TangibleTouch: A Toolkit for Designing Surface-based Gestures for Tangible Interfaces

Dominic Potts
Lancaster University
Lancaster, United Kingdom
d.potts2@lancaster.ac.uk

Martynas Dabravalskis
Lancaster University
Lancaster, United Kingdom
m.dabravalskis@lancaster.ac.uk

Steven Houben
Eindhoven University of Technology
Eindhoven, The Netherlands
s.houben@tue.nl

ABSTRACT
This pictorial introduces *TangibleTouch*, a toolkit to design and build interactive tangible cubes using capacitive sensing. This toolkit enables designers to quickly prototype and explore tangible cubes with exchangeable capacitive panels that allows touch-based gestures and interactions that can be used for tangible input in VR, AR, or physical computing. We introduce a design space for *TangibleTouch*, and present the technical implementation, fabrication approaches, and software support for designers. We evaluate our toolkit by demonstrating its use and application for 3 different case studies: a media controller, a platform game, and a 3D model inspection tool. The contribution of our work is a novel toolkit method for constructing and fabricating a cube-based interactive tangible user interface.

Authors Keywords
Tangibles; Gestural Interaction; Toolkit; Rapid Prototyping; Capacitive Sensing; Mixed Reality

CSS Concepts
Human-centered computing ~ User interface toolkits

TEI '22, February 13–16, 2022, Daejeon, Republic of Korea

ACM ISBN 978-1-4503-9147-4/22/02…$15.00
https://doi.org/10.1145/3490149.3502263

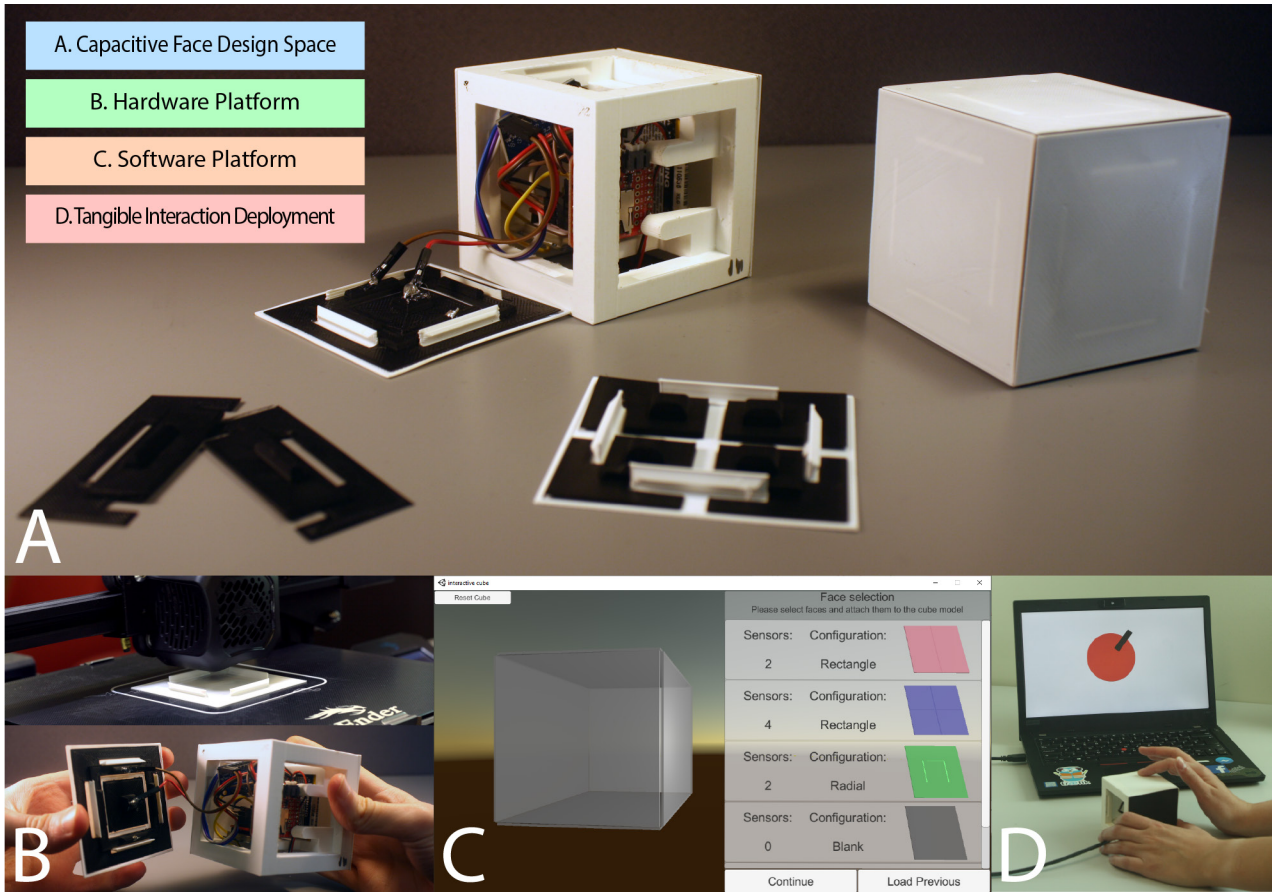


Figure 1. The *TangibleTouch* toolkit composed of: **A**) *Capacative Face Design Space*, **B**) modular *Hardware Platform*, **C**) *Software Platform* to train surface gestures, and **D**) *Deployable Interactions* to any Unity application.

INTRODUCTION

Tangible User Interfaces (TUI) are commonly adopted interaction devices for their innate affordances, manipulability, and mechanisms for rich input and output [27]. Cubic tangibles are a common form factor and widely adopted interaction device in tangible research, due to their physical simplicity, spatial stability, multi-functionality, and abstract nature [2]. A common means of interacting with tangibles, cubic or otherwise, is via physical manipulation and gesturing upon or with the object [1, 14]. Tracking and detecting interaction with a physical artefact as an expression of touch input can be difficult as it requires either sophisticated external tracking or instrumentation of the object. One popular approach to creating touch sensitive devices is by using capacitive sensing to detect complex surface-based gestures and even proxemic interaction [11, 4]. However, it is difficult to fabricate such capacitive tangibles that can be configured to support a variety of expressive touch gestures for different contexts and applications.

The *TangibleTouch* toolkit (Figure 1) is a novel fabrication method and rapid prototyping toolkit for designing modular, interactive cubic tangibles with interchangeable capacitive faces, facilitating a multitude of surface-based gestures. *TangibleTouch* provides the following: i) a modular and extendable hardware platform enabling the rapid fabrication of cubic tangibles capable of detecting different surface-based gestures, ii) a Unity[1]-integrated user interface that supports the configuration and mapping of capacitive faces on a given cube, and iii) a run time system to train surface-based gestures and detecting them in real-time.

In this pictorial we present the intersecting literature that is the basis of our work, visualise the interaction space for cubic tangibles, and explore the design of capacitive faces and the types of surface-based gestures they support. We then detail the design of the cube, the hardware implementation and fabrication process, and the components of the toolkit. Next, we evaluate the *TangibleTouch* toolkit through three demonstrative applications, displaying the range of surface-based gestures supported and highlighting the generality of the toolkit. Finally, we reflect on and discuss the features and implications of the toolkit.

1 https://unity.com/

BACKGROUND

Our work builds on different areas of related work including Tangible UI, cubic tangibles, fabrication methods, capacitive sensing, and toolkit research.

Tangible Prototyping & Cubes

Tangible User Interfaces (TUI) provide a physical means of representing and manipulating digital information [9, 27, 31], and there has been considerable work around prototyping TUIs without the need for instrumentation. For example, the work of Kelly et al. [17] and Zheng et al. [36] focused on producing popular TUI elements, such as knobs and sliders, using a combination of low fidelity material and fiducial-based computer vision tracking. Beyond low-fidelity prototyping, there are table top approaches that also use external computer vision tracking to build toolkits and development platforms for TUIs such as: *reacTable* [15] and *reacTIVision* [16], *Madgets* [32], and *SLAP Widgets* [33]. Rather than constructing tangible interface components from composite objects, another approach is to have an interactive artefact. For example, Savage et al. focused on interaction with physical objects while utilizing a computer-vision based tracking approach to detect user gestures and create 'active' objects [24].

Artefact-based interfaces are commonly used in Mixed Reality environments due to their manipulability and ability to be used as a proxy [2, 7, 8, 21, 37]. The work of Feick et al. [8] produced a toolkit for prototyping such tangible artefacts for virtual environments, using low-fidelity and modular 'shape primitives', to support proxy-based interactions. Beyond proxy interaction, the work of Angelini [1] and Van den Hoven et al. [14] shows how combining a tangible artefact with gestures, both motion and surface-based, can effectively recreate common UI elements using a single artefact. In addition, the benefits of cubes as tangible interaction devices have previously been explored [19, 28]. Particularly the work of Lefeuvre et al. [19] categorises the distinct affordances and properties of cubes by "*manipulation, placement in space, arrangement, multi-functionality, randomness, togetherness, physical qualities, containers, and pedestal for output*". Inspired by this work, *TangibleTouch* aims to build upon previous tangible prototyping toolkits with a first exploration of an emergent design space for prototyping surface-based gestures leveraging the benefits of cube affordance.

