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

Chris Larson, Josef Spjut, Ross Knepper, and Robert Shepherd

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 defines 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 classifier recognizes five 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, human–computer 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 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 films, 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 $2n$ 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) filter 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 briefly discuss recent advances in shape-
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 briefly 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 fields of human–computer interaction (HCI), human–robot interaction (HRI), and virtual reality (VR).

Recently, researchers have started to adopt strategies from the field of soft robotics to augment the touch experience, creating tangible interactions that go beyond tapping, swiping, and clicking. Follmer et al. used the concept of particle jamming, developed by Rodenberg and Amend, to create a passive haptic interface that the user can free-form shape and then freeze in place. More recently, Stanley 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 microfluidic technologies 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, and even the human epidermis as a medium for communication. A significant challenge in this pursuit pertains to sensing finite 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 finger tracking, as well as discrete states to recognize user intent or emotion. Pai et al. developed a passive haptic ball with embedded accelerometers and an outer enclosure containing flexible capacitors. They used an extended Kalman filter to estimate ball orientation and finger positions using their bimodal sensor input. Han and Park created a conceptually similar device and demonstrated the ability to recognize different grips with a classification accuracy of ~98% using a support vector machine (SVM) classifier. Tang and Tang developed a dome-shaped foam interface and used Hall-effect sensors positioned around the base of the interface to capture a set of predefined interactions. In perhaps the most simple approach, Nakajima et al. 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 classification accuracy of 81.4% using an SVM classifier. Vision-based sensing has also been explored. Harrison and Hudson 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. 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 benefit 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, which has enabled new capabilities across many applications such as in vivo biosensing, robotics, and soft robotics. Charge conduction in stretchable media can be achieved using many different strategies, such as back filling channels embedded in elastomers with low melting point liquid eutectic alloys or ionically conducting hydrogel polymers, depositing silicon thin films with serpentine patterns to

FIG. 1. Illustration of the OrbTouch concept. A dome-shaped balloon is inflated 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 specified 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 finger 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.
enable them to stretch by uncoiling, and using CNTs. Yamada et al. and Lipomi et al. recently made transparent electrode films 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 addition 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, enabling tactile sensing machinery to reason about the physical world in a meaningful way will likely require high-capacity models that learn from data efficiently. This is important for emerging touch-sensing methods in VR, wearable sensing, HRI, and HCI 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 classification and segmentation of both natural and medical images, playing Atari games, playing high complexity board games, interpreting natural language, and sequence recognition. Artificial neural networks are known for their representational power, and convolutional filtering 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 filtering 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 analog–digital 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 configuration of the CNT dielectric elastomer sensors and OrbTouch membrane. Each sensor consists of a parallel plate capacitor with two blended multiwalled carbon nanotube (MWNT)–single-walled carbon nanotube (SWCNT) thin film 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).

Our process is performed in several steps: (1) in a beaker, a blended mixture of MWNT (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) An ~0.5 mm layer of silicone rubber (Ecoflex-0030; Smooth-on Corp.) is cast...
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 of ~1 kΩ. (5) The mask is then removed and a thin (~0.5 mm) dielectric layer (Ecoflex-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 \( j^{th} \) 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. In each measurement cycle, we set one vertical electrode HIGH (i.e., \( V_{dc} \)) whereas the \( j^{th} \) column electrodes 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 lift the polarity of \( V_{dc} \) into the 0–3.3 V range for the RBPI3.

Results

Deformation–capacitance model

The sensors in OrbTouch behave according to the parallel plate capacitance formula, \( C \propto A d \), where \( C \) is the capacitance of the sensor, \( A \) is the surface area of the sensor, and \( d \) is the dielectric thickness. To validate this experimentally, we develop a simple model of capacitance for incompressible inflating shells, and compare its predictions to measured values that we obtain by inflating the interface.

We first define three principle stretches, \( \lambda_1, \lambda_2, \) and \( \lambda_3 \), using a Cartesian basis as shown in Figure 5a. In an incompressible (i.e., \( \lambda_1 \lambda_2 \lambda_3 = 1 \)) rubber dielectric under equibiaxial tension (i.e., \( \lambda = \lambda_1 = \lambda_2 \)), the fractional change in capacitance is a function of only its radial stretch,

\[
\frac{C}{C_0} = \frac{\lambda_1 \lambda_2}{\lambda_3} = \frac{\lambda_1^2 \lambda_2^2}{\lambda_3} = \lambda^4
\]

(1)

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

\[
A_{\text{orb}} \approx \left( \frac{r^{16/5} + 2 (r d_{\text{def}})^{8/5}}{3} \right)^{5/8},
\]

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

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

Figure 5b plots the mean capacitance of our \( 5 \times 5 \) capacitor grid versus our parameterized function, \( \lambda^4(d_{\text{def}}, r) \), under
controlled inflation. The observed behavior undershoots our prediction; this has been observed previously, 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 first 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 simplified 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 defined, varying monotonically with the quartic radial stretch. This behavior suffices 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 define 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: \( z = \frac{C}{C_0} \) (\( z \in \mathbb{R}^{5 \times 5} \)), where \( C_0 \) is the mean baseline capacitance taken over a 10s 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 \( m \) sensor images, \( z_t : z_0 \), to a categorical probability distribution, \( p_c \), over \( n_c \) gesture classes (\( \mathbb{R}^{5 \times 5} \rightarrow \mathbb{R}^{n_c} \)). 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 (\( \mathbb{R}^{5 \times 5} \rightarrow \mathbb{R}^{d} \)). We use \( F_2 \) to estimate touch location on the curved surface (i.e., \( d = 2 \)); however, it could also be used to estimate membrane deflection, touch pressure, or other continuous quantities.

