Capacitive Touch Sensing on General 3D Surfaces

GIANPAOLO PALMA, CNR - ISTI, Italy
NARGES POURJAFARIAN, Saarland University - Saarland Informatics Campus, Germany
JÜRGEN STEIMLE, Saarland University - Saarland Informatics Campus, Germany
PAOLO CIGNONI, CNR - ISTI, Italy

Fig. 1. Our computation fabrication method for multi-touch sensing on general 3D surfaces. (Left) 3D models with grooves to accommodate the touch sensor conductors and the internal pipes to connect the surface lines with the touch controller. (Center) Photo of the 3D printed prototypes equipped with the touch sensor grid made of enamelled unipolar solid copper conductor. (Right) An example of touch interaction with two fingers with the relative positions computed on the 3D model.

Mutual-capacitive sensing is the most common technology for detecting multi-touch, especially on flat and simple curvature surfaces. Its extension to a more complex shape is still challenging, as a uniform distribution of sensing electrodes is required for consistent touch sensitivity across the surface. To overcome this problem, we propose a method to adapt the sensor layout of common capacitive multi-touch sensors to more complex 3D surfaces, ensuring high-resolution, robust multi-touch detection. The method automatically computes a grid of transmitter and receiver electrodes with as regular distribution as possible over a general 3D shape. It starts with the computation of a proxy geometry by quad meshing used to place the electrodes through the dual-edge graph. It then arranges electrodes on the surface to minimize the number of touch controllers required for capacitive sensing and the number of input/output pins to connect the electrodes with the controllers. We reach these objectives using a new simplification and clustering algorithm for a regular quad-patch layout. The reduced patch layout is used to optimize the routing of all the structures (surface grooves and internal pipes) needed to host all electrodes on the surface and inside the object's volume, considering the geometric constraints of the 3D shape. Finally, we print the 3D object prototype ready to be equipped with the electrodes. We analyze the performance of the proposed quad layout simplification and clustering algorithm using different quad meshing and characterize the signal quality and accuracy of the capacitive touch sensor for different non-planar geometries. The tested prototypes show precise and robust multi-touch detection with good Signal-to-Noise Ratio and spatial accuracy of about 1mm.

CCS Concepts: • **Computing methodologies** → **Shape analysis**; • **Human-centered computing** → **Interaction devices**.

Additional Key Words and Phrases: capacitive touch sensing, geometry processing, interactive surface

ACM Reference Format:
Gianpaolo Palma, Narges Pourjafarian, Jürgen Steimle, and Paolo Cignoni. 2024. Capacitive Touch Sensing on General 3D Surfaces. *ACM Trans. Graph.* 43, 4, Article 103 (July 2024), 20 pages. https://doi.org/10.1145/3658185

1 Introduction

The rapid advancement of capacitive touch sensing technology has revolutionized how we interact with digital devices and interfaces. While capacitive touch sensing has been widely implemented on flat surfaces such as touchscreens, the growing demand for more intuitive and immersive user experiences has led to exploring touch sensing on 3D surfaces. Capacitive touch sensing on complex 3D surfaces offers new possibilities for interaction and expands the range of objects that can be transformed into interactive interfaces. However, extending high-resolution, multi-touch sensing to 3D

Authors' Contact Information: Gianpaolo Palma, gianpaolo.palma@isti.cnr.it, CNR - ISTI, Visual Computing Lab, Pisa, Italy; Narges Pourjafarian, pourjafarian@cs.uni-saarland.de, Saarland University - Saarland Informatics Campus, Human-Computer Interaction Lab, Saarbrücken, Germany; Jürgen Steimle, steimle@cs.uni-saarland.de, Saarland University - Saarland Informatics Campus, Human-Computer Interaction Lab, Saarbrücken, Germany; Paolo Cignoni, paolo.cignoni@isti.cnr.it, CNR - ISTI, Visual Computing Lab, Pisa, Italy.

ACM 1557-7368/2024/7-ART103
https://doi.org/10.1145/3658185

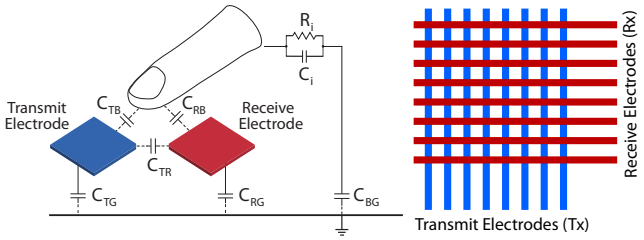


Fig. 2. (Left) Mutual capacitance touch sensing principle. (Right) Mutual capacitance sensor grid.

surfaces poses several significant challenges that must be addressed to ensure accurate and reliable touch detection. These challenges are related to the complex geometry and irregular surfaces of 3D objects, which can cause uneven electric field distribution, leading to inconsistent touch sensitivity across different areas of the object and affecting touch detection accuracy.

Capacitive Sensing Principles. Mutual capacitance sensing is the most common technology for achieving high-resolution, multi-touch detection on planar surfaces. This technique leverages the capacitive coupling effect, which occurs when two conductive objects (the transmit and receive electrodes) are positioned in close proximity. When an additional conductive object, such as the human body, approaches the electrodes, it establishes a capacitive coupling with the electrodes, causing a displacement of the current through the body to the ground (left Figure 2). By measuring the reduction in current at the receive electrode, the proximity of the body can be accurately determined [Barrett and Omote 2010; Grosse-Puppendahl et al. 2017].

A mutual capacitance touch sensor comprises two layers of conductors: the transmitting electrodes and sensing electrodes - conventionally called transmit electrodes (Tx) and receive electrodes (Rx). These layers are electrically insulated from each other by a dielectric material. The electrodes are arranged in a two-dimensional regular grid pattern, creating intersecting points. These intersections yield distinct touch-coordinate pairs, enabling the touch controller to measure each intersection independently. Our current implementation uses a rectangular electrode layout formed by overlapping straight lines (right Figure 2). Although this pattern is marginally less efficient than the more commonly used diamond pattern, it is easier to implement and fabricate. In mutual capacitance sensors, it is essential to maintain a uniform spacing between electrodes to achieve reliable and accurate touch detection. The spacing between the transmit and receive electrodes influences the capacitance coupling between them. If the spacing between the electrodes is not uniform, it can lead to variations in the coupling capacitance at different points on the sensor surface. These variations can result in inconsistent or inaccurate readings.

Contribution. This paper presents a new computational fabrication method to automatically arrange practical sensor grids on objects with a general shape, enabling multi-touch detection across their surfaces. The proposed method focuses on three design requirements of a mutual-capacitive line pattern sensor. The first requirement is to uniformly distribute the intersections of the sensing grid on the surface, aiming to ensure precise and accurate multi-touch detection. The second requirement is to minimize the number of capacitive touch controllers for sensing the entire surface, reducing the complexity and cost of the electronic hardware prototypes. The third requirement is to minimize the number of internal pipes to connect the sensor lines to the controllers to ensure a practical fabrication process. To meet these requirements, our approach proposes a new decomposition, simplification, and packing algorithm for the quad patch layout of a quad mesh that aims for efficient sensor placement.

Our input is the watertight triangle mesh of the object to be enriched with multi-touch sensing. We compute a proxy geometry using a quad meshing technique that preserves feature lines like borders and sharp creases [Pietroni et al. 2021]. The isometry and regularity properties of the quad meshing algorithm provide a uniform distribution of the electrodes' intersection points on the surface. Since a quad mesh contains singularities, we have to decompose it into a layout of patches containing only regular arrangements of quads that can be used for creating sensor grids. For this reason, we compute a quad patch layout with the motorcycle graph algorithm and further simplify it to get a coarse layout. Then, we pack these patches into the rectangular region that the capacitive touch controller can sense. This algorithm is designed to generate the minimum number of regular clusters of close patches, optimizing the continuity of the Tx-Rx field to reduce the number of controllers and input/output conductors. Using the packed layout, we generate the sensor grid mesh as the dual edge-graph of the quad mesh. This edge mesh provides the information to arrange the transmit and receive electrodes on the surface.

From this geometric arrangement, we define the 3D model for the physical prototype by back-projecting the sensor grid mesh onto the triangular mesh. We use the grid to generate the surface grooves that are conduits to accommodate the line conductors on the surface. Finally, we compute a routing of pipes that internally connect the surface conductors with the controllers placed outside the object's volume. The physical prototype is 3D printed with a dielectric material. The sensor conductors (made of enamelled unipolar solid copper) are manually pulled inside the pipes and along the grooves and connected to the capacitive touch controller. Since the proposed technique is generic, we can extend it with straightforward modifications to more automatic and advanced fabrication processes based on the simultaneous printing of dielectric and conductive materials.

We evaluated the 3D-printed prototypes by measuring the Signal-to-Noise Ratio (SNR) and the spatial accuracy of the touch detection in static and dynamic conditions. The test results show good SNR values and an excellent spatial accuracy of about 1mm, with minimum degradation for a few challenging surfaces with sharp features and issues due to manual fabrication and assembly.

2 Related Work

In this section, we place our contributions within three different contexts: the creation of touch sensing objects with a general shape;

the quad patch decomposition of a 3D model; the cable routing problem within a bounded 3D volume.

2.1 Touch Sensing

Touch is a natural way to interact with everyday objects and surfaces and can provide an intuitive and convenient user interface. To enable touch sensing, a range of technologies has been developed, including optical methods such as frustrated total internal reflection (FTIR) [Han 2005] and depth cameras [Harrison et al. 2011a; Palma et al. 2021; Wilson 2010], as well as acoustic [Harrison et al. 2011b; Michael C. Brenner 1985] and resistive methods [Sundholm et al. 2014]. Additionally, electric field sensing [Zimmerman et al. 1995], impedance profiling [Sato et al. 2012], time-domain reflectometry [Wimmer and Baudisch 2011], and electric field tomography [Zhang et al. 2017] have been explored. Among these technologies, projected capacitive sensing has emerged as the most widely adopted method [Gray 2019; Grosse-Puppendahl et al. 2013]. This technology offers the advantage of high accuracy and resolution in detecting touch events. There are various operating modes of projected capacitive sensing, which have been reviewed by Grosse-Puppendahl et al. [Grosse-Puppendahl et al. 2017]; we refer to mutual-capacitance sensing technology, commonly used for commercial touchscreens and detecting multiple simultaneous touch contacts.

Touch Sensing on Everyday Objects. Researchers have developed a variety of approaches to enable touch interaction on everyday objects and surfaces while preserving their distinct geometric, visual, and tactile features. Inherently conductive objects can act as the touch sensor [Sato et al. 2012]. A common approach for sensing on a wider range of objects is to print a deformable touch sensor on different materials using inkjet printing [Kawahara et al. 2013; Khan et al. 2019; Pourjafarian et al. 2022] or screen printing [Olberding et al. 2014]. Moreover, existing objects can be enhanced with a thin sensing layer through hydrography [Groeger and Steimle 2018], by spraying functional materials on the objects [Wessely et al. 2020; Zhang and Harrison 2018; Zhang et al. 2017], or by attaching functional stickers or patches [Cheng et al. 2020; Klamka et al. 2020; Strohmeier et al. 2018]. Another approach involves creating artificial skin with embedded tactile sensation for human-robot interaction [Cannata et al. 2008; Mukai et al. 2008; Teyssier et al. 2021; Tomo et al. 2018]. However, covering a large and highly irregular surface with touch sensing remains challenging, in particular due to the internal wiring and readout of the sensing electrodes.

Sensors in 3D Printed Objects. A stream of research investigates how to embed customized sensors in 3D-printed objects, exploring several sensing techniques. Examples include using optical fibers [Willis et al. 2012], pipes routed inside the 3D objects [Savage et al. 2014], or transmission of acoustic signals [Laput et al. 2015]. More recent work has 3D printed conductive elements along with the object, for instance, to integrate capacitive touch sensing [Burstyn et al. 2015; Schmitz et al. 2015, 2019], deformation sensing [Schmitz et al. 2017] and to integrate sensing into 3D printable metamaterial structures [Gong et al. 2021]. The field, however, still lacks a systematic workflow for the design and fabrication of high-resolution multi-touch sensors as suggested by [Götzelmann and Althaus 2016]. Building on this research, this work takes up the idea of seamless integration of touch interfaces on complex geometries. It presents a novel approach for designing and fabricating 3D printed objects with integrated high-resolution multi-touch sensors.