Capacitive Sensing & Fabricating Interactivity

Considering the above related work, we can describe two general approaches to sensing tangible interaction: (i) using an external tracking system or (ii) instrumenting an interface or user. While both approaches have limiting factors for prototyping, external tracking has challenges and requirements when detecting surface interaction on objects, primarily down to occlusion of the object and gestures [3, 12, 35]. Considering detecting interaction on instrumented objects, capacitive sensing is a common and well-documented approach, with particular benefits for rabid fabrication and prototyping [5, 11, 23, 25, 26]. The work of Schmitz et al. [26] and Burstyn et al. [5] explored capacitive input on 3D printed interactive objects, specifically leveraging conductive filament and dual-extrusion printing, to create discrete touch-sensitive areas on any 3D object. Capricate [26] in particular provided tools for designers to modify virtual models to be printed and instrumented with touch-sensitive areas.

Beyond 3D printing of capacitive objects, so-called 'loading mode' capacitive sensing has been adopted for gesture recognition [11]. Commonly, a number of capacitive areas are used to form capacitive sensor arrays to track touch across a designated area over time [10, 22, 29, 30]. Nelson at al. [22] incorporated 4 to 12 interactive areas to detect gestures on fabric and provided an initial exploration of different capacitive plate shapes and designs. For *TangibleTouch*, we aim to combine the fabrication techniques of 3D printed interactive objects with previous work on capacitive sensing for gesture recognition into one rapid prototyping toolkit.

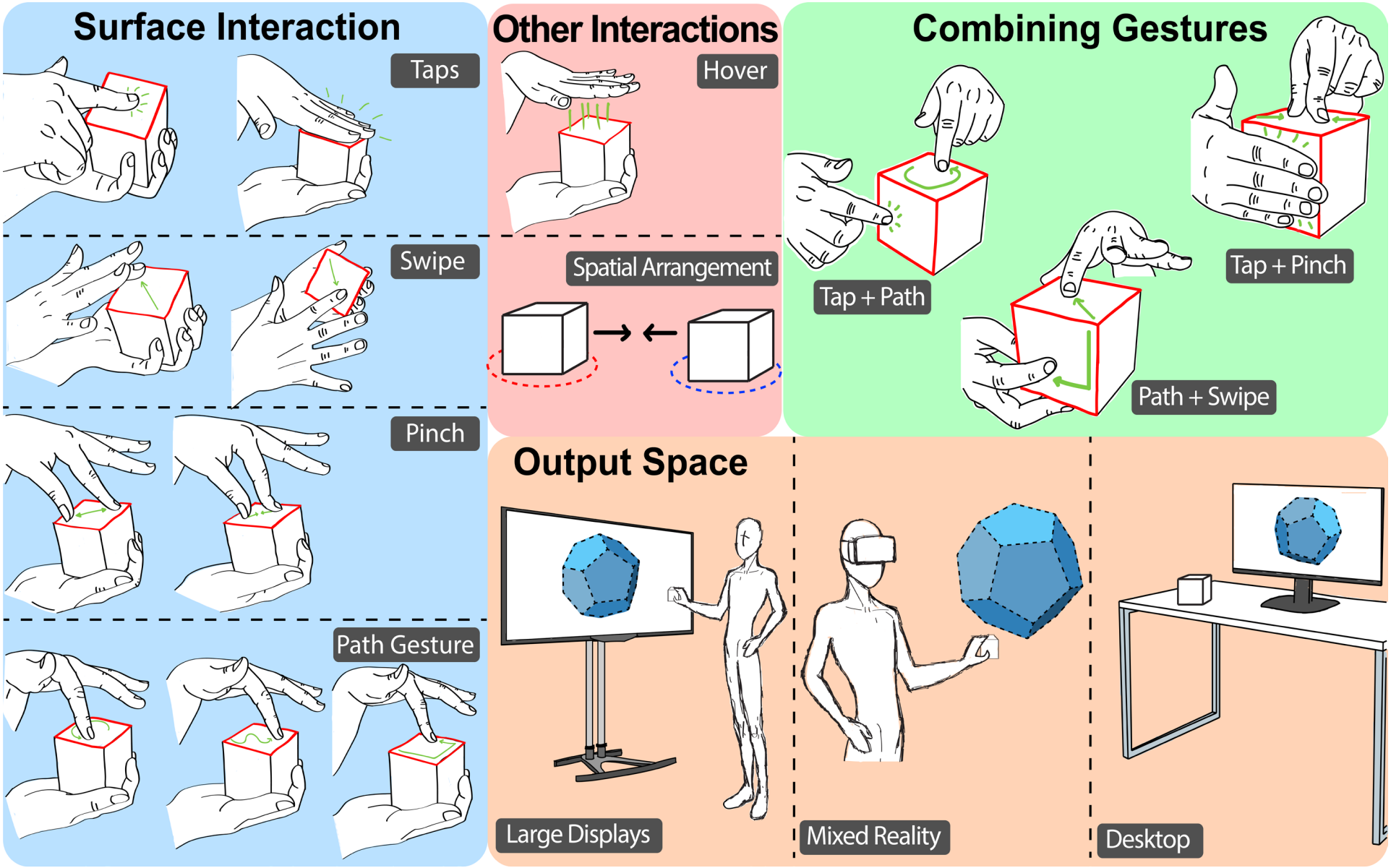


Figure 2. A visualisation of the interaction and output space combining cubes and surface-based gestures.

Toolkits

For prototyping tangible interaction, toolkits have been proposed as a productive approach to mitigate engineering challenges in building interfaces [18]. These toolkits range in levels of fidelity from paper prototyping [17, 36] to sophisticated electronic toolkits [8, 2]. For gesture recognition on physical objects, there is a significant barrier to entry regarding the time to design and develop such interfaces. There is also a lack of transferability across application contexts as tangible devices are often designed for bespoke tasks. Our work aims to provide a scalable and modular toolkit, both in hardware and software, for prototyping surface-based gestures on tangibles objects deployable to a variety of different application contexts.

A common evaluation approach for understanding the feasibility and generalisability of toolkits is to actually use the toolkit to design and develop a number of demonstrative applications [13, 18, 20]. We adopt this approach for *TangibleTouch* and develop 3 demonstrative applications using the toolkit.

INTERACTION SPACE

As a starting point, we explore the potential of a cubic tangible as an interaction device to highlight the interactions it can afford, as well as illustrate the challenges in detecting these interactions (see Figure 2). The space is divided into surface-based input, combining gestures, interactions beyond gesture, and output space.

1. Surface-based Input

The design space of single and multi-touch surface gestures is based on cubic tangible and surface-based computing literature [19, 34]. The surface interactions described in Figure 2, while not an exhaustive list, demonstrates the range of touch gestures that can be performed on a simple cube: single touch gestures such as taps, swipes, and path gestures, and multi-touch gestures such as multi-finger taps and pinch gestures. Due to the stable form factor of cubes and the inherent graspability, these gestures can be performed in hand or while the cube is at rest. By combining cube affordance and simple surface-based gestures, common TUI metaphors can be mimicked: swipes to represent a slider, taps as button input, path traces as directional input or as a dial, and pinch gestures to replicate common touchpad input.

2. Combining Gestures for Input

Cubes are multiplexers of interaction by virtue of their form factor. Each face, while appearing identical to one another, can be leveraged as a separate area for input and even configured on the fly depending on the context of interaction. In the case of surface-based gestures a cube can support simultaneous and heterogeneous surface-based gestures on any face. Further to this, a cubic tangible can support modal or state-based interactions depending on whether the cube is on a surface, in hand, or actively touched on a particular area.

3. Other Interactions

There are multitudes of other interactions that are significant for the design space of cubic tangible interaction. Based on the work of Lefeuvre et al [19], cube affordance alone has interesting properties such as manipulability, spatial arrangement, and multi-functionality. Additional interactions are afforded when you instrument an object using a particular sensing approach. In our work, capacitive sensing enables the detection of proxemic interactions such as non-surface-based gestures, or even other capacitive devices.

