracts on Human Factors in Computing Systems. Atlanta, GA: ACM, 2010, pp. 2785Ð2792.

13. Nakajima K, Itoh Y, Hayashi Y, et al. Emoballoon: A balloon-shaped interface recognizing social touch interactions. In Virtual Reality (VR), 2013 IEEE. Boekelo, The Netherlands: IEEE, 2013, pp. 1–4.
14. Harrison C, Hudson SE. Providing dynamically changeable physical buttons on a visual display. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Boston, MA: ACM, 2009, pp. 299–308.
15. Steimle J, Jordt A, Maes P. Flexpad: Highly flexible bending interactions for projected handheld displays. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. Paris, France: ACM, 2013, pp. 237–246.
16. Rogers JA, Someya T, Huang Y. Materials and mechanics for stretchable electronics. Science 2010;327:1603–1607.
17. Viventi J, Kim D-H, Vigeland L, et al. Flexible, foldable, actively multiplexed, high-density electrode array for mapping brain activity in vivo. Nature Neurosci 2011;14: 1599–1605.
18. Kim J, Lee M, Shim HJ, et al. Stretchable silicon nanoribbon electronics for skin prosthesis. Nat Commun 2014;5: 5747.
19. Larson CM, Peele B, Li S, et al. Highly stretchable electroluminescent skin for optical signaling and tactile sensing. Science 2016;351:1071–1074.
20. Park Y-L, Majidi C, Kramer R, et al. Hyperelastic pressure sensing with a liquid-embedded elastomer. J Micromech Microeng 2010;20:125029.
21. Keplinger C, Sun J-Y, Foo CC, et al. Stretchable, transparent, ionic conductors. Science 2013;341:984–987.
22. Khang D-Y, Jiang H, Huang Y. A stretchable form of single-crystal silicon for high-performance electronics on rubber substrates. Science 2006;311:208–212.
23. Yamada T, Hayamizu Y, Yamamoto Y. A stretchable carbon nanotube strain sensor for human-motion detection. Nat Nanotechnol 2011;6:296–301.
24. Lipomi DJ, Vosgueritchian M, Tee BC, et al. Skin-like pressure and strain sensors based on transparent elastic films of carbon nanotubes. Nat Nanotechnol 2011;6:788–792.
25. LeCun Y, Bengio Y, Hinton G. Deep learning. Nature 2015;521:436–444.
26. Shepherd R, Peele B, Murray BM, et al. Stretchable transducers for kinesthetic interactions in virtual reality. In ACM SIGGRAPH 2017 Emerging Technologies. ACM, 2017, p. 21.
27. Stoppa M, Chiolerio A. Wearable electronics and smart textiles: A critical review. Sensors 2014;14:11957–11992.
28. Hughes D, Lammie J, Correll N. A robotic skin for collision avoidance and affective touch recognition. IEEE Robot Autom Lett. 2018;3:1386–1393.
29. Roh E, Hwang B-U, Kim D, et al. Stretchable, transparent, ultrasensitive, and patchable strain sensor for human–machine interfaces comprising a nanohybrid of carbon nanotubes and conductive elastomers. ACS Nano 2015;9: 6252–6261.
30. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, NV, 2016, pp. 770–778.
31. Mnih V, Kavukcuoglu K, Silver D, et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.
32. Silver D, Huang A, Maddison CJ, et al. Mastering the game of go with deep neural networks and tree search. Nature 2016;529:484–489.
33. Mikolov T, Chen K, Corrado G, et al. Efficient estimation of word representations in vector space. arxiv:1301.3781, 2013.
34. Bengio S, Vinyals O, Jaitly N, et al. Scheduled sampling for sequence prediction with recurrent neural networks. In Proceedings of the 28th International Conference on Neural Information Processing Systems. Montreal, Canada, 2015, pp. 1171–1179.
35. Adkins J, Rivlin R. Large elastic deformations of isotropic materials ix. the deformation of thin shells. Philos Trans R Soc Lond A Math Phys Eng Sci 1952;244:505–531.
36. Kingma D, Ba J. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980, 2014.
37. Abadi M, Agarwal A, Barham P, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. arXiv preprint arXiv:1603.04467, 2016.
38. Nesterov Y. A method of solving a convex programming problem with convergence rate o (1/k2). Soviet Math Doklady 1983;27:372–376.
39. Ordóñez FJ, Roggen D. Deep convolutional and lstm recurrent neural networks for multimodal wearable activity recognition. Sensors 2016;16:115.
40. Amend JR, Brown E, Rodenberg N, et al. A positive pressure universal gripper based on the jamming of granular material. IEEE Trans Robot 2012;28:341–350.
41. Kordás K, Mustonen T, Tóth G, et al. Inkjet printing of electrically conductive patterns of carbon nanotubes. Small 2006;2:1021–1025.
42. Larson CM. Orbtouch. Available at: https://github.com/chrislarson1/orbtouch 2017 (accessed May 1, 2017).

Address correspondence to:
Robert Shepherd
Department of Mechanical Engineering
Cornell University
553 Upson Hall
Ithaca, NY 14853

E-mail: rfs247@cornell.edu