2.2 Quad Patch Decomposition

Our proposed quad-patch decomposition method draws inspiration from existing solutions for partitioning surfaces into quadrilateral patches. The computation of a quad patch layout serves various purposes, such as quad remeshing [Campen et al. 2015], high-order surface approximation [Panozzo et al. 2011], isomorphisms between meshes [Eppstein et al. 2008], and surface parameterization. For a comprehensive overview of quad layout generation, please refer to [Campen 2017]. A common approach for quad layout generation involves tracing boundary lines over the surface using triangular meshes accompanied by a cross-field as input [Campen et al. 2015; Pietroni et al. 2016; Razafindrazaka et al. 2015]. Similar solutions have been proposed for irregular and regular quad meshes, where the tracing becomes a simple traversal of mesh edges across regular vertices [Eppstein et al. 2008; Tarini et al. 2011]. Another class of solutions [Pietroni et al. 2016; Razafindrazaka and Polthier 2017; Razafindrazaka et al. 2015; Zhang et al. 2016] is based on creating a graph of separatrices between irregular vertices, followed by a simplification procedure formulated as a global binary optimization problem to prune the set of all possible arcs. These solutions result in an over-segmentation (for helical configuration, the graph converges towards solutions with patches made by a single quad), leading to highly complex layouts. On the contrary, our solution, starting from a quad layout induced by a Motorcycle graph [Eppstein et al. 2008], proposes a pruning of edge chains using a global binary optimization. This guarantees the creation of a simplified quad layout by removing low-quality patches (such as strip and single quad patches) and ensures a feasible design for the sensor grid on the surface. Furthermore, we define a set of geometric operators to improve the shape of the patches, similarly to [Myles et al. 2010],

The generalization of the Motorcycle Graph proposed in [Schertler et al. 2018] allows a robust quad patch decomposition for UV mapping that is insensitive to local irregularities in quad-dominant meshes. While the ultimate goal of our patch decomposition algorithm remains the same, the generalized Motorcycle Graph generates patches that are non-isomorphic to a regular grid, as the singularities can be incorporated inside the patches. A similar constraint on the singularities is proposed in [Wu et al. 2022] to cut the surface into a single topological disk with a region-growing procedure without any restrictions on the size of the disk shape. Conversely, for our purposes, we require a decomposition into regular patches to create a regular grid of touch sensors for each.

2.3 Cable Routing in a 3D domain

Routing cables and pipes in a 3D domain helps create interactive 3D-printed objects. Several solutions have been proposed to establish electrical connections among electronic components attached to the 3D-printed object by fabricating conductive traces on its surface. *SurfCuit* [Umetani and Schmidt 2017] lays out the electric parts and

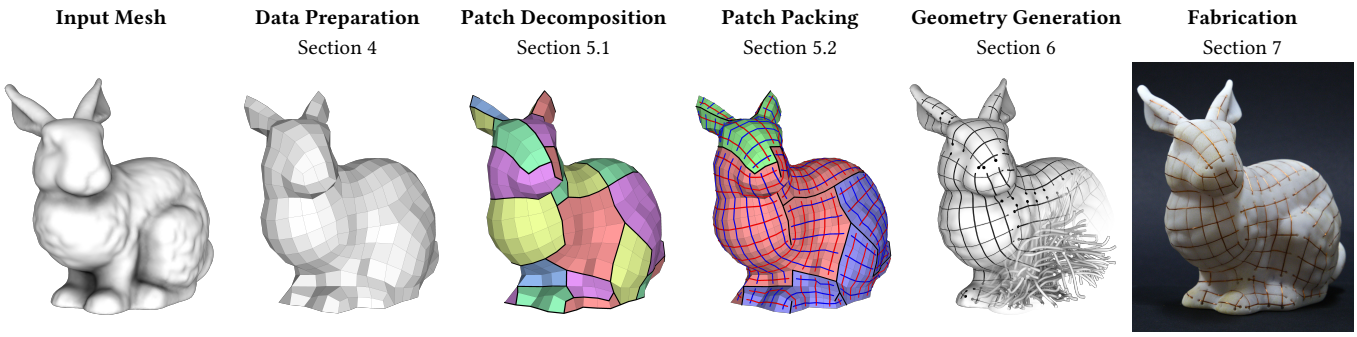


Fig. 3. Processing pipeline. Starting from a triangle mesh, we compute a proxy quad mesh, cluster the quads in a coarse quad patch layout, and pack this layout in the sensing regions of the touch controllers. Then, we compute the final geometry of the prototype, generating the surface grooves to accommodate the sensor conductors and internal pipes to connect these conductors with the controllers. The final step is the physical fabrication of the prototype.

traces on a 3D surface using a user-assisted geodesic algorithm starting from the 2D circuit diagram. *MorphSensor* [Zhu et al. 2020b] is a design tool that morphs existing sensor modules onto a 3D shape of a physical prototype. It involves assisted electronic component arrangements and a manual drawing tool to create conductive traces on the surface. *Plain2Fun* [Wang et al. 2018] allows for the automatic design of circuit layouts onto the surfaces of scanned 3D models of existing objects using geodesics on the surface. Other solutions are based on printing the circuit on a 2D layout that is deformed to adapt it to the 3D surface by thermoforming and copper electroplating [Hong et al. 2021] or by computing flat structures that self-morph into preprogrammed 3D shapes when triggered by external heating [Wang et al. 2020].

Another approach, more similar to our solution, involves creating internal pipes inside the 3D object to connect electronic components. The most relevant work [Savage et al. 2014] is based on creating internal paths using an A* search algorithm on a voxel representation of the object. It is followed by physical simulation with force constraints to smooth the paths. The pipes are routed and smoothed serially without any global optimization. The algorithm has been used in several papers focusing on the construction of touch and force sensors for interactive objects [Schmitz et al. 2015, 2019], aiming to generate a limited number of internal pipes. Expanding this approach to more complex layouts with a higher number of internal channels was only possible with the manual feedback from the user [He et al. 2022]. On the contrary, our method can generate a compact bundle of paths inside the object, avoiding that a path prevents the creation of other ones. *CurveBoards* [Zhu et al. 2020a] embeds the structure of a breadboard into the surface of a physical prototype to generate pinholes on a 3D model's surface and establish relative connections using quad meshing. In this case, generating internal pipes is easier because they connect sets of close pinholes along the same curve. Generating a few internal pipes is also used to enhance deformable input devices with internal sensors [Bächer et al. 2016].

Finally, there are solutions to generate internal pipes inside a volume to fabricate 3D-printed fiber optics automatically. One approach involves generating a compact set of fiber optics paths between two input surfaces to route light between them for sensing and display [Pereira et al. 2014]. The algorithm optimizes light transmission by minimizing fiber curvature and maximizing fiber separation while respecting constraints such as fiber arrival angle. However, it does not impose any constraints on the containing volume of the fibers. *FibAR* [Tone et al. 2020] proposes an automatic method to generate a set of fiber optics inside an object, creating a constellation of active markers on its surface for dynamic projection mapping. The method is based on an iterative refinement of initial paths defined as straight lines between the two endpoints. Nonetheless, the iterative refinement process becomes less robust as the number of routes to generate increases.

3 Method Overview

The input of our method is a watertight triangle mesh of the object to augment with multi-touch sensing. The final goal is to create a sensor grid on the mesh that is as regular as possible, enabling the detection of multiple touch points on the surface of the 3D-printed prototype. Following the capacitive sensor design guideline [Microchip 2012], we target a $7mm$ spacing among the electrode lines in the grid. To accommodate the sensor grid on the object, we need to generate two types of traces: a grid of grooves on the surface to accommodate the Rx and Tx lines of the capacitive touch controller and a set of internal pipes to connect the electrode lines on the surface to the touch controller through the object's interior. We propose the processing pipeline shown in Figure 3.

We start with a data preparation step (Section 4) that completes two tasks. The first task detects and removes the critical surface regions of the triangle mesh. The second task performs a quad meshing to achieve as regular as possible quads, ensuring an average edge length matching the desired spacing among the touch sensor lines. In this way, we attain a more uniform distribution of the intersection points of the sensor grid on the surface. The positioning of the sensing grid is determined by the edges of the dual graph of the quad mesh, which we refer to as the sensor grid mesh.

The quad mesh is the input for a new simplification and packing procedure for quad patch layout (Section 5). The main goal is to compute the minimum number of clusters of quad patches that, once packed into the rectangular sensing region of the touch controller defined by its number of Rx and Tx lines, minimize the

number of required controllers and input/output pins. Initially, we compute the quad patch layout of the quad mesh using a straightforward motorcycle graph algorithm. The simplest solution is a 2D bin packing algorithm [Lodi et al. 2002], where each patch is packed independently. However, it requires a number of input/output pins to connect the electrodes of each patch on the surface and the touch controller equal to the sum of the semi-perimeter of all patches. We propose a new procedure based on two steps to achieve our goal. The first step is a simplification method (Section 5.1) that prunes edge chains in the motorcycle graph. This simplification reduces the number of patches, emphasizing eliminating small patches composed of a quad strip or a single quad. During simplification, we enforce the creation of regular patches where all the irregular vertices are along the borders (all the internal points are regular points). This constraint allows the dual-edge graph to transform each patch into a regular grid. The second step is a greedy clustering and packing procedure (Section 5.2) to partition the simplified quad layout into the minimum number of clusters of adjacent patches, preserving the continuity of the Tx-Rx field. These clusters can be packed efficiently within the sensing region of the touch controllers, minimizing both the number of pins and the number of required controllers. The outcome of this procedure is the sensor grid mesh, which indicates the electrode placement on the surface of the quad mesh based on the constraints defined by the packing process.

The next stage involves generating the geometry of the prototype for 3D printing (Section 6). Once the sensor grid mesh is projected onto the triangle mesh, we carve the original geometry with grooves to accommodate the Tx-Rx sensor lines. Then, since the packing procedure potentially maps quad strips located far apart on the surface to the same sensing line, we need to establish connections among these strips in a serial way using only a single conductor. In this way, we simplify the fabrication step without joining multiple conductors of the same line by soldering. We generate internal pipes within the object to link these strips with the touch controller. While generating the pipes, we enforce two constraints: a curvature constraint to facilitate the smooth passage of the conductors and a minimum distance constraint among the pipes to prevent the cross-talk effect between closely parallel lines.

The final step is the fabrication process (Section 7). The 3D prototype is created by material jetting 3D printing using a plastic dielectric material. Subsequently, we manually thread the copper conductors through the grooves and internal pipes, following the order shown by a visualization tool. Finally, we set up the hardware boards, enabling touch sensing over the surface of the fabricated prototype.

4 Data preparation

The data preparation involves two steps: i) the detection and removal of the critical regions of the triangular mesh that cannot be sensorized and ii) the quad meshing of the remaining surface.

The first step is identifying regions too thin to physically accommodate the sensor grid. We require a minimum object thickness to ensure the proper arrangement of sensors made by conductors with a diameter of d_{wire}. The thickness is the sum of two terms $t_{min} = d_{sensors} + d_{pipes}$: the minimum depth $d_{sensors} = 4d_{wire}$ required to host the two types of sensing lines (Rx and Tx) within the grooves just below the surface (at different depths z_{Rx} and z_{Tx} to permit the physical creation of intersection between the lines) keeping them isolated by the internal pipes; the minimum space needed to create at least an internal pipe with diameters d_{pipes}. Our approach detects and removes surface areas with a thickness below the threshold t_{min}. Initially, we select faces of the input mesh that have a vertex at a distance from the inner surface at depth t_{min} above the threshold $t_{min} + \alpha$, where α is the voxel size used for extracting the inner surface. We then eliminate isolated regions with small areas or stretched shapes by applying the morphological opening on the selected triangles. Removing these regions avoids introducing additional border constraints that could lead to more irregular vertices, low-quality quad remeshing, and fragmented quad patch decomposition. Then, we compute smooth polylines to handle jagged borders. We use the smooth polylines to cut out the critical regions from the input triangle mesh. Additionally, we remove a user-selected area, typically at the bottom of the object, for the exit of the internal pipes connecting the electrodes with the controllers.

The second preparation step involves the quad remeshing of the remaining triangle mesh (e.g., the portion of the original surface that needs to be sensorized). We employ a state-of-the-art algorithm [Pietroni et al. 2021] that ensures high-quality isometric, pure-quad, conforming meshing while preserving feature lines such as borders and sharp creases. The remeshing algorithm's isometry and regularity properties enable the generation of quads with more uniform edge lengths, resulting in a sensor grid with evenly distributed line intersections. Preserving feature lines is important for achieving a high-quality quad mesh on the borders of the cut areas.

5 Patch Decomposition and Packing

Starting from the quad remeshing, we compute a quad patch layout decomposition using the motorcycle graph algorithm [Eppstein et al. 2008] to find a layout of regularly gridded patches that can host the Tx-Rx sensor lines. The algorithm traces motorcycle particles along the edges of the quad mesh spawned at each no-border edge around the irregular vertices. Specifically, an internal irregular vertex with a valence of $\deg(v) \neq 4$ generates $\deg(v)$ motorcycles (represented by red and green vertices in Figure 4), while a border irregular vertex with a valence of $\deg(v) \geq 4$ generates $\deg(v) - 2$ particles (represented by yellow vertices in Figure 4). The tracing is done in parallel, and each motorcycle advances straight in a topological sense. A motorcycle stops when it reaches a vertex already visited by another motorcycle or a mesh border. If two motorcycles collide in the same regular vertex from orthogonal directions during the same tracing iteration, we stop the motorcycle created by the irregular vertex with higher valence. If they have the same valence, we stop the motorcycle generated by the vertex with the higher index. The output is a partition of the mesh in regular rectangular patches with all the irregular vertices located on the border (Figure 4a and 4d).