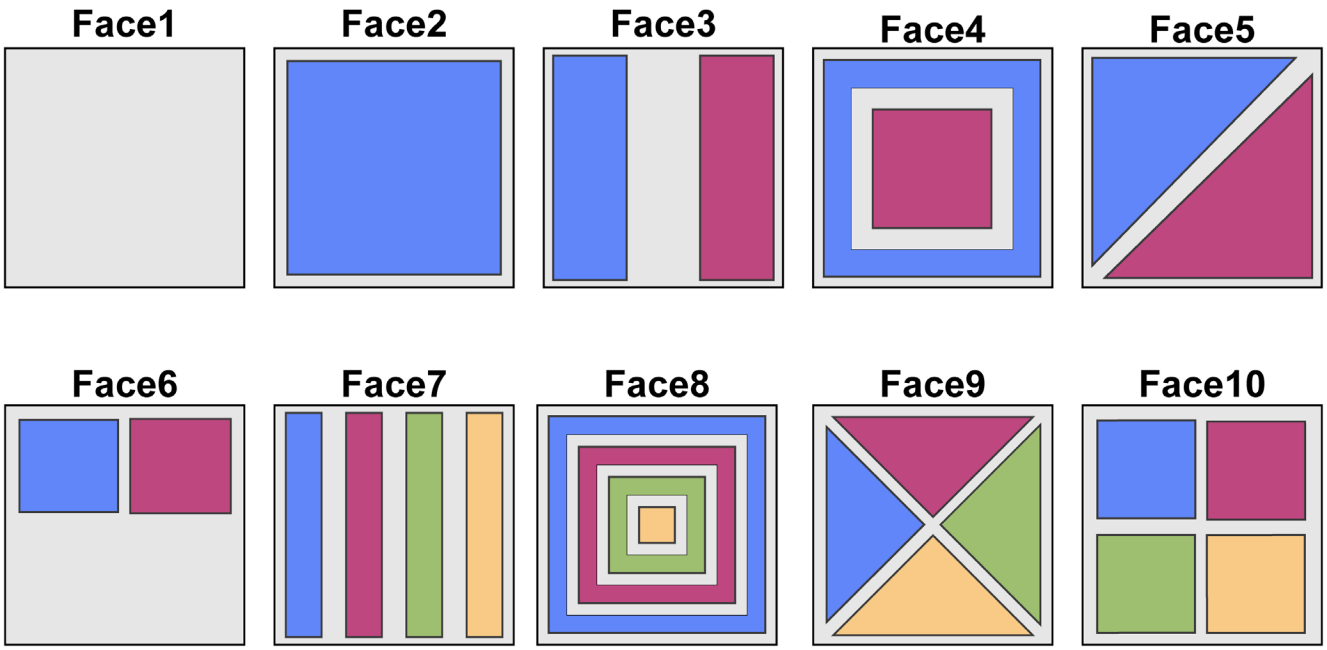
4. Output Space

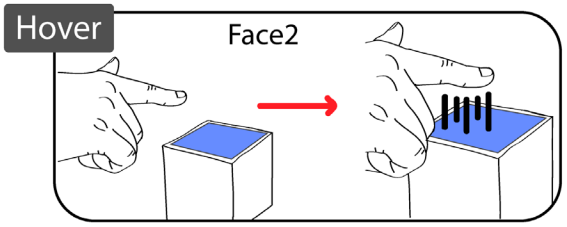
Depending on the interaction context, different advantages of cube affordance can be leveraged in addition to surface-based interactions. For example, a cube's dimensionality, manipulability, and ability to be a pedestal for output make them ideal candidates as interaction devices in Mixed Reality applications. Combine these inherent advantages with gesture detection and a wide variety of unique interactions can be supported. Moreover, the inherent tangibility of cubes makes them an ideal interaction device to support collaboration, having the ability to be freely passed from one user to another or placed in the environment. Cubes are also suited in output spaces that leverage physical surfaces, such as desktops or interactive surfaces. The stability and space afforded by a surface not only allows users to easily arrange and configure cubic tangibles, but also combine surface-gestures simultaneously.

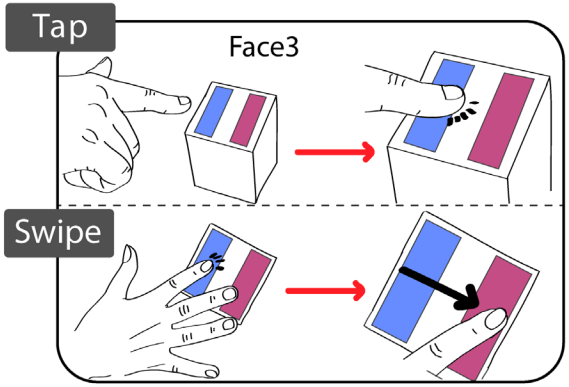
A rich design space for cubic interaction devices exists across a number of different output spaces (see Figure 2). However, supporting a wide array of surface-based gestures without adorning the user or using external tracking is complex to fabricate and often lacks scalability. Gesture recognition using capacitive sensing is also non-trivial and requires different sensing configurations depending on the surface-gesture being detected. To address this we developed the *TangibleTouch* toolkit that provides a method for fabricating extended and modular capacitive cubes to detect distinct on-surface gestures.

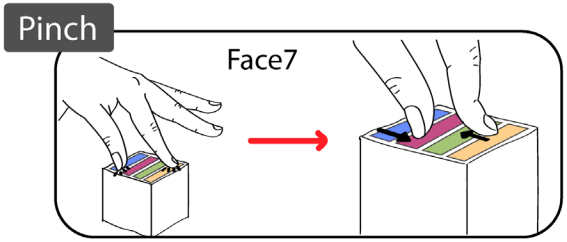
TOOLKIT DESIGN

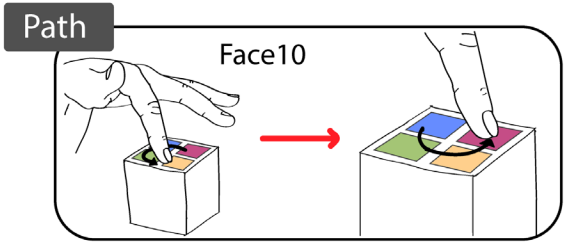
To address key challenges that exist in fabricating interactive tangible cubes, we developed *TangibleTouch*. The goal of this toolkit is to provide a **rapid fabrication** method that uses simple single-extrusion 3D printing to support the prototyping of **modular** cubic tangibles with **interchangeable capacitive faces** with different sensor configurations. Additionally, the toolkit aims to provide a **software platform** for digitally configuring the faces of a cube, training a particular face to detect one or more surface-gestures using machine learning, and a means of deploying those interactions in a variety of interaction contexts and applications. The toolkit consists of 3 parts: i) A **face design space** for divvying up the surface area of a cube to design for single and multi-touch gesture recognition using capacitive sensing. ii) An extendable **hardware platform** using conventional 3D printing that allows for interchangeable faces with different capacitive configurations. iii) A **software platform** that provides a user interface to add and configure interactive faces to a cube, record data of surface gestures and train a machine-learning model for gesture detection, and deploy designed interactions into a variety of different Unity based applications.











		Face1	Face2	Face3	Face4	Face5	Face6	Face7	Face8	Face9	Face10
Attributes	Sensor Number	0	1	2	2	2	2	4	4	4	4
	Configuration	Blank	Square	Rectangle	Radial	Cross	Square	Rectangle	Radial	Cross	Square
	Face Coverage	0%	100%	66%	66%	66%	66%	80%	80%	80%	80%
Interaction	Tap	X	√	√	√	√	√	√	√	√	√
	Pinch	X	X	X	X	X	X	√	X	X	X
	Swipe	X	X	√	√	√	√	√	√	√	√
	Path	X	X	X	X	X	X	√	X	√	√
	Hover	X	√	√	√	√	√	√	√	√	√

Figure 3. Capacitive face design space showing the configuration of sensors and the touch interactions afforded.

Face Design

To guide the design of the capacitive faces, we started with five general interactions that we wanted to detect using the lowest number of touch-sensitive areas: tap, swipe, pinch, path, and hover (see Figure 2). Figure 3 shows a number of different sensor configurations, varied by the number of interactive areas and their placement on a given face, followed by which surface-gestures these can support. The concept for each face design follows a principle of low complexity in terms of number of discrete touch areas. Instead, we rely on the multiplexed nature of a cube, with dedicated faces for particular interactions. While the toolkit can support more complex face designs, we focus on 10 simple face designs with 4 different sensor quantities (0, 1, 2, and 4), and 4 different face configurations: *square*, *radial*, *rectangle*, and *cross*. Figure 3 demonstrates how these variables determine if a surface gesture can be supported. For example, *face2* supports a tap but not a swipe as opposed to *face3* which supports a tap or swipe but requires double the amount of touch-areas. For detecting a swipe gesture in practice, *face3's* sensor0 is triggered at the start of a swipe gesture and sensor1 at the end and vice versa for a different swipe direction. Generally, if more complex gestures are to be detected the number of touch areas on a single face increases, or if a face needs to support more than one surface gesture. Another example is that both *face3* and *face5* support swipes, but the directionality of the swipe relative to the rest of the cube would be different, i.e. *face3* cannot support 'top-to-bottom' swipes whereas *face5* can.

Hardware Platform and Fabrication

The hardware platform for *TangibleTouch* (see Figure 5) consists of three main components all of which can be fabricated using a conventional, single-extrusion 3D printer: i) non-conductive face bases, ii) conductive face components, and iii) cube chassis. In the face design, we explicitly chose to design the conductive and non-conductive parts to be printed separately to make fabrication more viable for single-extrusion printers, which are generally more accessible and commercially available, opposed to dual-extrusion printers. We also use a capacitive sensor board, Arduino microcontroller, and lithium battery to instrument the cube.