Figure 6 shows the architectural features of the \( F_1 \) and \( F_2 \) models. \( F_2 \) convolves its kernels over the spatial dimensions...
of the input, whereas the network and preceeding layer. Equation (4) provides an algebraic representation of the convolutions in these networks,

\[ d_{mijk} = \sigma \left( w_{ij}^l * a_{mij}^{l-1} + b_m^l \right)_{ijk} \]

\[ \leftarrow \sigma \left( b_m^l + \sum_n \sum_q \sum_r a_{i-n,j-q,k-r}^l m_{nqr} \right), \tag{4} \]

where \( d_{mijk} \) refers to the \((i,j)\)th node in the \(m\)th feature map in layer \(l\), \(w_m^l\) and \(b_m^l\) 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 \(a_{mij}^{l-1}\) 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, whereas \(F_2\) 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, \(\omega\). Ideally we would use deep models in combination with a high-bandwidth input; however, we cannot simultaneously maximize model depth, \(t\), and \(\omega\) in our compute- and time-constrained system. Through observing different users, we noticed that touch gestures are typically \(\sim 1\) s in duration. Using \(t=\omega=1\) s as a constraint, we found that a window of \(t=10\) and a sampling rate of \(\omega=10\) 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 \(\sim 5\) min of labeled training data for each of the mentioned input classes, yielding \(n=1.75 \times 10^4\) total examples. The parameters in \(F_1\) are optimized using the categorical cross-entropy loss, \(\mathcal{L}_{CE}\) [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 \(\lambda_{CE1} = 5 \times 10^{-4}\), \(\lambda_{CE2} = 1 \times 10^{-2}\). Optimization was implemented using the adaptive momentum estimation algorithm from Kingma and Ba.

\[ \mathcal{L}_{CE}(h(z)) = -\frac{1}{n} \sum_{i=0}^{n} \left( y_i \log(h(z_i)) + (1 - y_i) \log(1 - h(z_i)) \right) + \lambda_{CE1} \sum_{l=1}^{M} \sum_{i=0}^{l-1} \left\| w_m^l \right\|_2^2 \]

\[ + \lambda_{CE2} \sum_{l=3}^{5} \left\| w_l^l \right\|_2^2. \tag{5} \]

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 \(\sim 98.8\%\) after \(\sim 500\) epochs. Figure 7b plots the learning curve between this model and data set, indicating that the model achieves >95% classification accuracy versus training epoch for CNN-3D on the gesture recognition data set. 2D, two-dimensional; CNN, convolutional neural network. Color images are available online.
classification accuracy using $5 \times 10^3$ 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 ($n_{c,\text{user}}=10$ vs. $n_{c,\text{gesture}}=5$) and much more nuanced differences between the $n_{c,\text{user}}$ 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 \times 10^4$). We use ridge regression [Equation (6)] to optimize the parameters in $F_2$ 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=0.09 mm, whereas Figure 7f plots the learning curve for this data set. Our best convergence and training performance were achieved using minibatches of $n=128$, gradient clipping ($\|\nabla_{\text{global}}\|_2 \leq 10.0$), regularization constants $\lambda_{\text{MSE}1}=1 \times 10^{-5}$, $\lambda_{\text{MSE}2}=5 \times 10^{-6}$, and by adding zero-mean Gaussian noise (SD=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 $d_{xy}=r$, a multiplicative factor of $\pi/2$ provides an approximation of the true error along the curvilinear surface of the orb.

$$
\ell_{\text{MSE}}(h(z)) = \frac{1}{n} \sum_{i=0}^{n} (y - h(z))^2
+ \lambda_{\text{MSE}1} \sum_{i=1}^{2} \sum_{m=0}^{M} \|w_m\|_2^2
+ \lambda_{\text{MSE}2} \sum_{i=3}^{5} \|w_i\|_2^2.
$$

(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 $\text{argmax}(p_i)$. 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=1.2%), during game play we pass it...
through a secondary debouncing filter, 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 filter. 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 finger 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 identification 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, $H(z)$,

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

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

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

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 first project the data and labels onto the interval $[0,1]$ using min–max normalization, $z \leftarrow (z - z_{\min})/(z_{\max} - z_{\min})$, for each sensor–gesture 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 five of our gesture classes occur in equal proportion, that is, $H(y = \text{Unif.}) = \log_2(5) = 2.32$ 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 identification data set ($P_{\text{static}} = 0.57$). 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 $z$. 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 classifier. 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 difficult 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.
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 finger presses, akin to a capacitive touch screen, but one that interprets shape change rather than finger 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 define 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; 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 finite impulse response filters 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 fields 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 first 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 overfit, 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 human–computer interfaces, we believe this approach can also be applied more generally in robotics. OrbTouch’s 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, for robust identification 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 500 \(\mu\)m using commercially available inkjet printers, 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.

Acknowledgments

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

Author Disclosure Statement

No competing financial interests exist.

References


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

E-mail: rfx247@cornell.edu