As discussed in [Eppstein et al. 2008], the resulting partition of the motorcycle graph algorithm is an over-segmentation, and computing a partition with the minimum number of patches is an NP-complete problem. A well-defined approach to reduce the number of patches is to trace a smaller number of motorcycles

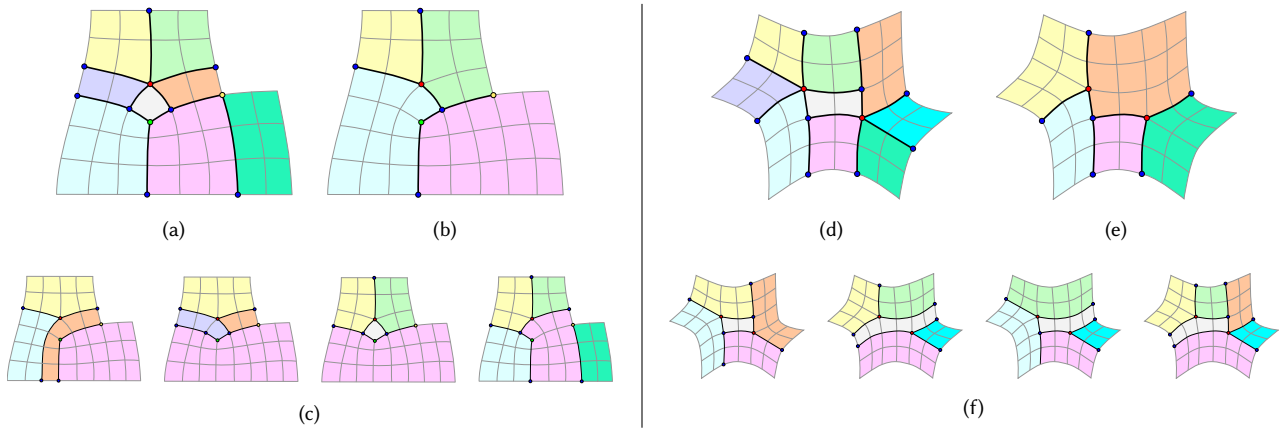


Fig. 4. Starting from the irregular vertices, the motorcycle graph algorithm computes a quad patch layout (Figures 4a and 4d blue: *T-vertex*, green: deg(3), red: deg(5), yellow: *irregular border*). As described in Section 5, we reduce the number of patches and the over-segmentation (Figure 4b and 4e). In the bottom (Figure 4c and 4f), we show some not-optimal quad layouts generated by motorcycle tracing using $n = \lceil \deg v/2 \rceil$ particles for each irregular vertex; while coarser than the plain motorcycle output, they present problematic issues (quad strips, single quad patches or a higher number of patches).

$n = \lceil \deg(v)/2 \rceil$ for each irregular valence-deg(v) vertex, ensuring that no two consecutive incident edges of the irregular vertex remain unused. However, the selection criteria for determining which edges to use around each irregular vertex significantly impact the quality of the final quad layout partition, specifically in terms of the number of patches and their shape quality, as shown in Figures 4c and 4c. Our objective is to avoid the creation of a partition with quad strips or single quad patches, as they require a higher number of input/output pins concerning the covered area. These low-quality patches arise from singularity configurations generated by the quad meshing algorithm. The first configuration involves a quad with two irregular vertices on the diagonal, each having a different valence (typically 5 and 3). This configuration generates a single quad patch and two quad strips (as shown in Figure 4a). The second configuration produces a quad strip (as depicted in Figure 4d) due to the misalignment of two irregular vertices. This misalignment originates from the singularity alignment term within the patch-side tessellation process described in [Pietroni et al. 2021], which may not be satisfied in specific shape configurations because it is modelled as a soft constraint.

To minimize the number of patches and improve their shape quality, we propose a new simplification algorithm for the quad patch layout generated by the motorcycle graph (Figure 5a). The algorithm prunes edge chains generated by the motorcycle particles by solving a global Integer Linear Program (ILP) followed by an energy optimization to improve the shape quality of the patches (Section 5.1 and Figure 5b). Once the patch layout has been simplified, we pack the patches within the rectangular sensing region defined by the number of Tx and Rx lines of the capacitive touch controller (Section 5.2). The objective is to establish a mapping function of each patch inside the sensing region that preserves the adjacencies between patches as much as possible. We tackle this problem by solving the global assignment of the Tx and Rx roles to each patch side, aiming to minimize discontinuities between adjacent patches. Subsequently, we employ a greedy approach to cluster the quad patches, ensuring the minimum number of clusters is created to reduce the total number of input/output pins (Figure 5c). Each cluster must have a bounding rectangle that does not exceed the size of the sensing region. Finally, we pack the clusters within the sensing region, optimizing the placement to reduce the overall number of required capacitive touch controllers (Figure 5d).

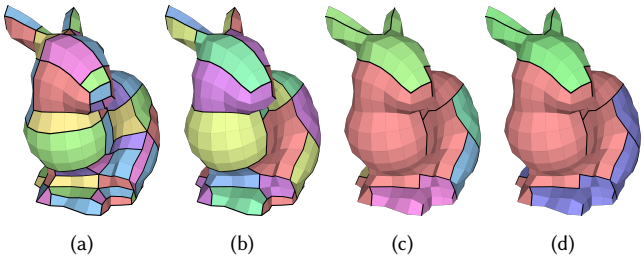


Fig. 5. Quad patch decomposition and packing at the different stages of the procedure described in Section 5: (a) patch layout by the motorcycle graph; (b) patch layout after the simplification algorithm in Section 5.1; (c) clustering of the patch layout; (d) packing of the cluster into the sensing region of the controllers.

5.1 Quad Patch Layout Simplification

Using the paths generated by the motorcycles, we build the relative tracing graph $G = \langle V = V_s \cup V_t, E \rangle$. The set V contains the irregular vertices V_s and the T-vertices V_t generated by a motorcycle collision with the path of another motorcycle or with the mesh border (blue vertices in Figure 4). The set E contains an edge for every pair of vertices in V connected by a chain of edges of the quad mesh visited by a single motorcycle or two motorcycles after a head-on collision. In the first case, the edge connects an irregular vertex and a T-Vertex, and we assign it a multiplicity weight $m_i = 1$. In the second case, it connects two aligned irregular vertices, and we assign it a multiplicity weight $m_i = 2$. Each edge stores its list of

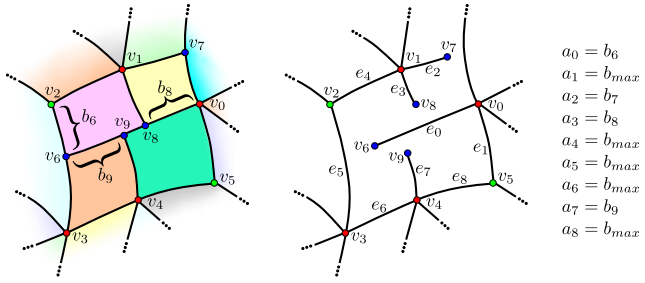


Fig. 6. (Left) Example of patch decomposition by the motorcycle graph. The figure shows the irregular vertices ($v_0 \dots v_5$) and the T-vertices ($v_6 \dots v_9$) with the relative orthogonal weights b_i. (Right) Tracing graph created by the motorcycles. The figure shows the generated edges ($e_0 \dots e_8$) with the relative weight a_i. The edges e_1, e_4, e_5, e_6, e_8 have multiplicity $m_i = 2$, while edges e_0, e_2, e_3, e_7 have $m_i = 1$.

edges within the quad mesh as a half-edge list. Additionally, we define a per-vertex multiplicity c_i to handle loops around the same irregular vertex. This multiplicity is $c_i = 2$ if there exists an edge $e_i = \langle v_j, v_k \rangle$ such that $v_j = v_k$, and $c_i = 1$ otherwise.

For every T-vertex $v_i \in V_t$, we assign the orthogonal edge that led to its generation, corresponding to the graph edge where the motorcycle collided during the tracing process. Additionally, we compute the orthogonal weight b_i for each T-vertex, defined as the quad edge distance from the close vertex along the orthogonal edge. Specifically, b_i represents the shortest distance from the T-vertex to the next corners of the two patches created by v_i. To compute this distance, we navigate the half-edge list of the internal perimeter of these patches in opposing order (one counterclockwise and the other clockwise) starting from v_i. Furthermore, we compute the orthogonal weight a_i for each edge e_i. If the edge e_i is connected to a T-vertex, the weight a_i equals the orthogonal weight b_i of the T-vertex. If the edge connects two irregular vertices, the weight a_i is set to the maximum value of b_i among all the T-vertices. Figure 6 shows an example of patch decomposition and the relative tracing graph with the orthogonal weights.

ILP Graph Reduction. We use the tracing graph G as input for a binary ILP problem to prune its edges. The objective is to minimize the number of quad patches created by the graph, preserving a regular quad partition and removing as many quad strips and single quad patches as possible. Let $x_i \in \{0, 1\}$ be a binary variable for the edge e_i in G, where $x_i = 0$ means that the edge is pruned in the final layout, and $x_i = 1$ indicates that is kept. We set the following minimization problem:

$$\min \sum_{e_i \in E} w_i x_i \tag{1}$$

where the edge weights w_i are defined in equation 9 in the following. To model different requirements, we employ three sets of linear constraints. The first set of constraints encodes the idea of using $n = \lceil \deg(v)/2 \rceil$ incident edges for each irregular $\deg(v)$ vertex to produce a valid partition with fewer quad patches. These constraints are defined as follows:

$$\sum_{e_i \in E} m_i x_i \quad \geq t \tag{2}$$

$$\forall v \in V_s \quad \sum_{e_i \in I(v)} c_i x_i \quad \geq \left\lceil \frac{\deg(v)}{2} \right\rceil \tag{3}$$

$$\forall \langle e_i, e_{i+1} \rangle \mid e_i, e_{i+1} \in I(v) \quad x_i + x_{i+1} \quad > 0 \tag{4}$$

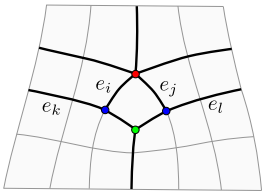
Specifically, equation 2 constrains the minimum number of edges in the simplified layout to be at least t to ensure a regular quad patch partition. The value of t is defined as:

$$t = \sum_{v_i \in V_{si}} \left\lceil \frac{\deg(v)}{2} \right\rceil + \sum_{v_i \in V_{sb}} \left\lceil \frac{\deg(v) - 2}{2} \right\rceil \tag{5}$$

where V_{si} represents the internal irregular vertices and V_{sb} represents the irregular vertices on the border ($V_s = V_{si} \cup V_{sb}$). Equation 3 enforces the minimum number of edges to be selected for each irregular vertex. Equation 4 ensures that at least one edge must be used in the final layout for every pair of consecutive incident edges on an irregular vertex. The operator $I(v_i) \subset E$ returns the set of incident edges for the vertex v_i ordered counterclockwise.

The second set of constraints focuses on a specific issue that frequently affects the final quality of the patch layout: single quad patches. For this situation, we model the edge selection around a single quad patch with two irregular vertices of different valence on the diagonal (as shown in the right inset):

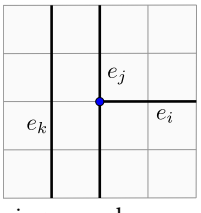
$$\begin{aligned} x_i + x_j &= 1 \\ x_k + x_l &= 1 \\ x_i - x_k &= 0 \\ x_j - x_l &= 0 \end{aligned} \tag{6}$$



These constraints choose between merging the single quad with the left patches (by removing the edges e_i and e_k in the right inset) or the right patches (by removing the edges e_j and e_l in the right inset). In this way, we can remove the single quad patch without creating dangling T-vertices.