Non-conductive face bases and cube chassis can be printed using generic PLA or ABS. The conductive components can be printed using either conductive PLA or ABS with conductance of at least 4.6*102 Ohms/cm. For the tangible cube prototypes we developed using the toolkit, the non-conductive parts were printed using white Filamentive PLA and the conductive parts were printed using U3 conductive ABS with a conductance of 4.64*102 Ohms/cm. We recommend ABS for the conductive components as acetone can be used as a means of adhering the conductive pieces to the non-conductive base plate giving reliable adhesion with less impact on the surface capacitance. Conductive PLA can be used, and can often provide better conductivity, but an adhesive agent is needed to mount conductive components to the non-conductive face base plate, which may affect the surface capacitance.

All 3D printed parts were printed on an Ender3 V2 at 50mm/s and a layer height of 0.16mm. The cube chassis took 8 hours to print and each face base plate took 30 minutes (11 hours total for non-conductive parts). Conductive part print times can vary depending on the surface coverage, from 30 minutes to 1 hour.

Dupont cables were used to connect the conductive parts to the capacitive sensor board, by heating a male connector using a soldering iron and inserting it into the mounting points. The prototype conductive components were measured at around 30kohm resistance across the conductive surface, and 10kohm from the conductive surface to the connecting cable. We also tested ProtoPasta conductive PLA mounted to the base plate using hot glue, which measured at 6kohm resistance across the conductive surface and 4kohm from the surface to the connecting dupont cable.

Once conductive components are mounted to the face base plate and the cables have been connected to the conductive mounting points, the face can be simply

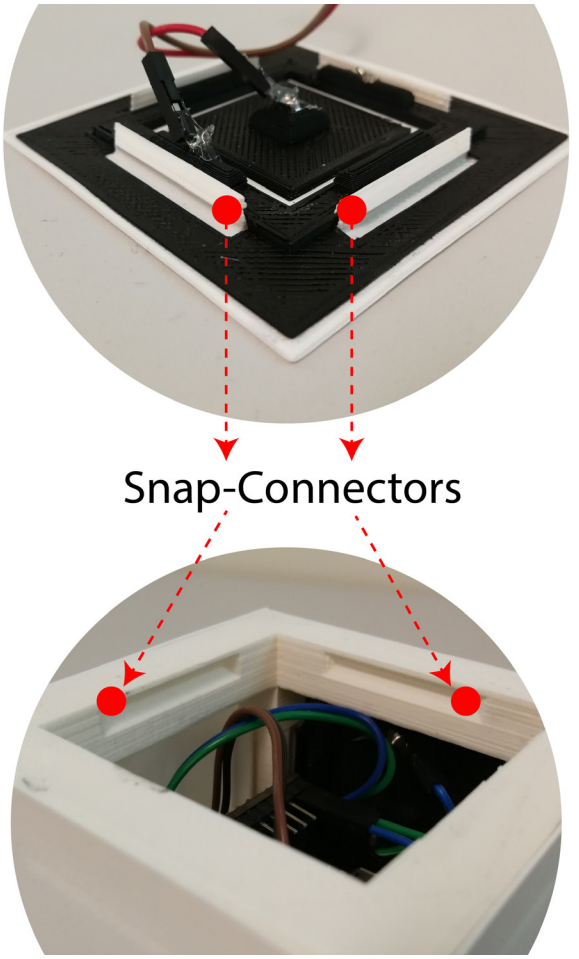


Figure 4. The snap connectors to mount capacitive faces.

attached to the cube chassis using 'snap-fit' connectors and the cables routed to the 12-channel capacitive sensor board (see Figure 4). In this case, a single cube can have a maximum of 12 discrete touch areas and, using the modular faces, distributed in any manner across the cube. For example, 2 faces with 4 touch areas, 2 faces with 2, and 2 with 0 or 6 faces with 2 touch areas.

We used an Adafruit MPR121 12-Key Capacitive Touch Sensor breakout board for detecting capacitance,

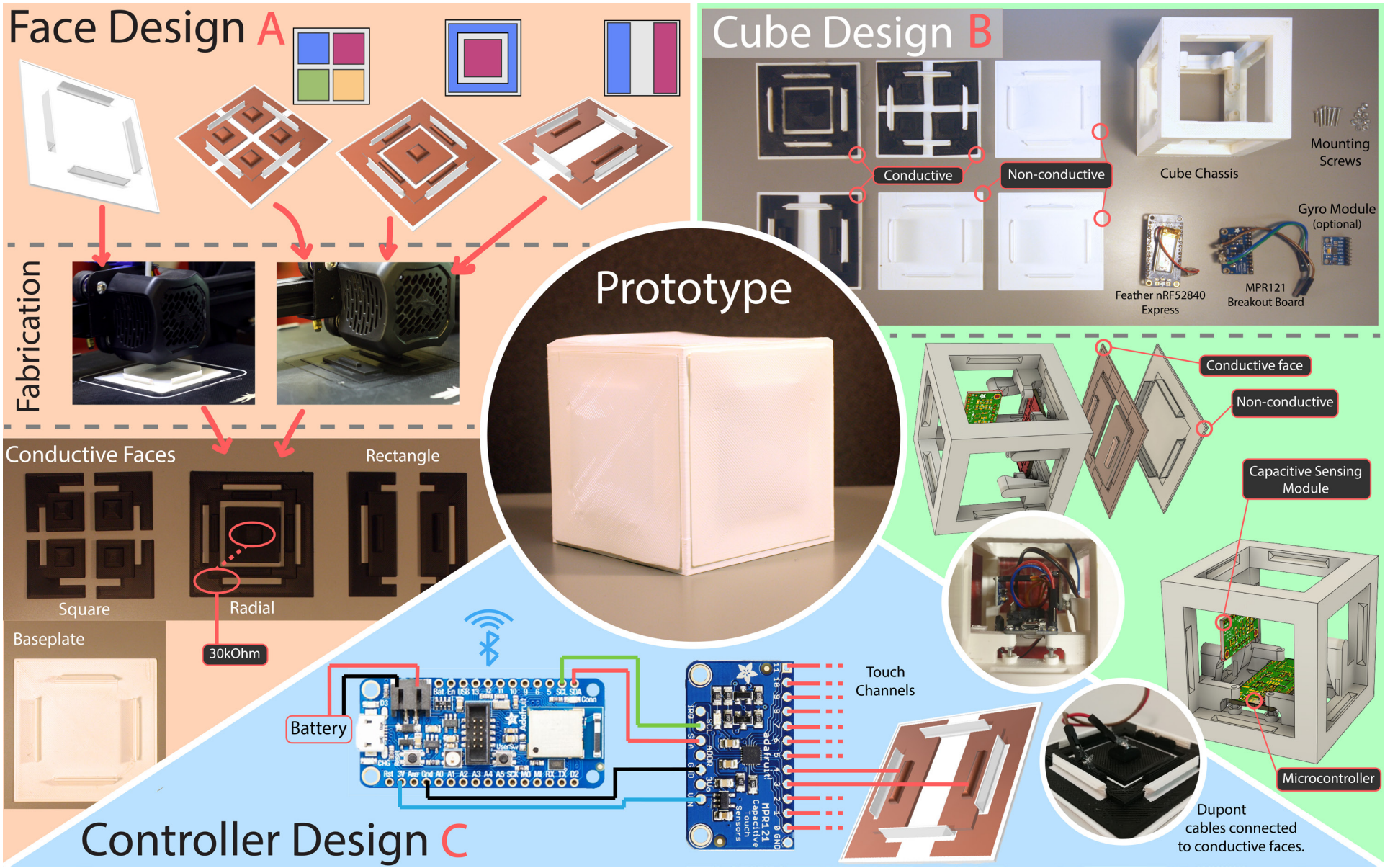


Figure 5. The toolkit's hardware components and design. **A**) The face design and fabrication, **B**) the cube components and modular design, and **C**) the controller design, and connection to the conductive faces.

connected to an Adafruit Feather nRF52840 Express microcontroller powered by a small 3.7v 110mah lithium polymer battery, all of which can be mounted inside the cube chassis using M2 screws and nuts. The Adafruit Feather has low energy Bluetooth capabilities for transmitting data to other devices, which using the 110mah battery can run for 11 hours on a single charge.

We experimented with an additional gyroscope module that also has space for mounting within the cube chassis. The purpose of this is to make configuring the cube using the software interface easier, as a designer can determine which faces map to the digital representation by simply tilting the cube. However, this is optional and is not necessary for configuring the cube.

Software Platform

The *TangibleTouch* software platform is used to digitally configure the cube with the appropriate interactive faces, train a particular surface gesture for a given face using machine learning, and deploy the trained model of a surface gesture to an application (see Figure 6). The software platform consists of three modules: i) An *Interface* library, ii) a *gesture-training* library, and iii) a *data processing* and hardware library.