The third set of constraints deals with the T-vertices to ensure a valid patch layout after the pruning process (see the right inset):

$$x_j - x_i \geq 0 \tag{7}$$



TThe constraint prevents the creation of a dangling T-vertex. If the edge e_i is selected in the final layout and it creates a T-vertex on the edge e_j then we force the selection of e_j. This constraint could be removed by applying an additional tracing procedure starting from all the created dangling T-vertices at the end of the ILP solving. The additional tracing procedure can generate a new quad strip, as shown in the last three cases in Figure 4f, not ensuring the creation of a better quality layout. The constraint can be relaxed, allowing for the pruning of the orthogonal edge e_j while preserving edge e_i, if there exists an edge e_k that can be reached by the dangling T-vertex with a single tracing step (as

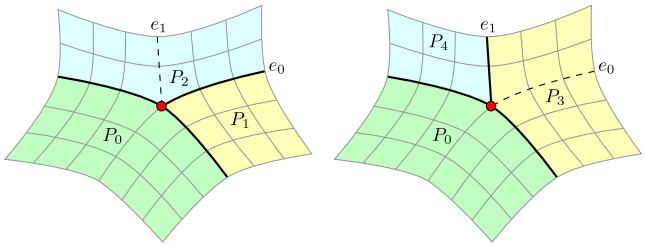


Fig. 7. Example of an edge swap operation. We obtain a new layout with a lower patch shape energy by deleting the edge e_0 and adding the edge e_1 to the patch layout on the left. In this case, the operation reduces the energy of a value $\Delta\mathcal{E} = \mathcal{E}(P_1) + \mathcal{E}(P_2) - \mathcal{E}(P_3) - \mathcal{E}(P_4) = 0.52\overline{7}$.

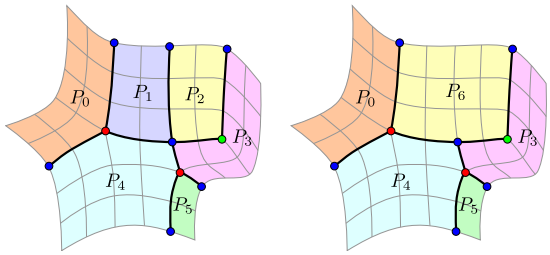


Fig. 8. Example of merge operation of the patches P_1 and P_2 in the new one P_6. In this case, the operation reduces the energy of a value $\Delta\mathcal{E} = \mathcal{E}(P_1) + \mathcal{E}(P_2) - \mathcal{E}(P_6) = 1.7\overline{2}$.

shown in the right inset). In such cases, the tracing step is considered safe since it does not result in additional quad strips, and we substitute the constraint with the new one:

$$x_k + x_j - x_i \geq 0 \tag{8}$$

Finally, we address the quad strip patches created by two misaligned irregular vertices, introducing a soft constraint in the minimization problem defined in Equation 1. This constraint is based on the weight w_i assigned to each edge, given by the equation:

$$w_i = 1 - \frac{a_i}{a_{max} + 1} \tag{9}$$

where a_{max} is the maximum orthogonal weight a_i over all the edges. The purpose of this weight assignment is to prefer the removal of edges with T-vertices resulting in narrow patches, such as the quad strip illustrated in Figure 4d. We opt for a soft constraint to avoid an unsolvable problem for some graph configurations.

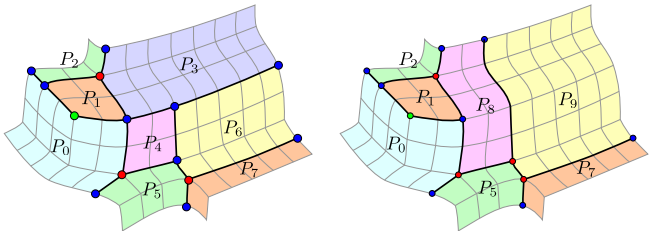
Patch Geometric Optimization. To ensure the solvability of the ILP problem, the output patch layout may contain more edges than the minimum number defined by t (Equation 5). These additional edges are introduced by the inequalities in Equations 2 and 3. Setting these constraints to strict equalities often makes the ILP solver unable to find a feasible solution for more complex shapes. Therefore, to further improve the quality of the layout, we proceed with a classical greedy geometric optimization. The main objective is to modify the patch layout through a sequence of local operations to decrease the energy by modifying the patches' size. The total energy of the partition is defined as the sum of the energies of all the individual patches. For each patch P, the energy is computed using the following equation:

$$\mathcal{E}(P) = \frac{\mathcal{P}(P)}{\mathcal{A}(P)} \max\left(\mathcal{S}(P), \frac{1}{\mathcal{S}(P)}\right) \tag{10}$$

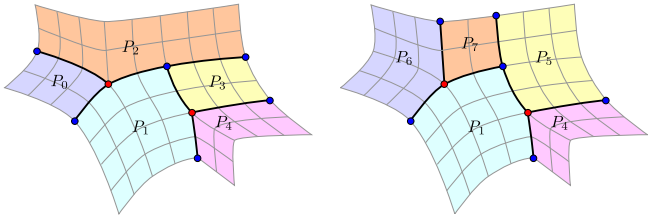
where $\mathcal{P}$, $\mathcal{A}$, and $\mathcal{S}$ are the perimeter, the area, and the shape ratio (width-to-height ratio) operators. The energy takes into account both the compactness of the patch and its shape ratio. Its minimization encourages the creation of larger, square-shaped patches while preventing small and elongated ones. This approach aims to reduce the input/output pins required for the patches, which is proportional to their semi-perimeter.

The geometric optimization process consists of two iterative steps, each aimed at reducing the partition energy until further improvement is not possible. In the first step, we define the *Edge Swap* operator to identify a sequence of swap operations to invert the selection state of some edges in the tracing graph G. At each iteration, we gather all the feasible potential swap operations on pairs of consecutive edges $\langle e_i e_j \rangle$, belonging to the same irregular vertex v, but with opposite selection states. A swap operation is feasible if it meets three conditions: the edge to be deselected should have a multiplicity $m = 1$ (it creates a T-vertex), its deselection should not result in the creation of dangling T-vertices, and the operation must not violate the constraint outlined in Equation 4. For each potential swap, we evaluate the energy changes $\Delta\mathcal{E}$ introduced in the sizes of the affected patches. In Figure 7, $\Delta\mathcal{E}$ is calculated as $\Delta\mathcal{E} = \mathcal{E}(P_1) + \mathcal{E}(P_2) - \mathcal{E}(P_3) - \mathcal{E}(P_4)$, where P_1 and P_2 represent the patches that are removed by the swap, and P_3 and P_4 denote the newly created ones. Then, we sort the potential operations by energy change and perform the one with the maximum energy decrease (with the maximum $\Delta\mathcal{E}$). We prioritize the swaps that result in the most significant energy reduction.

In the second step, we apply a series of split and merge operations. We define two types of operators: the *Merge* operator and the *Split&Merge* operator. The *Merge* operator joins two adjacent patches if their corners over the shared side coincide (Figure 8). This operation reduces the number of patches by one. The *Split&Merge* operator divides a patch by tracing a set of particles from the selected side for the split to the opposite one. We generate a particle for each T-vertex and no-corner irregular vertex along the selected side. Then, it merges the adjacent patches on the split side using the same condition of the *Merge* operator. The *Split&Merge* operator can reduce the number of patches by one if the merge operations can be applied to all the adjacent patches on the selected side for the split. Otherwise, it only reduces the energy while keeping the number of patches constant. We refer to these two versions as *Symmetric Split&Merge* (Figure 9a) and *Asymmetric Split&Merge* (Figure 9b), respectively. The optimization process begins by collecting a list of all *Merge* and *Split&Merge* operations that can be performed on the layout. We select and perform the operation that maximizes the decrease in energy, defined as the difference between the energy of the removed patches and the energy of the newly created ones. Finally, we update the adjacency information in the layout and the



(a) *Symmetric Split&Merge*



(b) *Asymmetric Split&Merge*

Fig. 9. Example of *Split&Merge* operations. (a) *Symmetric Split&Merge* of the patch P_3. In this case, the operation reduces the energy of a value $\Delta\mathcal{E} = \mathcal{E}(P_3)+\mathcal{E}(P_4)+\mathcal{E}(P_6)-\mathcal{E}(P_8)-\mathcal{E}(P_9) = 1.81$. (b) *Asymmetric Split&Merge* operation of the patch P_2. In this case, the operation reduces the energy of a value $\Delta\mathcal{E} = \mathcal{E}(P_0) + \mathcal{E}(P_2) + \mathcal{E}(P_3) - \mathcal{E}(P_5) - \mathcal{E}(P_6) - \mathcal{E}(P_7) = 1.9\overline{4}$.

list of operations. We iterate the process until no further operations can be performed.

5.2 Patch Layout Packing

In this stage, we generate the sensor grid using the edges of the dual graph of the quad mesh. We propose a greedy procedure to assign the sensing role (Tx or Rx) to each line of the sensor grid. The procedure is composed of two steps: the clustering of the quad patches of the simplified layout that, preserving the adjacency among the patches, reduces the total number of input/output pins; the packing of the computed clusters inside the sensing regions of the minimum number of capacitive touch controllers. Before the procedure, if a patch exceeds the dimensions of the sensing region of the touch controller, we split it into multiple patches, avoiding the creation of quad strips or single-quad patches.

The procedure starts pre-assigning the sensing roles (Tx or Rx) to each side of the quad patches to minimize role cuts among adjacent patches. A role cut occurs when two adjacent patches have different role assignments on their shared side, generating orthogonal Tx-Rx fields. Since each cut introduces an additional conductor to connect the grid electrodes to the touch controllers, we can decrease the required input/output pins by minimizing the length of these cuts. To solve the pre-assignment, we formulate a simple ILP problem. We define two binary variables for each quad patch, $x_{i0}, x_{i1} \in \{0, 1\}$. These variables represent the role assignment for a pair of opposite sides of the patch, where $x_{ij} = 1$ indicates that the grid lines orthogonal to the corresponding sides are transmitters and $x_{ij} = 0$ indicates receivers. For each patch, we constrain the two variables to have opposite values $x_{i0} + x_{i1} = 1$. Let S be the set of all pairs of adjacent patches $\langle P_i, P_j\rangle^{lk}$. For each pair, we define a binary cut variable $s_{ij}^{lk} = x_{il} \oplus x_{jk}$ equals to the exclusive OR between the variables of the adjacent sides of the two patches. Here, l and k represent the indices of the adjacent sides of the two patches (modulo two). The variable s_{ij}^{lk} indicates whether the shared edge between the two patches is a cut (i.e., when x_{il} and x_{jk} have opposite values) or not. The assignment problem is defined as a minimization of the weighted sum of the cut variables:

$$\min \sum_{\forall \langle P_i, P_j\rangle^{lk} \in S} w_{ij} s_{ij}^{lk} \tag{11}$$

where w_{ij} is the length of the shared edge between the adjacent patches defined as number of shared quad edges. The objective is to minimize the total length of the sensing role cuts.

We use the sensing role pre-assignment in a greedy clustering procedure to create the smallest number of clusters, each with a bounding rectangle contained in the sensing region of the touch controller. For each patch, we compute its largest potential cluster through an iterative method that, in each iteration, adds an adjacent patch to the current cluster border. The next patch to insert into the cluster must satisfy three conditions. The first condition allows adding the patch if at least one edge shared with an adjacent patch on the cluster border is not a role cut. With this condition, we can create uninterrupted sensor lines across the border edges, reducing the number of input/output pins by one for each adjacent quad edge that is not a role cut. The second condition checks if adding the patch would not result in a bounding rectangle of the cluster exceeding the size of the sensing region. The third condition checks if adding the patch does not create an overlap collision with other patches inside the cluster. Among the patches that meet these requirements, we add the patch with the maximum number of shared non-cut edges on the adjacent border to the cluster. This selection criterion maximizes the removal of input/output pins. Given the potential clusters, each one generated by a patch, we keep the cluster with the maximum number of removed input/output pins, mark all the patches belonging to the cluster as visited, and restart the method using only the not-visited patches. This method concludes when all patches are successfully assigned to a cluster.