TT Interface and Configuration

The Unity-based interface can be loaded as a scene in a developer's application to then configure a cube, train gestures, and deploy interactions packaged as Unity events, which applications can subscribe to. On loading the interface, a designer scans for Bluetooth devices or selects the cube directly if connected via UART. Once the cube is found and selected, the designer is taken to the configuration screen. Here, a designer can see the virtual representation of the cube device with 6 blank faces and a side panel with faces of varying sensor configurations. If a gyroscope module is connected to the cube, then the virtual cube will mimic the rotation of the physical cube to decipher the face positions, otherwise a developer can manipulate the cube using the mouse. Developers can easily add additional face configurations by invoking the *face* class and providing an '.obj' file.

To first configure the cube, a developer needs to select each face on the virtual model and assign a face configuration. Once all face configurations have been assigned the designer then cycles through each interactive touch area on the virtual model and touches the corresponding capacitive areas on the physical cube to calibrate the sensor channels on the capacitive sensor to the face configurations. If a previous configuration has been made, then a designer can load this into Unity from file by selecting the 'Load Previous' button. A cube configuration is cleared by clicking the 'Reset Cube' button. Once a designer is happy with their cube configuration, they can move to the gesture-training screen by clicking 'Continue'.

A designer adds a new gesture to the cube by uniquely naming the gesture and selecting 'Add New Gesture'. An optional keyboard binding can be added for that gesture that will be triggered on gesture detection. The designer then selects the newly added gesture and selects 'Record Gesture'. Now the designer performs the desired gesture 20 times, each with a sample size of 200 over a 3 second window. The number of gesture samples, the sample size, and duration can all be altered, but these were the most optimal settings considering the time to set up and accuracy. Once any number of new gestures are recorded, the model can be retrained to include the newly added gestures by selecting 'Train Model'. 'Start Detection' then deploys the trained model and firing Unity events or triggers keyboard input depending on whether any gestures are detected. Developers can have an external class subscribe to the events fired by the gesture detection class, with each event containing the unique gesture name. Figure 6 shows an example of recording data for a swipe gesture and then deploying this interaction to a simple slider.

TT Data Processing and Gesture Training

For the capacitive sensor board, the default firmware settings were sufficient for the most part, but after testing it was found that the non-conductive base plate covering the conductive areas causes the baseline signal to not adjust quickly enough when a sensor is touched, which affected the performance of gesture detection. Setting the filter delay register (MPR121_FDLF) to the maximum value of 255 greatly improved the touch sensitivity and baseline stability.

Gesture detection was implemented using TensorFlow[2], a recurrent neural network, incorporated into the Unity environment using a standalone Python program. Once the model is ready to be trained, recorded data is loaded from a CSV file, and any non-configured sensors are zeroed. To account for a user holding the cube or the cube resting on a surface while performing a gesture, any sensor channels that are triggered for more than 40% of a gesture sample are disregarded and zeroed. The model itself consists of 5 layers.

First, the data goes through feature extraction, consisting of maximum and minimum pool layers. These are typically used to down-sample and extract features from images by partitioning them into a set of non-overlapping rectangles and, for each such sub-region, outputs the maximum/average or minimum. In this case, max-pooling is used to increase the sequence length to smooth out any anomalous samples. As max-pooling decreases the number of lows, i.e. timesteps where the sensors are not touched, between touches, min-pooling had to be performed to increase these gaps between touches. Setting the strides to 2 also down-sampled the data from 200 timesteps to 96. Pool sizes were selected by experimentation.

The main processing is done via the Gated recurrent unit (GRU) layer. We chose GRU as related work shows better temporal performance while maintaining equivalent accuracy [6]. Best accuracy was achieved with a unit count of 3 * number of labels. A Gaussian noise layer was added to prevent overfitting and an RMSprop optimizer was used, with a learning rate of 0.02. Nadam and Adam optimizers were also tested, but they were slightly worse in terms of prediction accuracy. Learning rates up to 0.03 can be used to improve the speed at the cost of less stable changes between epochs.

2 https://www.tensorflow.org/

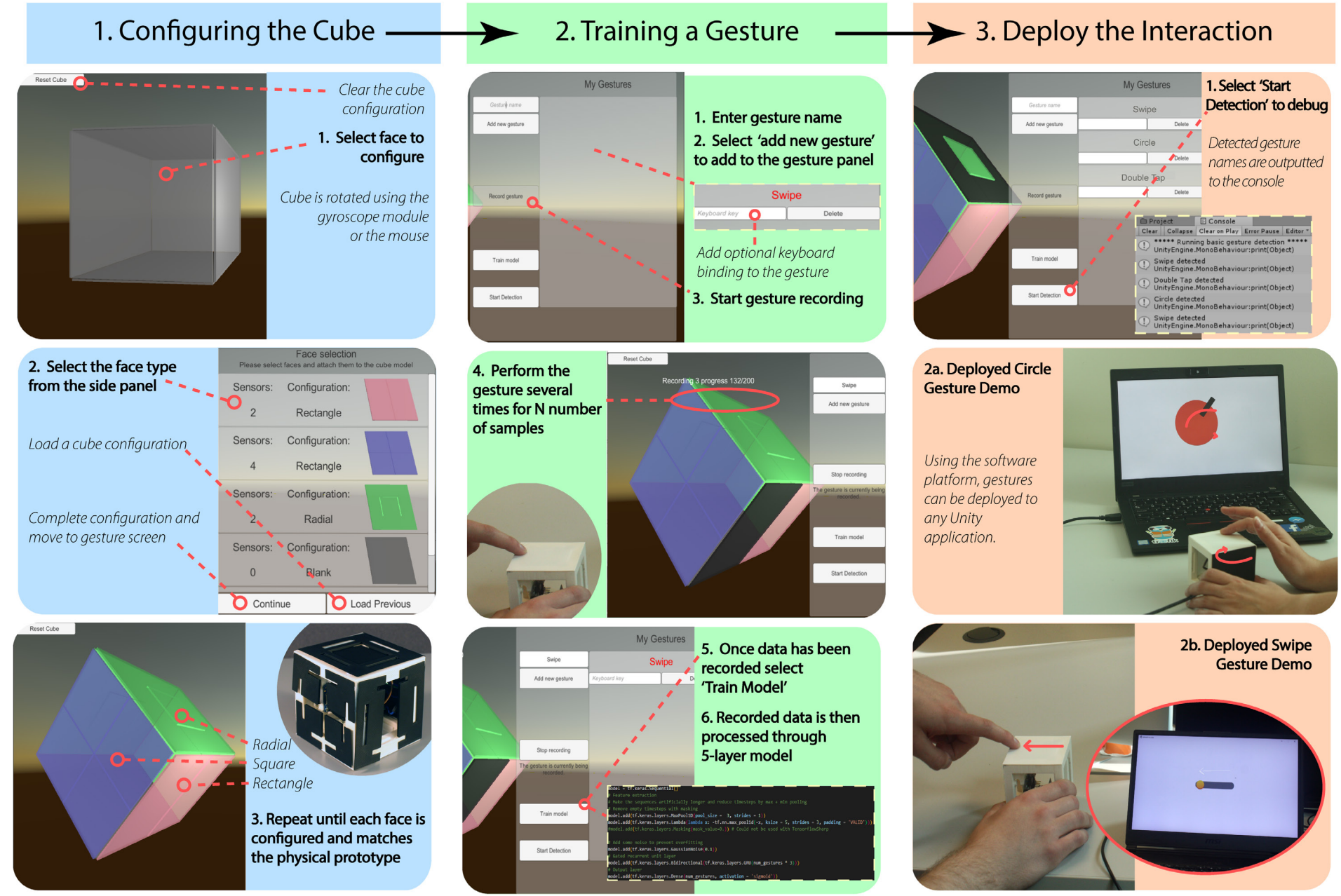


Figure 6. The software platform to configure, design, train, and test surface gestures including some examples of deployable interactions.

A learning rate schedule was used to decrease the learning rate over time.

Categorical cross-entropy was used as the loss function and Softmax was used as the activation for the output layer. The model is trained over 50 epochs with a batch size of 32. The batch size can be increased to increase speed, but it will reduce the maximum accuracy that could be achieved and the model will converge slower, requiring more epochs. The epoch count can also be decreased, at the cost of stability. An average accuracy of 93% was recorded, with the validation and training producing similar accuracy. The trained gesture is then loaded as a frozen graph.

For live gesture detection, the capacitive data is filtered and then sent to the model by using TensorFlowSharp[3]. TensorFlowSharp is a runtime that allows for TensorFlow models to run from C# and therefore in Unity. Two 100 sample rolling-windows were used for continuous detection, one with 100 latest samples and the other with 100 samples from the previous window. Once a gesture is detected, a Unity event is fired with the corresponding gesture name. Finally, the toolkit source code and 3D-models of the hardware components are entirely open source[4].