The final step involves packing the computed clusters within the sensing region of the touch controllers. We use a simple raster-based texture packing algorithm. Given a texture with dimensions corresponding to the maximum number of Tx and Rx lines, we aim to pack the clusters in the minimum number of controllers while minimizing empty space. The algorithm starts by sequentially packing the clusters in the sensing region of the first controller, following a descending order based on the cluster area. When no more clusters can be accommodated in the current controller, the algorithm allocates a new one and restarts the process using the remaining unpacked clusters. The packing position inside the sensing regions is computed by minimizing modifications to the top and right horizons before and after inserting the cluster. This energy change is computed using two lists that store all the free space in the texture along its rows (from left to right) and columns (from bottom to top). To achieve better packing efficiency, we compute the best position

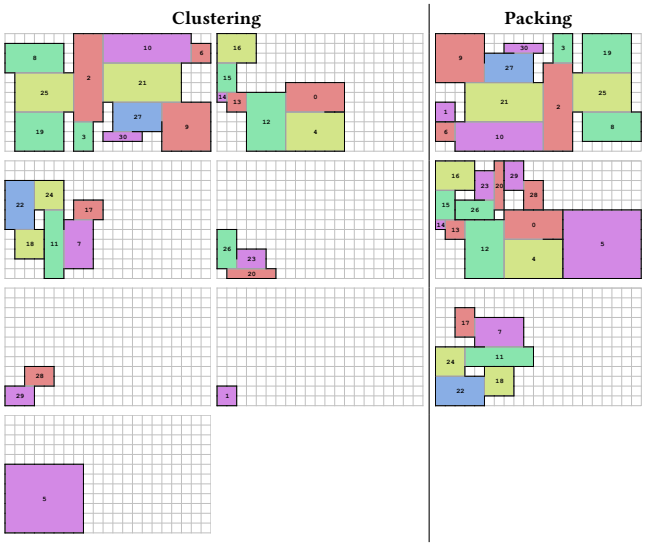


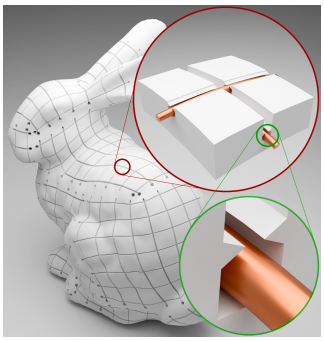
Fig. 10. Starting from the 31 patches obtained on the Bunny model with the method in Section 5.1, the algorithm in Section 5.2 creates 7 clusters and packs the clusters in 3 touch controllers with 21 Tx and 12 Rx.

for each cluster by evaluating a set of transformed versions. These transformations include ±90° rotation (if the rotated cluster remains within the sensing region), 180° rotation, vertical and horizontal flipping, and their combinations. Figure 10 shows the clustering and the packing results for the Bunny using 3 touch controllers with 21 Tx and 12 Rx.

6 Geometry Prototype Generation

The clustering and packing of the quad patch layout into the sensing regions of the capacitive touch controllers provide us with the information needed to compute the position of the sensor electrodes on the surface. For each controller, we create an independent sensor grid using the dual graph of the mesh, which we refer to as the sensor grid mesh. Each intersection in the grid represents a quad in its dual form. Within each controller, we create an electrode for each line (transmitters and receivers) consisting of one or more polyline segments on the surface. We divide the electrode whenever it crosses an edge between two quads mapped in consecutive positions along the line but not adjacent along this edge in the mesh. Since the sensing controllers require a single conductor for each line, we need two types of traces to place the sensor on the surface: surface grooves to accommodate the conductors on the surface and internal pipes to connect each line to the controller and the segments of the same line in series through the object's volume.

The first step involves projecting the sensor grid mesh onto the triangular mesh. We move each vertex of the sensor grid mesh to the nearest point on the triangle mesh, and we apply an iterative refine procedure to create smoother polylines for each line segment. Specifically, we perform Laplacian smoothing while keeping the vertices on the cross between the transmitter and receiver lines fixed. We then reproject the smoothed vertices onto the triangulated surface. Finally, we simplify and refine the polylines to adapt them to the underlying triangulation. We generate the grooves with a procedural method that uses different depths, denoted as z_{tx} and z_{rx}, for transmitters and receivers, respectively. The two depths allow us to create the intersections between Tx and Rx (the sensing points) with the physical arrangement of the sensors just beneath the surface.



For each polyline in the sensor grid mesh, we extrude a rectangular profile with a snapping mechanism to hold the physical conductors on the surface still. In the inset figure, the red detail illustrates the arrangement of the electrodes at different depths, while the green detail focuses on the snapping mechanism. To determine the orientation of the profile, we use the vector connecting each vertex of the polyline with its closest point on the inner surface at distances z_{tx} for the transmitters and z_{rx} for the receivers.

Then, we generate two types of pipes with diameters d_{pipe} for each sensor line: exit pipes and intra-pipes. The exit pipes connect the electrodes on the surface with the touch controller. They extend from the starting point of the first groove of each line to a point inside the exit region selected by the user (typically located at the bottom of the object). The potential endpoints are precomputed and evenly distributed around the centre of the exit region. They are assigned to each pipe during the routing process to form a compact cluster around the centre. The intra-pipes are required to connect the different grooves in the same line in series. Each intra-pipe connects the end of a groove to the starting point of the next one. To prevent conflicts with the surface grooves, the pipes are generated inside the volume of the inner surface I of the triangle mesh at a distance t_{min}. Consequently, we project all the start and end points of the grooves onto the inner surface I, establishing the initial edge for all the pipe paths, which remains unchanged throughout the subsequent processing steps.

The routing of the pipes is computed using Dijkstra's algorithm on the graph constructed from the edges of the solid voxelization of the volume of the inner surface at t_{min}. This graph is enhanced by incorporating the endpoints of the grooves and the edges connecting them at the 4-nearest voxelization vertices. We perform a voxelization with an edge length equal to d_{pipe} to ensure that the pipes do not intersect during path generation. Before routing, we reevaluate the order in which the conductors must traverse the different grooves belonging to the same line in series. The objective is to minimize the conductor length while preserving the intersections between the transmitter and the receiver.

The routing method proceeds by handling one path at a time: it extracts a path, removes the used vertices from the graph, and continues extracting the following path. Our approach aims to generate paths as close as possible to the centre of the volume, thereby preventing a pipe from obstructing the generation of other pipes due to the proximity to the surface. To meet this requirement, we focus on two aspects. The first aspect involves the distance function L used in Dijkstra's algorithm between the current vertex n_i and an

unvisited one n_j:

$$L(n_i, n_j) = \|n_i - n_j\|_2 \frac{a}{w_j + a} \tag{12}$$

where w_j is the shortest distance from the unvisited node to the bounding inner surface I. This function reduces the distance for points farther from the border, attracting the path closer to the centre. The time-dependent constant a determines the speed of this attraction effect. During the routing of the exit pipes, we increase a to relax the routing constraint within the centre of the volume, as the previously generated pipes should have already occupied this region. In particular, we set $a = \sqrt{k}$, where k is the generation order index of the current path.

The second aspect concerns the order in which the paths are generated. We begin by routing the exit pipes, followed by the intra-pipes. Within each group, we first compute the pipe with the longest path. To approximate the path length, we use the shortest path tree computed from the centre of the exit region to the endpoints of the polylines for all sensor lines. For an exit pipe, we consider the length of the path connecting the root of the tree to the start point of the pipe. For an intra-pipe, we consider the shortest path connecting the two endpoints of the pipe on the shortest path tree.

While the start and end points of the intra-pipes are well-defined, determining the endpoint for the exit pipe requires selecting from the candidate points generated around the centre of the exit region. For the path generation of each exit pipe, we get the set of not-assigned exit points on the border of the regions containing the ones already assigned to the previously generated pipes. Then, we compute the shortest paths from each of them to the start point of the pipe. We select the path generated by the exit point that minimizes the sum of the path length and its distance from the barycenter of the already assigned exit points. Both distances are normalized by the longest path length and the farthest exit point distance, respectively. The objective is to choose the exit point that not only results in the shortest path but also minimizes the spread of the exit points, thereby creating a compact cluster. See the supplemental materials for the figures of the compact clusters of exit points at the bottom of the processed models.

The final step involves smoothing the generated paths with an iterative procedure by alternating Laplacian and Bilaplacian smoothing. Bilaplacian smoothing prevents small curvature radii along the path, particularly near the fixed endpoints, which could pose challenges during the physical pulling of the sensor conductors (Figure 11). Given the set of paths, each one C^j defined as a polyline of ordered vertices $C^j = \{\mathbf{c}_i\}$, the procedure in Algorithm 1 computes the Laplacian vector $\mathbf{l}$, the Bilaplacian vector $\mathbf{h}$, and two repulsive forces for each vertex $\mathbf{c}_i$ in the path. The first force $\mathbf{f_A}$ guarantees a minimum distance between the pipes, defined as a multiple α of the pipe diameter d_{pipe}, preventing cross-talk among the sensor lines. This force is applied from the closest points of neighboring pipes. The force magnitude for each close point is the displacement to move it at least at the minimum distance from the vertex c_i. The second repulsive force $\mathbf{f_B}$ confines the position of the paths within the inner volume I. This force is applied from the vertices of I with a magnitude equal to the displacements to move the center of the pipes at least at a distance d_{pipe} from the boundary. With these

Algorithm 1 Pipe smoothing

Input:
$C \leftarrow \{C^0, ..., C^n \mid C^j \leftarrow \{\mathbf{c}_i\}\}$ ▷ Input pipe paths
I ▷ Inner boundary surface
$\alpha \leftarrow 2$ ▷ Scale factor for min distance between paths
$\beta \leftarrow 3$ ▷ Weight for bilaplacian vector
$\lambda \leftarrow 0.1$ ▷ Displacement weight

Procedure:
for $t \leftarrow 1, 1000$ **do** ▷ Smoothing iteration
  for all $C^j \in C$ **do** ▷ For each path
    for all $c_i^t \in C^j$ **do** ▷ For each vertex in the path
      $\mathbf{l}_i \leftarrow (\mathbf{c}_{i+1}^t + \mathbf{c}_{i-1}^t)/2 - \mathbf{c}_i^t$
      $\mathbf{h}_i \leftarrow (\mathbf{l}_{i+1} + \mathbf{l}_{i-1})/2 - \mathbf{l}_i$
      $A_i \leftarrow \{\mathbf{a} \in C^k \mid (k \neq j) \wedge \|\mathbf{c}_i^t - \mathbf{a}\| < \alpha d_{pipe}\}$
      $B_i \leftarrow \{\mathbf{b} \in I \mid \|\mathbf{c}_i^t - \mathbf{b}\| < d_{pipe}\}$
      $\mathbf{f}_{\mathbf{A}i} \leftarrow \sum_{\mathbf{a} \in A_i} (\alpha d_{pipe} - \|\mathbf{c}_i^t - \mathbf{a}\|) \frac{(\mathbf{c}_i^t - \mathbf{a})}{\|\mathbf{c}_i^t - \mathbf{a}\|}$
      $\mathbf{f}_{\mathbf{B}i} \leftarrow \sum_{\mathbf{b} \in B_i} (d_{pipe} - \|\mathbf{c}_i^t - \mathbf{b}\|) \frac{(\mathbf{c}_i^t - \mathbf{b})}{\|\mathbf{c}_i^t - \mathbf{b}\|}$
      $\mathbf{f}_i \leftarrow (\mathbf{f}_{\mathbf{A}i} + \mathbf{f}_{\mathbf{B}i})/(|A_i| + |B_i|)$
      if $t \equiv 0 \pmod 2$ **then**
        $\mathbf{c}_i^{t+1} \leftarrow \mathbf{c}_i^t + \lambda(\mathbf{l}_i + \mathbf{f}_i)$
      else
        $\mathbf{c}_i^{t+1} \leftarrow \mathbf{c}_i^t + \lambda(\mathbf{f}_i - \beta \mathbf{h}_i)$

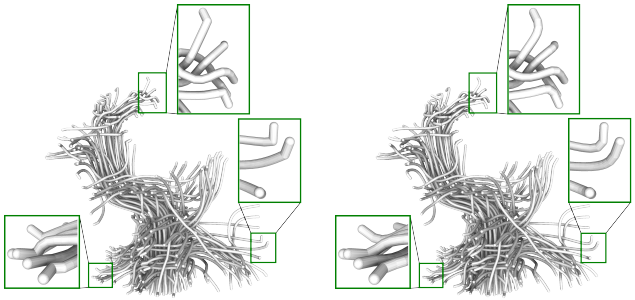


Fig. 11. Smoothing of the pipes without Bilaplacian (left) and with Bilaplacian term (right). The Bilaplacian prevents the creation of small curvature radii near the endpoints of the path. The first edge in the path stays fixed during the smoothing process to avoid conflicts with surface grooves.

vectors, the procedure computes the displacement to update the position of each point $\mathbf{c}_i$. The magnitude of the update at each iteration is determined by the displacement weight λ. For all the tested models in the paper, we use 1000 iterations in the smoothing procedure with parameters $\alpha = 2$, $\beta = 3$, and $\lambda = 0.1$. The final paths of the pipes serve as the basis for generating their geometry, achieved through the extrusion of a circular profile along the paths.

7 Prototype Setup

We generate the geometry of the final prototype by performing a boolean difference operation between the triangle mesh and the shapes of the grooves and internal pipes. For the 3D printing of

the prototypes, we employ a material jetting 3D printer (3DSystem ProJet MJP 2500 Plus) based on the use of two materials: a plastic dielectric material (VisiJet M2R-WT) for the object and a wax-based support material (VisiJet M2 SUP). The support material is removed in post-processing using a warm oil bath. We create a hollow interior within the object to ensure more effective support removal, particularly in the internal pipes. It involves generating a thin shell inside the object, leaving some openings at the bottom of the object. This hollow interior speeds up and facilitates wax melting within the internal pipes.

The 3D printed prototype is equipped with conductors of enamelled unipolar solid copper of diameter $d_{wire} = 0.5mm$. These conductors are manually placed in the grooves by pressing them from the outside and inside the pipes using a nylon probe for pulling. Despite the absence of a solid dielectric material to separate the transmitters and receivers at their intersections, copper wires with an enamelled coating ensure no short circuits between the conductors. We begin by placing the conductors for the transmitters, which are positioned at a greater depth from the surface, followed by the placement of conductors for the receivers. We employ a simple visualization tool to display the order in which each conductor must be pulled along the correct sequence of grooves and internal pipes generated for its sensor line. The lack of robust and accessible automatic technologies capable of printing simultaneously dielectric and conductive materials is the reason behind the manual procedure for placing the sensor conductors on the 3D-printed prototypes. The existing 3D printing solutions have two main drawbacks. Fused Deposition Modeling (FDM) technology relies on conductive PLA filaments doped with graphite or infused with copper, which have higher resistivity (ranging from $15\Omega \cdot$ cm to $6 \times 10^{-3}\Omega \cdot$ cm) than common capacitive sensor materials like Indium Tin Oxide ($7.5\times10^{-4}\Omega\cdot$cm) or copper ($1.62\times10^{-6}\Omega\cdot$cm). The higher resistivity makes them less suitable for capacitive sensing. Additionally, even the most conductive filaments face issues during nozzle extrusions due to printing temperature, printing speed, and retraction. These issues can cause conductive material smearing, compromising the sensor lines' electric insulation. A more promising approach is the Multi Jet Fusion technology [Wittkopf et al. 2019] with a jettable conductive agent, although it requires further study on the precision of the electric insulation properties.

After the fabrication of the sensor, the transmit and receive electrodes are connected to the touch controllers. Although various commercially available touch controllers can be selected, we employed the Muca breakout board[1], featuring the FocalTech FT5316DME capacitive touch controller. This touch controller supports up to 21 transmit (Tx) and 12 receive (Rx) lines. For designs requiring more Tx and Rx lines, we use two or more touch controllers that are read sequentially. Serial communication between the touch controller and the host processor is established using a two-wire interface based on the I2C protocol. We used a Teensy 4.1 microcontroller board to communicate with the touch controllers. The raw mutual capacitance values, with a 16-bit resolution, were transmitted to a PC by USB communication at a scan rate of 10Hz. We calibrate each touch controller by subtracting the background noise at the system startup. It is achieved by averaging the capacitive values sensed in the first 2 seconds without touch interaction.

[1]https://muca.cc/

7.1 Touch Points Computation

To compute and visualize the touch information on the 3D models, we have implemented a simple real-time tool to display the raw values sensed by the touch controllers on the quad proxy geometry, along with the interpolated version of these data to calculate the relative touch points on the triangle mesh. For the last task, we use a simple algorithm to compute continuous touch positions on a touch surface. Starting from the dual-edge graph of the quad mesh, we project all its vertices onto the surface of the triangle mesh, inheriting both the position and surface normal. Each vertex receives the raw value from the corresponding quad in the proxy geometry. Then, we apply a simple blob detection algorithm on the vertices of the dual graph, using a local maximum-minimum thresholding. Finally, for each blob, we compute the centroid of the positions and normals of its vertices and perform a ray marching procedure to determine the relative position in the triangle mesh. The computed touch positions (up to five simultaneously) are displayed on the triangle mesh surface in real-time.

8 Results

In this section, we present results for two different aspects. The first involves the patch decomposition and packing algorithm, showcasing data from various stages of the proposed algorithm (Section 8.1). We also evaluate the number of conductors and touch controllers needed to sensorize the prototypes (Section 8.2) for different input parameters. The second aspect focuses on the quality of the touch sensing, defined as Signal-to-Noise Ratio (SNR) (Section 8.3), and the spatial accuracy (Section 8.4), evaluated on the fabricated prototypes. These estimations were performed directly on the raw quantized capacitive values sent by the touch controllers without any software noise estimation and removal. Section 9 provides a discussion of the obtained results, outlines the limitations of the proposed solution, and suggests potential future extensions.

8.1 Touch Sensing Prototypes

We tested the proposed algorithm on 3D models with different sizes and features. Figure 12 displays the intermediate results of the various steps of the algorithm, illustrating the progression from the initial triangle mesh to the final model ready for printing and sensorization. Table 1 contains data regarding each processing step. We always use the same parameters for all the results in the paper. In the proxy quad mesh computation, we used the isometry weight 0.3 (defined in [Pietroni et al. 2021]) to guarantee a uniform distribution of Tx-Rx intersection on the surface and an average edge length closer to 7mm. In the patch decomposition and packing, we use the footprint of the Muca board (21 Tx and 12 Rx). In the generation of the final model, we used a wire diameter $d_{\text{wire}} = 0.5$mm and a pipe diameter $d_{\text{pipe}} = 3d_{\text{wire}}$ to facilitate a smooth pulling of conductors inside the prototype. For the models Bunny, Max Planck, Sphere A, Sphere B, and Cube, we fabricated the relative sensorized prototypes, as shown in Figure 1. The accompanying video in the supplemental materials demonstrates real-time interactions with

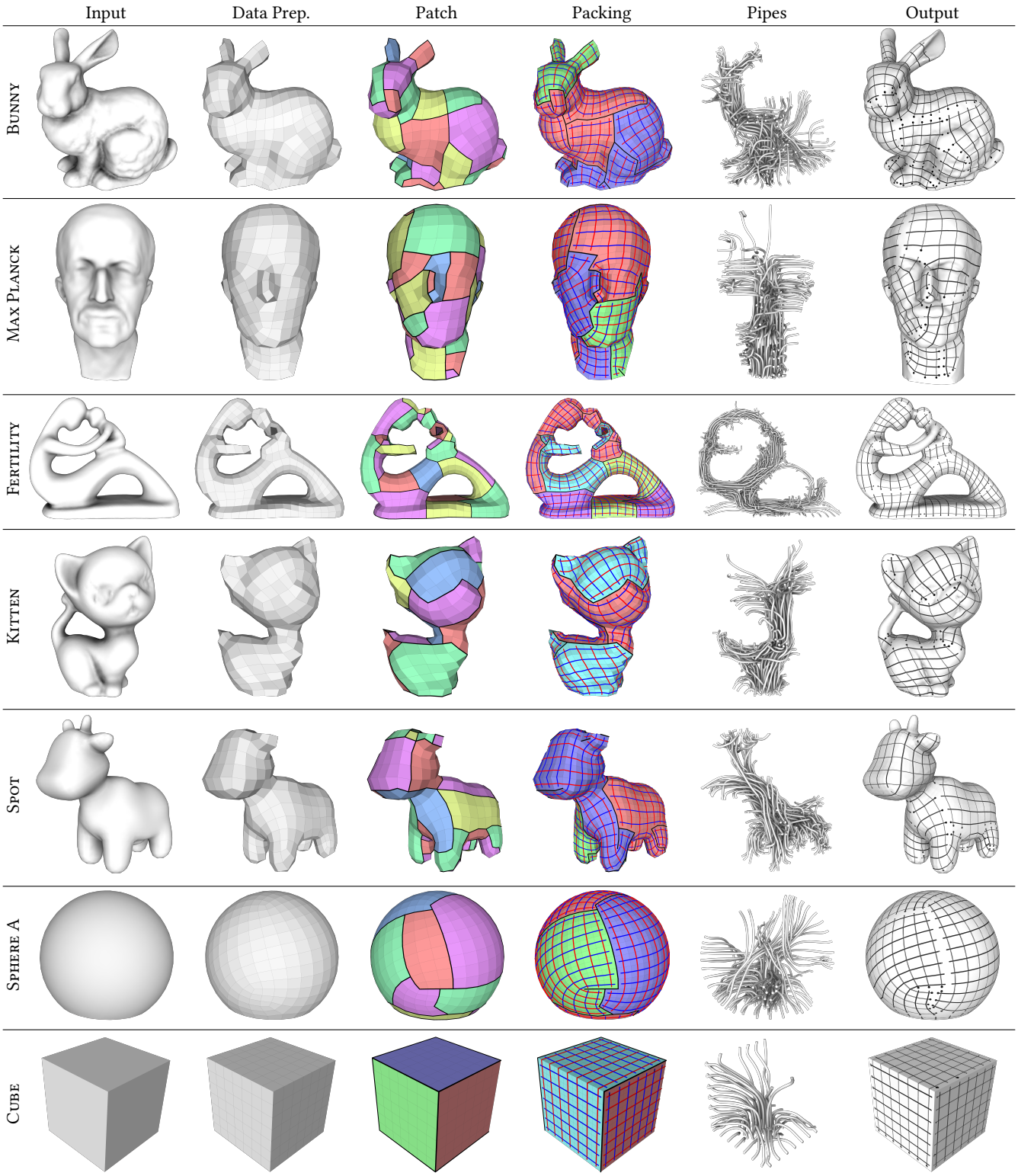


Fig. 12. Results for each input model through the various steps of the algorithm. From left to right: the input triangle mesh, the proxy quad mesh, the patch decomposition of the quad mesh, the packing of the patch decomposition along with the corresponding sensor grid mesh, the internal pipes generated to connect the sensors on the surface with the controllers, and the final mesh ready to be printed and sensorized.

Table 1. Processing data for the tested model. For each model, the table includes: the number of triangles in the input mesh (Size Tri) and the number of quads in its proxy geometry (Size Quad); the size of the bounding box in millimeters; statistics on the distances between adjacent intersections in the sensor grid (mean, standard deviation, minimum, and maximum distance); the number of patches in the decomposition after Motorcycle Graph computation (M.G.), ILP graph reduction (ILP), and patch geometric optimization (GOPT); the count of geometric optimizations, including *Edge Swap* (S), *Merge* (M), *Asymmetric Split&Merge* (ASM), and *Symmetric Split&Merge* (SSM); the number of clusters; the count of used touch controllers after cluster packing with the relative occupancy term; the number of internal pipes generated, reported as exit-pipes (Exit) and intra-pipes (Intra); and the meters of unipolar copper wires used to sensorize the 3D printed prototype on the surface (Surf.) and in the internal pipes (Pipes).

Model	**Mesh size**	**BBox size**	**Intersection dist.**	**N. Patches**	**N. Geom. Opt.**	**N. Clusters**	**Controllers**	**N. Pipes**	**Conductors**
	Tri / Quad	*mm*	μ (σ) [min, max]	M.G. / ILP / GOPT	S / M / ASM / SSM		N. / Occup.	Exit / Intra	Surf. / Pipes *m*
Bunny	56K / 450	120 × 118 × 92	7.55 (1.34) [3.8, 12.84]	90 / 33 / 31	7 / 1 / 2 / 1	7	3 / 0.6	86 / 64	6.56 / 10.14
Max Planck	54k / 420	70 × 120 × 90	6.98 (1.33) [3.21, 10.02]	81 / 34 / 31	1 / 2 / 3 / 2	4	3 / 0.56	88 / 31	5.66 / 8.25
Fertility	146k / 664	150 × 115 × 59	6.44 (1.28) [3.26, 12.45]	118 / 42 / 37	0 / 4 / 1 / 1	10	4 / 0.66	120 / 102	8.23 / 17.7
Kitten	37k / 326	65 × 100 × 59	6.66 (1.42) [2.37, 10.39]	60 / 22 / 18	7 / 3 / 0 / 1	4	2 / 0.65	62 / 28	4.26 / 5.16
Spot	5.8k / 445	60 × 108 × 110	6.68 (1.33) [3.2, 9.83]	120 / 41 / 38	5 / 3 / 5 / 0	5	3 / 0.59	90 / 62	5.75 / 9.57
Sphere A	1218 / 447	100 × 85 × 100	7.61 (1.17) [3.99, 12.15]	24 / 8 / 8	1 / 0 / 0 / 0	5	3 / 0.59	83 / 27	6.46 / 7.33
Sphere B	1218 / 149	50 × 42.5 × 50	6.51 (0.82) [4.51, 9.07]	18 / 8 / 6	3 / 0 / 1 / 2	2	1 / 0.59	30 / 15	1.84 / 1.44
Cube	12 / 245	50 × 50 × 50	7.14 (0.0) [7.14, 7,14]	5 / 2 / 2	0 / 0 / 0 / 0	2	2 / 0.49	49 / 0	3.32 / 2.11

Table 2. Time processing data for the tested models. For each model, the table includes the time needed for the mesh preparation in Section 4 (Prep.), the patch decomposition and packing in Section 5 (Patch & Packing), the generation of surface grooves and internal pipes in Section 6 (Grooves & Pipes), and the time to compute boolean operations to create the final printable model (Booleans).