Gesture Detection Accuracy

To test the gesture detection accuracy, we conducted a small preliminary study involving 4 participants. Participants would test 3 pre-trained gestures, *double-tap finger*, *double-tap hand*, and *finger swipe*, and 1 *custom gesture* they create and trained themselves (for a total of 40 recordings). The entire study was done using the 2-sensor radial face configuration. Each of the 4 gestures was tested 20 times by each participant. The detection accuracy for *double-tap finger* was **93%**, *double-tap hand* was **100%**, *finger swipe* was **85%**, and the *custom gesture* was **85%**.

3 https://github.com/migueldeicaza/TensorFlowSharp
4 https://github.com/TangibleTouch/Toolkit

APPLICATIONS

To evaluate the functionality of *TangibleTouch,* we employ a Type 1 evaluation strategy [18] to demonstrate the feasibility of the toolkit and its ability to rapidly prototype a tangible interface. We created three exemplar applications developed in Unity and deployed across 3 different output spaces: *Model Inspector* (Mixed Reality), *2D Platformer* (Desktop), and a *Media Player* (Public display). Figure 7, 8, and 9 show the generative breadth of the *TangibleTouch* toolkit, illustrating different touch gestures used in each application.

Application 1: Model Inspector

This application allows the cube to manipulate a 3D model, loaded into Unity, via rotation and scaling. The cube designed for this application uses 4 blank faces and 2 interactive faces: a 4-sensor square, and a 2-sensor rectangle. The model inspector was deployed in Mixed Reality, a popular medium for Tangible Interaction, using a Microsoft Hololens2 as shown in Figure 7. A 3D model is rotated by performing a circular path gesture on the cubes 4-sensor square face, with the direction of rotation mapped to the direction of the gesture performed. The user can cycle through the 3 different axes for rotation, roll, pitch, and yaw, by tapping either of the two sensors on the rectangle face. Object scaling is performed by swiping from one sensor on the rectangle face to the other, and the direction of the swipe determines whether the object grows or shrinks. The Model Inspector allows users to manipulate and separate the rotational degrees of freedom of a virtual model over distance.

Application 2: 2D Platformer

This application demonstrates a simple platformer game controlling a 2D character to jump, move, draw, and release an arrow. The cube uses 2 interactive faces and 4 blank faces: a 4-sensor cross, and a 2-sensor radial. The application was deployed to a desktop PC shown in Figure 8. In this application, we make use of the key mapping function in the toolkit software to map touch input on the cube's 4-sensor square face to key bindings in Unity, W, A, S, and D, for character movement. The user can perform an additional action of drawing the

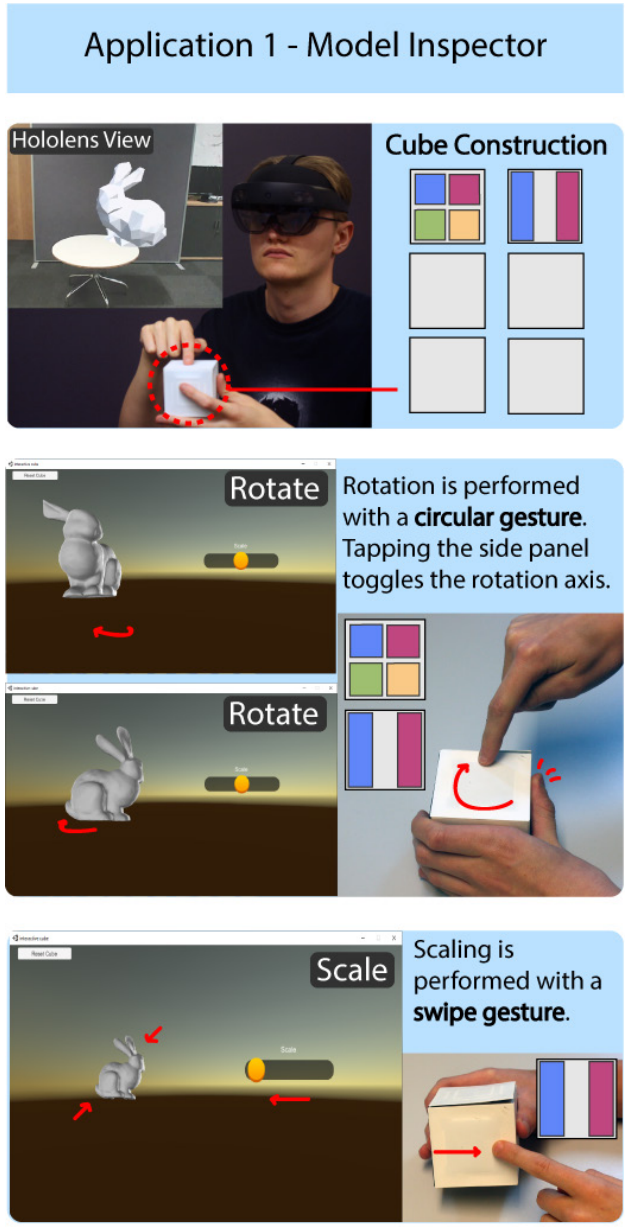


Figure 7. Model Inspector application in mixed reality.

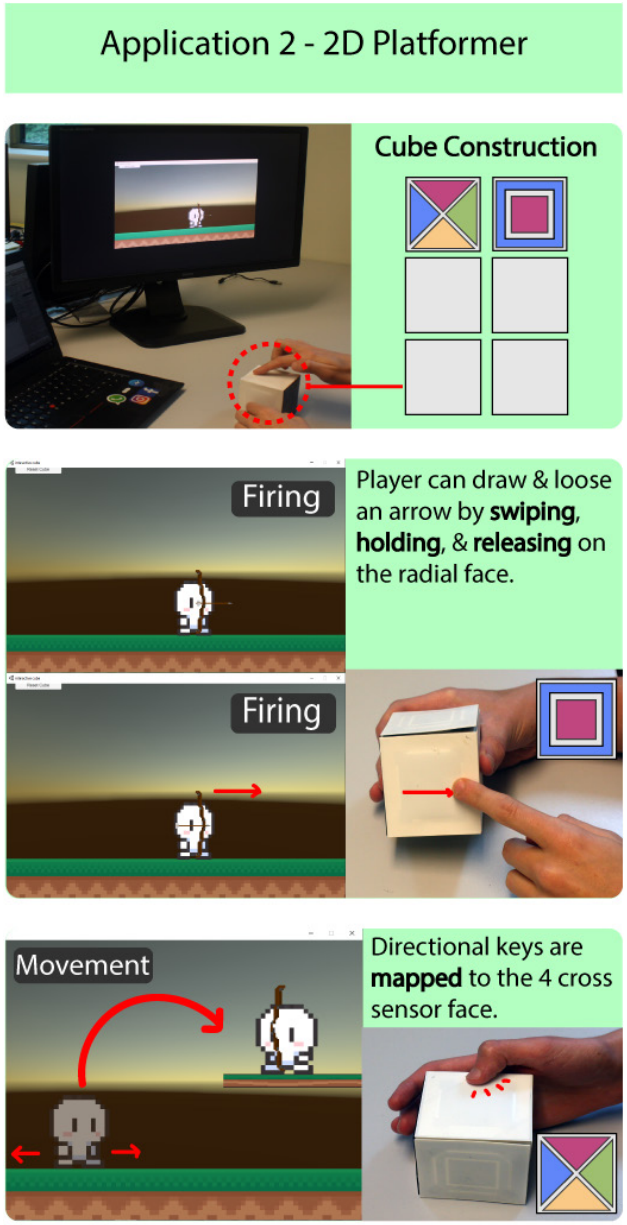


Figure 8. 2D Platformer application on a desktop PC.

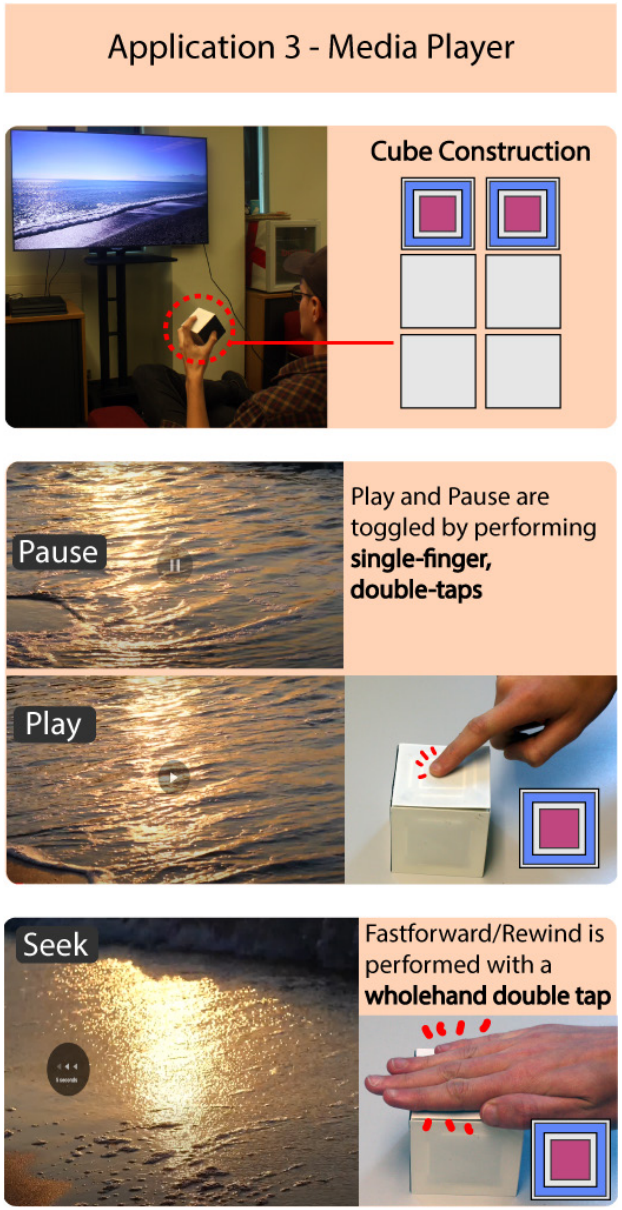


Figure 9. Media Player application on a public display.

bow by swiping in any direction on the 2-sensor radial face and then firing the bow by releasing the finger from the sensor.

Application 3: Media Player

The final application demonstrates a simple media controller for playing, pausing, forwarding, and rewinding a video deployed on a public display. The cube designed for this application makes use of 4 blank faces and 2 interactive faces: two 2-sensor radials. This application also makes use of the key mapping function to work with web-based video players. As shown in Figure 9, a user can double-tap the centre of one radial face with a single finger to toggle pause and play, and perform a whole-hand double tap to fast forward. The same whole-hand gesture is performed on the other radial face to rewind a video.

DISCUSSION

Prototyping tangible objects with touch capabilities is complex, and designing for intricate surface-based gestures is non-trivial. Our toolkit, *TangibleTouch*, addresses these challenges by providing an extendable and modular hardware platform, that leverages cube affordance, and a software platform for configuring and designing tangible gesture interfaces across many different output spaces. Designers can: i) design their own bespoke sensor configurations for a cube, ii) create, train, and test in real-time a machine-learning model for a set of surface-gestures using the provided interface components, and iii) effortlessly deploy interactions to a variety of applications built in Unity.

Expert designers can build upon the capacitive face design space introduced, as well as the hardware platform, to build entirely new face configurations for different or more complex surface gestures. The *TangibleTouch* interface and cube configuration software can support any number of face designs, and gestures can be trained with any sensor configuration. To ensure accessible fabrication, we designed the toolkit components to be produced entirely using single extrusion 3D printing.

Using three demonstrative applications that include a number of different sensor configurations and surface-based gestures, we highlight the generality of interactions supported as well as the versatility of deployment in different output spaces. Additionally, we show the toolkit's ability to rapidly prototype and explore rich and expressive input in tangible interaction. A fundamental limitation, but also a distinct advantage, is the use of a cubic form factor. The abstract nature of cubes means that any designed gestures can often be transferrable to other, more complex tangible form factors. Furthermore, designers can multiplex surface gestures in a single artefact by simply using the discrete faces inherent to the cubic form factor. Additionally, the modular nature of the cube allows for on-the-fly reconfiguration of the interaction device, ideal for exploring a breadth of interactions during prototyping stages of a tangible interface.

Building upon *TangibleTouch*

There is a clear avenue for future work building on the foundational elements of *TangibleTouch*. Firstly, to develop more face designs in terms of capacitive sensor configurations but also explore face surface texture, form, and colour. By using the same modularity principles, there is potential to not only design and explore further surface-gestures, but also explore haptic experiences and output. For example, to differentiate and communicate gesture mappings to faces or better convey desired interactions to novice users through the materiality and affordance of a face. To achieve this, more sophisticated fabrication approaches could be incorporated to expand the toolkits design space further.

In terms of leveraging cube affordance in tangible interaction, our toolkit has scratched the surface. Previous work has touched on the benefits of cube affordance in interaction [19], and the *TangibleTouch* toolkit could be expanded to explore cubes manipulability, spatial stability and arrangement, and capacity as a pedestal for output. The highly decoupled nature of the toolkit enables the surface-based gesture detection to be incorporated with additional sensing approaches or devices such as inertia-measurement, proximity sensors, and visual displays. Furthermore, our initial characterisation of the interaction space briefly touched on the implications of multiple cubes. The interplay between multiple interactive cubes and their utility in collaborative tasks warrants exploration in of itself. The cube form factor could be condensed, by adjusting the level of instrumentation, opening up a design space for 100s of stackable, miniaturised cubes that can be configured into new and unique geometries. Finally, while we have evaluated *TangibleTouch* through demonstrative applications [18], future work could employ different evaluation methodologies such as case-studies, a usability study, or heurisitic evaluation.

CONCLUSION

Through our toolkit, *TangibleTouch*, designers can prototype and develop bespoke surface-based gestures for tangible interfaces using a modular and easily fabricated hardware platform and a software framework that abstracts away complex data processing and machine learning. The toolkit also reduces testing complexity using a run-time environment to detect designed gestures in real-time. Both the hardware and software platforms are highly decoupled and extendable for expert designers to create other capacitive face designs, incorporate additional sensors, and implement into any Unity-based applications. Future work includes building on the basis the toolkit has provided to explore: cube affordance beyond surface-based gestures, supporting multi-cube interactions, and different types of sensors and instrumentation.

REFERENCES

[1] Leonardo Angelini, Denis Lalanne, Elise Van den Hoven, Khaled Omar Abou, Elena Mugellini. 2015. Move, Hold and Touch: A Framework for Tangible Gesture Interactive Systems. *Machines*, 3(3), 173-207. https://doi.org/10.3390/machines3030173

[2] Jatin Arora, Aryan Saini, Nirmita Mehra, Varnit Jain, Shwetank Shrey, and Aman Parnami. 2019. VirtualBricks: Exploring a Scalable, Modular Toolkit for Enabling Physical Manipulation in VR. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Paper 56, 1–12. https://doi.org/10.1145/3290605.3300286

[3] Daniel Avrahami, Jacob O. Wobbrock, and Shahram Izadi. 2011. Portico: tangible interaction on and around a tablet. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (*UIST '11*). Association for Computing Machinery, New York, NY, USA, 347–356. https://doi.org/10.1145/2047196.2047241

[4] Till Ballendat, Nicolai Marquardt, and Saul Greenberg. 2010. Proxemic interaction: designing for a proximity and orientation-aware environment. In *ACM International Conference on Interactive Tabletops and Surfaces* (*ITS '10*). Association for Computing Machinery, New York, NY, USA, 121–130. https://doi.org/10.1145/1936652.1936676

[5] Jesse Burstyn, Nicholas Fellion, Paul Strohmeier, Roel Vertegaal. 2015. PrintPut: Resistive and Capacitive Input Widgets for Interactive 3D Prints. *15th Human-Computer Interaction* (*INTERACT*), Sep 2015, Bamberg, Germany. pp.332-339, https://doi.org/10.1007/978-3-319-22701-6_25

[6] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS 2014 Workshop on Deep Learning*, December 2014. https://arxiv.org/abs/1412.3555

[7] David Englmeier, Julia Dörner, Andreas Butz and Tobias Höllerer, 2020 A Tangible Spherical Proxy for Object Manipulation in Augmented Reality. *2020 IEEE Conference on Virtual Reality and 3D User Interfaces* (*VR*), 2020, pp. 221-229, https://doi.org/10.1109/VR46266.2020.00041

[8] Martin Feick, Scott Bateman, Anthony Tang, André

Miede and Nicolai Marquardt. 2020 Tangi: Tangible Proxies For Embodied Object Exploration And Manipulation In Virtual Reality. *2020 IEEE International Symposium on Mixed and Augmented Reality* (*ISMAR*), 2020, pp. 195-206, https://doi.org/10.1109/ISMAR50242.2020.00042

[9] Kenneth P. Fishkin. 2004. A taxonomy for and analysis of tangible interfaces. *Personal Ubiquitous Computing 8*, 5 (September 2004), 347–358. https://doi.org/10.1007/s00779-004-0297-4

[10] Tobias Grosse-Puppendahl, Yannick Berghoefer, Andreas Braun, Raphael Wimmer and Arjan Kuijper, 2013. OpenCapSense: A rapid prototyping toolkit for pervasive interaction using capacitive sensing. 2*013 IEEE International Conference on Pervasive Computing and Communications* (*PerCom*), pp. 152-159, https://doi.org/10.1109/PerCom.2013.6526726