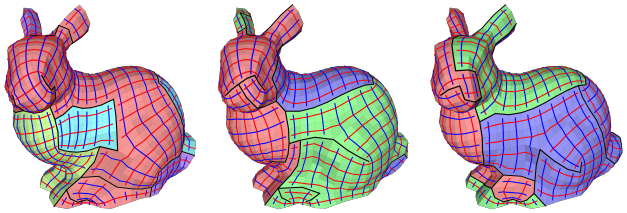
Model	**Prep.**	**Patch & Packing**	**Grooves & Pipes**	**Booleans**
	s	*s*	*s*	*s*
Bunny	419	4.7	360	752
Max Planck	416	3.3	209	490
Fertility	426	6.6	392	2060
Kitten	354	3.5	143	415
Spot	412	2.2	222	668
Sphere A	7	0.9	279	460
Sphere B	6	0.7	34	80
Cube	8	0.6	62	150

these prototypes during different sessions. Figure 1 provides an example of interaction using two fingers with the Bunny prototype. Table 2 reports the processing time of the main steps of the algorithms, where most of the time is taken by the quad meshing in the data preparation and the boolean operations for generating the prototype to print. Computations were performed on a laptop with an Intel(R) Core(TM) i9-9980HK CPU clocked at 2.40GHz and 32 GB of RAM. We used [Gurobi Optimization 2018] as ILP solver, libIGL for computing geometric booleans [Zhou et al. 2016], and OpenVDB for offset surface extraction [Museth 2013].

8.2 Patch Decomposition and Packing Performance

We assessed the robustness of the patch decomposition and packing algorithm from two different perspectives. In the first aspect, we conducted tests using various quad meshes of the same model, increasing the isometry weight (Figure 13). The results suggest that a higher isometry ensures a more uniform distribution of intersection points on the surface, leading to improved packing performance in terms of the number of touch controllers and internal pipes.

For the second aspect, we evaluated the improvements introduced by the quad patch layout simplification procedure presented in Section 5.1. The patch decomposition and packing procedures consist of four different steps: motorcycle graph computation (MG), ILP graph reduction (ILP), patch geometric optimization (GOPT),



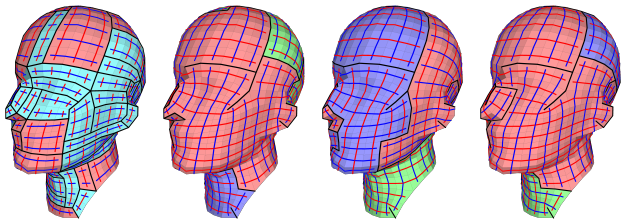
Isometry	**N.quad**	**Intersection dist.**	**N.controllers**	**N.pipes**
0.01	534	6.95 (1.65)[1.67, 13.15]	4 / 0.53	170 (107/63)
0.1	487	7.26 (1.41)[2.53, 11.14]	3 / 0.63	145 (95/50)
0.2	461	7.44 (1.45)[2.61, 14.39]	3 / 0.61	145 (94/51)
0.3	450	7.55 (1.34) [3.8, 12.84]	3 / 0.6	150 (86/64)

Fig. 13. Packing results using different quad meshes with increasing isometry weight: (left) 0.01; (center) 0.1; (right) 0.2. The table below presents, for each meshing, the number of quads, statistics on the distance between adjacent Tx-Rx intersections (mean, standard deviation, minimum, and maximum values), the count of used touch controllers with the relative occupancy term, and the total number of internal pipes, also reported as exit-pipes and intra-pipes. The table also includes results for the case with an isometry weight of 0.3, as presented in Figure 12 and Table 1.

and final packing (PACK). We tested three reduced pipelines by omitting the ILP and GOPT steps (MG+PACK, MG+GOPT+PACK, MG+ILP+PACK). Figure 14 reports the results of these tests, which can be compared with the complete pipeline presented in Figure 12 and Table 1. We also compare the result obtained by applying a state-of-the-art rectangular bin packing algorithm [Lodi et al. 1999] to the motorcycle graph decomposition (MG+BINPACK). The proposed pipeline can significantly reduce the number of internal pipes while slightly increasing the number of used controllers and the relative wasted spaces.

8.3 Signal-to-Noise Ratio

We conducted a systematic evaluation to assess the quality of touch sensing on general 3D surfaces. We identified the most demanding regions for touch sensing on the Bunny and Cube prototypes. These regions include those with the closest and farthest conductors from the controllers, areas near topological cuts among patches mapped on the same controller, regions located on the boundary of patches



Pipeline	N. patches	N.controllers	N. pipes
MG+BINPACK	81	2 / 0.83	369 (66/303)
MG+PACK	81	3 / 0.56	137 (88/49)
MG+GOPT+PACK	33	3 / 0.56	135 (94/41)
MG+ILP+PACK	34	3 / 0.56	132 (80/52)
MG+ILP+GOPT+PACK	31	3 / 0.56	119 (88/31)

Fig. 14. Packing results using different processing pipelines. From left to right: MG+BINPACK; MG+PACK; MG+GOPT+PACK; MG+ILP+PACK. For each test, the table below includes the number of patches before the packing procedure, the number of touch controllers used after packing with the relative occupancy term, and the total number of internal pipes reported as exit and intra-pipes. The table also presents results for the complete pipeline (MG+ILP+GOPT+PACK) as shown in Figure 12 and Table 1.

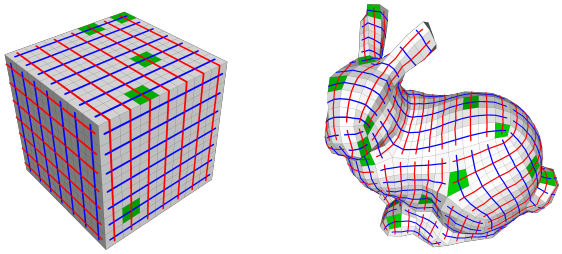


Fig. 15. The intersections on the green quads are the base points used to measure the SNR on the Cube and Bunny prototypes. A complete overview of all the measuring points is available in the supplemental materials.

mapped on different touch controllers, areas near the singularities of the mesh, and regions with sharp edges. We also included more standard, 'easy' regions in the middle of a regular patch mapped on a single controller (locally isomorphic to a regular grid) for a comprehensive comparison. For each region, we placed the relative measuring point at the close Tx-Rx line intersection. Figure 15 illustrates the distribution of some SNR measuring points for the two tested prototypes. Supplemental materials accompanying this document display the positions of each measuring point.

Sensing performance was evaluated by touching the sensor on the measuring point 10 times with the index finger at intervals of at least 1 second and computing the Signal-to-Noise Ratio (SNR) using the raw capacitance values acquired by the Muca controller. We use the formula proposed in [Davison 2010] (Equation 13) to compute the SNR, where μ_{nT} and σ_{nT} represent the mean and standard deviation values when there are no touch events, while μ_T denotes the mean value during touch contact. For robust touch sensing at an industrial level, the SNR threshold should ideally be at least 15; a minimum of 7 is absolutely required [Davison 2010].

Table 3. Signal-to-noise ratio on the Bunny and Cube prototypes under different touch conditions: (Single) single touch, (2-touches Rx) 2 touches with the second finger on the same Rx line, (2-touches Tx) 2 touches with the second finger on the same Tx line, (3-touches) 3 touches with the second and third fingers on the same Rx and Tx lines. The red text highlights tests with an SNR below the threshold of 15. The light grey background indicates tests requiring more than two trials due to SNR inconsistency.

Cube

	Single	2-touches Rx	2-touches Tx	3-touches
C1 Tx4 Rx4	62.4 / 82.7	24.5 / 27.5	113.9 / 141.0	35.7 / 48.3
C1 Tx0 Rx6	55.5 / 36.9	11.3 / 11.8	29.9 / 32.0	11.5 / 13.3
C1 Tx0 Rx4	47.2 / 55.5	14.6 / 15.1	49.8 / 54.5	18.5 / 37.9
C1 Tx12 Rx1	38.2 / 48.9	13.1 / 15.4	36.6 / 43.8	16.0 / 18.4
C1 Tx6 Rx2	40.6 / 54.3	12.7 / 17.4	28.5 / 37.8	19.6 / 29.8

Bunny

	Single	2-touches Rx	2-touches Tx	3-touches
C0 Tx19 Rx11	38.0 / 52.3	18.5 / 19.9	36.3 / 38.1	13.0 / 25.8
C0 Tx3 Rx10	58.4 / 66.8	23.7 / 28.8	32.4 / 36.8	39.1 / 39.1
C0 Tx3 Rx3	38.6 / 43.3	18.7 / 19.7	36.0 / 47.1	22.2 / 24.8
C0 Tx7 Rx9	53.8 / 67.1	12.2 / 28.8	54.8 / 77.4	26.8 / 44.9
C0 Tx6 Rx4	66.8 / 81.1	13.7 / 18.4	126.7 / 159.5	49.9 / 106.9
C0 Tx1 Rx2	54.1 / 67.5	22.8 / 23.1	54.6 / 74.7	54.4 / 76.9
C0 Tx17 Rx9	115.1 / 124.8	23.1 / 35.5	55.0 / 65.2	35.2 / 39.5
C1 Tx20 Rx1	136.8 / 728.2	27.2 / 29.2	280.7 / 321.4	109.2 / 120.6
C1 Tx15 Rx6	26.2 / 35.1	18.1 / 19.1	27.5 / 32.2	27.0 / 49.6
C1 Tx9 Rx9	136.5 / 146.4	30.0 / 31.4	55.9 / 73.7	227.5 / 302
C1 Tx10 Rx3	55.2 / 60.3	21.4 / 38.4	52.8 / 61.6	53.1 / 54.0
C1 Tx3 Rx7	70.7 / 145.1	37.7 / 66.1	101.8 / 161.0	40.9 / 56.9
C1 Tx16 Rx3	110.7 / 111.7	48.6 / 50.1	72.3 / 77.4	56.7 / 65.7
C1 Tx5 Rx7	41.5 / 61.7	23.7 / 32.0	62.4 / 69.6	24.2 / 48.7
C2 Tx0 Rx1	63.7 / 72.1	32.1 / 32.8	52.8 / 57.1	30.9 / 34.3
C2 Tx8 Rx8	32.5 / 35.3	12.9 / 15.4	27.4 / 29.5	10.4 / 13.5
C2 Tx6 Rx5	122.8 / 129.2	64.0 / 71.1	83.3 / 92.8	86.5 / 124.6
C2 Tx4 Rx0	72.0 / 85.6	15.8 / 16.6	44.1 / 45.8	30.9 / 30.7

$$SNR = \frac{|\mu_{nT} - \mu_T|}{\sigma_{nT}} \tag{13}$$

For each measuring point, we conducted four tests with different touch conditions: single-touch, simultaneous touching with a second finger on the same TX line, touching with a second finger on the same RX line, and touching with three fingers with the second and third fingers on the same Tx and Rx lines. In the case of multi-touch input, secondary and tertiary fingers were positioned on the respective TX or RX lines at three intersections from the base measuring point. We performed each test twice. We conducted additional tests for a few points that exhibited low consistency between the SNR values of the first two trials (usually one value below 15 and the other above 15). The low consistency could be explained by slight variations in the gap distance between Tx and Rx lines due to the manual fabrication, particularly on the border of regions where conductors bend to enter inside the volume of the object.

Table 3 summarises the SNR values for all the tested points. For each test, we report the values of both measures. For low consistency points, highlighted in grey, we present the maximum and minimum values over all the trials. The data of each trial are reported in the supplemental materials.

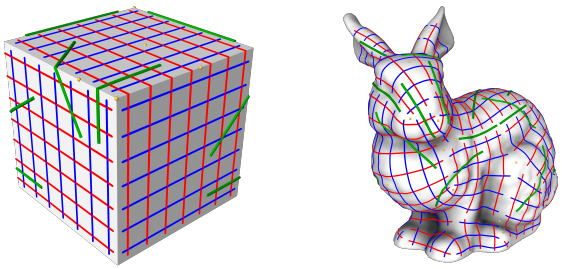


Fig. 16. The fixed paths (green lines) and points (yellow points) used to test the spatial accuracy on the Cube and Bunny prototypes. A complete overview of all the tested paths and points is available in the supplemental materials.

8.4 Spatial Accuracy

We evaluated the spatial accuracy of our approach under two distinct conditions: touching a fixed point and traversing a fixed path on the surface. Using the Bunny and Cube prototypes, we defined a set of ground truth fixed points and paths on the corresponding 3D models chosen to sample challenging regions in the 3D shape for touch sensing. Figure 16 shows some selected points and paths (see the supplemental materials for a more detailed visualization).