[11] Tobias Grosse-Puppendahl, Christian Holz, Gabe Cohn, Raphael Wimmer, Oskar Bechtold, Steve Hodges, Matthew S. Reynolds, and Joshua R. Smith. 2017. Finding Common Ground: A Survey of Capacitive Sensing in Human-Computer Interaction. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (*CHI '17*). Association for Computing Machinery, New York, NY, USA, 3293–3315. https://doi.org/10.1145/3025453.3025808

[12] David Holman and Hrvoje Benko. 2011. SketchSpace: designing interactive behaviors with passive materials. In *CHI '11 Extended Abstracts on Human Factors in Computing Systems* (*CHI EA '11*). Association for Computing Machinery, New York, NY, USA, 1987–1992. https://doi.org/10.1145/1979742.1979867

[13] Steven Houben and Nicolai Marquardt. 2015. WatchConnect: A Toolkit for Prototyping Smartwatch-Centric Cross-Device Applications. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (*CHI '15*). Association for Computing Machinery, New York, NY, USA, 1247–1256. https://doi.org/10.1145/2702123.2702215

[14] Elise Van den Hoven and Ali Mazalek. 2011. Grasping gestures: Gesturing with physical artifacts. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 25, no. 3, pp. 255–271, 2011. https://doi.org/10.1017/S0890060411000072

[15] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. 2007. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction* (*TEI '07*). Association for Computing Machinery, New York, NY, USA, 139–146. https://doi.org/10.1145/1226969.1226998

[16] Martin Kaltenbrunner and Ross Bencina. 2007. ReacTIVision: a computer-vision framework for table-based tangible interaction. *In Proceedings of the 1st international conference on Tangible and embedded interaction* (*TEI '07*). Association for Computing Machinery, New York, NY, USA, 69–74. https://doi.org/10.1145/1226969.1226983

[17] Annie Kelly, R. Benjamin Shapiro, Jonathan de Halleux, and Thomas Ball. 2018. ARcadia: A Rapid Prototyping Platform for Real-time Tangible Interfaces. *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (*CHI'18*). Association for Computing Machinery, New York, NY, USA, Paper 409, 1–8. https://doi.org/10.1145/3173574.3173983

[18] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. *Proceedings of the 2018 CHI Conference on Human Factors in Computing System*s (*CHI'18*). Association for Computing Machinery, New York, NY, USA, Paper 36, 1–17. https://doi.org/10.1145/3173574.3173610

[19] Kevin Lefeuvre, Soeren Totzauer, Michael Storz, Albrecht Kurze, Andreas Bischof, and Arne Berger. 2018. Bricks, Blocks, Boxes, Cubes, and Dice: On the Role of Cubic Shapes for the Design of Tangible Interactive Devices. In *Proceedings of the 2018 Designing Interactive Systems Conference* (*DIS '18*). Association for Computing Machinery, New York, NY, USA, 485–496. https://doi.org/10.1145/3196709.3196768

[20] Nicolai Marquardt, Robert Diaz-Marino, Sebastian Boring, and Saul Greenberg. 2011. The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. In *Proceedings of the 24th annual ACM symposium on User interface software and technology* (*UIST '11*). Association for Computing Machinery, New York, NY, USA, 315–326. https://doi.org/10.1145/2047196.2047238

[21] Thomas Muender, Anke V. Reinschluessel, Sean Drewes, Dirk Wenig, Tanja Döring, and Rainer Malaka. 2019. Does It Feel Real? Using Tangibles with Different Fidelities to Build and Explore Scenes in Virtual Reality. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (*CHI'19*). Association for Computing Machinery, New York, NY, USA, Paper 673, 1–12. https://doi.org/10.1145/3290605.3300903

[22] Alexander Nelson, Gurashish Singh, Ryan Robucci, Chintan Patel and Nilanjan Banerjee. 2015. Adaptive and Personalized Gesture Recognition Using Textile Capacitive Sensor Arrays. *IEEE Transactions on Multi-Scale Computing Systems*, vol. 1, no. 2, pp. 62-75, 1 April-June 2015, https://doi.org/10.1109/TMSCS.2015.2495100

[23] Simon Olberding, Nan-Wei Gong, John Tiab, Joseph A. Paradiso, and Jürgen Steimle. 2013. A cut-

table multi-touch sensor. In *Proceedings of the 26th annual ACM symposium on User interface software and technology* (*UIST '13*). Association for Computing Machinery, New York, NY, USA, 245–254. https://doi.org/10.1145/2501988.2502048

[24] Valkyrie Savage, Colin Chang, and Björn Hartmann. 2013. Sauron: embedded single-camera sensing of printed physical user interfaces. In *Proceedings of the 26th annual ACM symposium on User interface software and technology* (*UIST '13*). Association for Computing Machinery, New York, NY, USA, 447–456. https://doi.org/10.1145/2501988.2501992

[25] Valkyrie Savage, Xiaohan Zhang, and Björn Hartmann. 2012. Midas: fabricating custom capacitive touch sensors to prototype interactive objects. In *Proceedings of the 25th annual ACM symposium on User interface software and technology* (*UIST '12*). Association for Computing Machinery, New York, NY, USA, 579–588. https://doi.org/10.1145/2380116.2380189

[26] Martin Schmitz, Mohammadreza Khalilbeigi, Matthias Balwierz, Roman Lissermann, Max Mühlhäuser, and Jürgen Steimle. 2015. Capricate: A Fabrication Pipeline to Design and 3D Print Capacitive Touch Sensors for Interactive Objects. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology* (*UIST '15*). Association for Computing Machinery, New York, NY, USA, 253–258. https://doi.org/10.1145/2807442.2807503

[27] Orit Shaer and Eva Hornecker. 2010. Tangible User Interfaces: Past, Present, and Future Directions. *Found. Trends Hum.-Comput. Interact.* 3, 1–2 (January 2010), 1–137. https://doi.org/10.1561/1100000026

[28] Jennifer G. Sheridan, B. W. Short, Kristof Van Laerhoven, Nicolas Villar and Gerd Kortuem. 2003. Exploring cube affordance: towards a classification of non-verbal dynamics of physical interfaces for wearable computing. *IEE Eurowearable*, 2003, pp. 113-118, https://doi.org/10.1049/ic:20030156

[29] Gurashish Singh, Alexander Nelson, Ryan Robucci, Chintan Patel and Nilanjan Banerjee. 2015. Inviz: Low-power personalized gesture recognition using wearable textile capacitive sensor arrays. *2015 IEEE International Conference on Pervasive Computing and Communications* (*PerCom*), pp. 198-206, https://doi.org/10.1109/PERCOM.2015.7146529

[30] Christian Stetco, Stephan Mühlbacher-Karrer, Matteo Lucchi, Matthias Weyrer, Lisa-Marie Faller and Hubert Zangl. 2020 Gesture-based Contactless Control of Mobile Manipulators using Capacitive Sensing. *2020 IEEE International Instrumentation and Measurement Technology Conference* (*I2MTC*), pp. 1-6, https://doi.org/10.1109/I2MTC43012.2020.9128751

[31] Brygg Ullmer and Hiroshi Ishii. 2000. Emerging frameworks for tangible user interfaces. *IBM Syst. J.* 39, 3–4 (July 2000), 915–931. https://doi.org/10.1147/sj.393.0915

[32] Malte Weiss, Florian Schwarz, Simon Jakubowski, and Jan Borchers. 2010. Madgets: actuating widgets on interactive tabletops. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology* (*UIST '10*). Association for Computing Machinery, New York, NY, USA, 293–302. https://doi.org/10.1145/1866029.1866075

[33] Malte Weiss, Julie Wagner, Yvonne Jansen, Roger Jennings, Ramsin Khoshabeh, James D. Hollan, and Jan Borchers. 2009. SLAP widgets: bridging the gap between virtual and physical controls on tabletops. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (*CHI '09*). Association for Computing Machinery, New York, NY, USA, 481–490. https://doi.org/10.1145/1518701.1518779

[34] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-defined gestures for surface computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (*CHI '09*). Association for Computing Machinery, New York, NY, USA, 1083–1092. https://doi.org/10.1145/1518701.1518866

[35] Hui-Shyong Yeo, Ryosuke Minami, Kirill Rodriguez, George Shaker, and Aaron Quigley. 2018. Exploring Tangible Interactions with Radar Sensing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 4, Article 200 (December 2018), 25 pages. https://doi.org/10.1145/3287078

[36] Clement Zheng, Peter Gyory, and Ellen Yi-Luen Do. 2020. Tangible Interfaces with Printed Paper Markers. In *Proceedings of the 2020 ACM Designing Interactive Systems Conference* (*DIS '20*). Association for Computing Machinery, New York, NY, USA, 909–923. https://doi.org/10.1145/3357236.3395578

[37] Kening Zhu, Taizhou Chen, Feng Han, and Yi-Shiun Wu. 2019. HapTwist: Creating Interactive Haptic Proxies in Virtual Reality Using Low-cost Twistable Artefacts. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. Association for Computing Machinery, New York, NY, USA, Paper 693, 1–13. https://doi.org/10.1145/3290605.3300923