Fixed points were distributed across both convex and concave regions, close to the singularities of the sensor grid, on borders between controllers, on the cut between patches mapped on the same controller, along sharp edges and corners, and, for the sake of a comprehensive comparison, inside a regular patch mapped on a single controller. The paths were designed to run parallel to the Tx or Rx lines, cross diagonally over sensor lines, and follow the borders between controllers and sharp edges. We also formulated paths that span different controllers and patches, covering all possible Tx-Rx field directions (TxTx, RxRx, and TxRx). For the fixed point, the test protocol involved touching it for at least 2 seconds, recording all the positions computed, while the path involved moving the index finger along it at a constant speed, as shown in the accompanying video. Touch positions were computed throughout the tests using the procedure outlined in Section 7.1 without further processing. Small physical spherical and cylindrical reliefs were added to the surface of the sensorized prototypes during the 3D printing process to facilitate precise finger placement during the test. These reliefs provided both visual and haptic feedback. Real-time raw visualization of the computed touch points on the triangle mesh surface further enhanced feedback during the test. The reliefs used for the tests are visible in the accompanying video. Each test case included five trials, calculating the distance error of each recorded touch point from the ground truth. We compute the statistics of the distance error over all the touched points of each trial independently (mean errors, min and max values, mean absolute deviation, standard deviation, variance, root mean square error). Tables 4 and 5 present the aggregate statistics of the error distance over all five trials in millimeters for the fixed points and the paths with respect to the ground truth data. Figure 17 displays the results of two tests. Please refer to the supplemental materials for a comprehensive overview of all the tested points, paths, and acquired trials.

Table 4. Spatial accuracy error for the tests with the touch of a fixed point. For each test case, the table reports the average error, the minimum and the maximum error, the mean absolute deviation, the standard deviation, the variance, and the RMS error in millimeters of the recorded touched positions with respect to the ground truth.

Cube

Fixed points	μ	Min	Max	MAD	σ	σ^2	RMSE
3-cornerChipBorder1	2.37	0.90	4.18	0.67	0.80	0.70	2.52
3-cornerChipBorder2	1.93	0.94	3.19	0.42	0.53	0.30	2.01
4-chipBorderSharp	0.83	0.11	2.43	0.44	0.55	0.33	0.99
4-inside	0.60	0.33	0.95	0.15	0.17	0.07	0.65
4-sharpInside	0.57	0.38	0.76	0.07	0.09	0.01	0.58

Bunny

Fixed points	μ	Min	Max	MAD	σ	σ^2	RMSE
3-chipBorder1	0.80	0.30	1.63	0.36	0.42	0.21	0.93
3-chipBorder2	0.98	0.69	1.26	0.12	0.14	0.02	0.99
3-chipBorder3	0.59	0.33	0.91	0.12	0.15	0.02	0.61
3-patchBorder1	0.70	0.20	1.05	0.22	0.25	0.07	0.75
3-patchBorder2	0.64	0.32	0.92	0.15	0.18	0.03	0.66
3-patchBorder3	0.79	0.55	1.04	0.10	0.13	0.02	0.81
4-chipBorder1	0.73	0.33	1.03	0.17	0.20	0.05	0.77
4-chipBorder2	0.57	0.39	0.78	0.08	0.10	0.01	0.58
4-chipPatchBorder	1.70	0.91	2.79	0.44	0.55	0.41	1.82
4-farthestPoint1	0.37	0.17	0.65	0.10	0.13	0.02	0.40
4-farthestPoint2	0.50	0.31	0.68	0.08	0.10	0.01	0.52
4-inside1	0.59	0.41	0.78	0.08	0.10	0.01	0.60
4-inside2	0.91	0.59	1.27	0.16	0.20	0.04	0.94
4-inside3	0.83	0.43	1.21	0.19	0.23	0.06	0.87
4-inside4	1.44	1.11	1.83	0.17	0.21	0.05	1.46
4-inside5	0.53	0.23	0.90	0.17	0.20	0.05	0.57
4-patchBorder1	0.56	0.28	0.90	0.16	0.19	0.04	0.59
4-patchBorder2	0.82	0.46	1.36	0.21	0.27	0.08	0.87
4-patchBorder3	0.37	0.14	0.61	0.12	0.14	0.02	0.40
4-patchBorder4	0.62	0.36	0.96	0.15	0.18	0.04	0.65
4-patchBorder5	0.62	0.14	1.29	0.27	0.34	0.13	0.71
5-chipBorder1	1.06	0.41	1.69	0.25	0.31	0.10	1.10
5-chipBorder2	1.14	0.26	1.93	0.40	0.48	0.24	1.24
5-chipBorder3	0.86	0.45	1.18	0.16	0.20	0.04	0.89
5-patchBorder1	0.57	0.20	0.96	0.18	0.22	0.05	0.61
5-patchBorder2	0.45	0.20	0.71	0.13	0.15	0.02	0.48

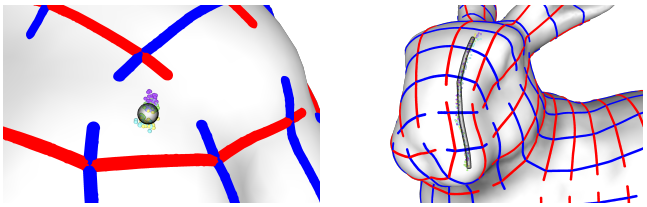


Fig. 17. Visual results of the spatial accuracy tests: (left) touch of a single point with mean error 0.55mm (RMS 0.61); (right) touch during the movement along a path with mean error 0.56mm (RMS 0.71). The colored spheres show the touch points acquired during the five trials. Each trial has a different color. The semi-transparent black geometry shows the ground truth position of each test.

Table 5. Spatial accuracy error for the tests with the movement of the finger along a path. For each test case, the table reports the average error, the minimum and the maximum error, the mean absolute deviation, the standard deviation, the variance, and the RMS error in millimeters of the touched positions along the path with respect to the ground truth.

CUBE

Paths	μ	Min	Max	MAD	σ	σ^2	RMSE
chipBorderSharpTxRx	1.15	0.02	3.77	0.60	0.76	0.60	1.38
chipBorderSharpTxTx	1.00	0.02	3.24	0.58	0.72	0.52	1.23
chipCrossDiagTxRx	1.30	0.04	3.82	0.88	1.00	1.01	1.64
chipCrossDiagTxTx	1.40	0.03	5.19	0.88	1.13	1.31	1.80
chipCrossParallTxTx	1.36	0.04	3.16	0.63	0.78	0.64	1.57
chipCrossParallTxRx	0.99	0.03	5.32	0.60	0.90	0.81	1.35
insideDiagCrossSharp	0.93	0.01	2.94	0.62	0.75	0.63	1.20
insideParallTx	0.59	0.02	2.01	0.40	0.50	0.26	0.78
insideParallCrossSharpRx	0.83	0.00	3.62	0.64	0.80	0.70	1.16
insideParallSharpTx	0.96	0.09	2.34	0.51	0.62	0.40	1.14

BUNNY

Paths	μ	Min	Max	MAD	σ	σ^2	RMSE
chipBorderTxRx	0.99	0.09	3.25	0.46	0.63	0.42	1.19
chipBorderRxRx	1.10	0.03	3.90	0.62	0.82	0.70	1.38
chipBorderTxTx	0.92	0.06	3.75	0.49	0.66	0.45	1.14
chipCrossDiagRxRx	0.98	0.06	2.10	0.46	0.55	0.32	1.13
chipCrossDiagTxRx	1.06	0.05	3.20	0.64	0.80	0.65	1.33
chipCrossDiagTxTx	0.83	0.08	1.94	0.39	0.48	0.23	0.96
chipCrossParallRxRx	0.56	0.04	2.06	0.31	0.44	0.21	0.72
chipCrossParallTxRx	0.76	0.07	4.18	0.61	0.91	0.91	1.19
chipCrossParallTxTx	0.59	0.04	1.89	0.30	0.38	0.15	0.70
insideDiag1	1.43	0.03	3.87	0.72	0.89	0.81	1.68
insideDiag2	1.42	0.04	3.85	0.67	0.85	0.74	1.66
insideParallRx	0.91	0.01	2.77	0.51	0.62	0.40	1.11
insideParallTx	0.98	0.04	4.29	0.71	0.96	0.94	1.38
patchBorderTxRX	0.85	0.07	2.43	0.48	0.60	0.43	1.04
patchBorderRxRx	0.85	0.02	2.98	0.50	0.67	0.45	1.08
patchBorderTxTx	0.90	0.03	3.06	0.57	0.73	0.62	1.17
patchCrossDiagRxRx	0.66	0.03	2.07	0.34	0.45	0.20	0.80
patchCrossDiagTxRx	0.77	0.02	2.03	0.43	0.53	0.28	0.93
patchCrossParallRxRx	1.36	0.06	4.77	0.92	1.17	1.44	1.80
patchCrossParallTxTx	1.40	0.10	3.49	0.54	0.69	0.49	1.57
patchCrossParallRxTx	0.48	0.01	2.19	0.34	0.44	0.20	0.65

9 Discussion

The quality estimation on the fabricated prototypes, reported in Table 3 for SNR and in Tables 4 and 5 for spatial accuracy, demonstrates precise and robust touch detection. It is further confirmed by the interaction sessions in the accompanying video, which show stable and accurate multi-touch sensing.

SNR Analysis. The estimated SNR consistently remains above the threshold of 15 [Davison 2010] for robust touch sensing at an industrial level, with very few cases having values close to or slightly below this threshold. The sensors exhibit high robustness in single-touch (Single) and double-touch scenarios with a second finger on the same Tx line (2-touches TX). Analysis of the raw data reported in the supplemental materials confirms the expected decrease in capacitive values during multi-touch scenarios. Few critical tests occur in touch conditions with a second finger on the same Rx line (2-touches Rx and 3-touches). The general trend is to achieve better SNR with 3 touches compared to the configuration with only a second finger on the same Rx line. In the 3-touch configuration, the finger on the Tx line contributes to a reduction in the standard deviation of the background noise during the no-touch event, leading to increased SNR values. In future works, an extensive study of this behavior is suggested, which could be attributed to hardware and fabrication aspects such as grounding issues, small vibrations of conductors on the surface during touches, and electromagnetic interference of the wires inside the internal pipes with the sensors on the surface. For example, the automatic creation of prototypes using dual-material 3D printing, as reported in Section 7, could serve as a solution to make SNR evaluation free from these fabrication issues.

Spatial Accuracy Analysis. The tests indicate an average error of 0.84mm for fixed points (RMSE 0.88 mm) and 0.97mm for paths (RMSE 1.22mm), with only two tests showing error values above or close to 2mm. Analyzing the results, we do not observe any accuracy degradation dependent on the mapping of sensors from different controllers on the surface. The sole factor contributing to degradation is the extreme curvature configuration of the surface. In the CUBE prototype, the most critical regions are the corners and paths that run along or cross a sharp edge. Lower accuracy performances in these cases are attributed to the suboptimal positioning of sensor intersections, which are too far from the touch point on the sharp feature (~5mm), hindering the detection of capacitive variation with sufficient strength. The geometry around these features also prevents placing the finger on the surrounding sensor intersections with a sufficient footprint to register reliable capacitive variation. Future solutions may involve forcing the placement of sensor conductors and intersections on the main feature lines of the geometry using dual quad meshing approaches to overcome this limitation. In the BUNNY prototype, the most critical touch points are in highly concave regions, where the occlusion of surrounding geometry makes physically touching these areas more challenging without unintentionally touching other regions of the prototype.

Future Work. The excellent touch sensing results open new avenues for future work to enhance and expand the fabrication design of sensors and the subsequent software layer for data interpolations. Specifically, our method can be easily extended to enable the automatic fabrication of the physical prototype using more advanced technologies for the simultaneous printing of dielectric and conductive materials. This extension reduces the fabrication time of the prototypes, especially avoiding the manual pulling procedure of the conductors. For example, for the BUNNY, we spent 31 hours for the 3D printing by material jetting, 4 hours for the cleaning of the support material, and 24 hours (not continuous) for the manual wire pulling. Additionally, we can generalize our approach for creating sensors with a diamond pattern using mid-point subdivision algorithms for quad meshing. On the hardware side, another possible extension is defining a standard pin layout at the bottom of the prototype to enable the rapid connection of different objects to the same touch-sensing board. It involves forcing the internal pipe generation to assign each touch controller line to the same position in the pin layout across the different objects. More advanced solutions can also be based on selecting multiple exit regions for the conductors to allow a more flexible generation of the internal

pipes. Finally, we can explore approaches based on machine learning for the interpolation procedure of the raw sensor data. These approaches could be more robust against background noise and the curvature of the geometry, enabling a more precise computation of touch points in highly concave regions.

10 Conclusion

We present a novel computational fabrication solution to enable mutual capacitive multi-touch sensing on a general 3D object by automatically generating a regular sensor grid on its surface. The method aims to distribute touch sensing points (intersections of the sensor grid) as uniformly as possible, minimizing the number of used touch controllers and exit conductors needed to connect to these controllers. Our proposed method is based on a new patch decomposition and packing approach for quad patch layouts derived from quad meshing of the input triangle mesh. It is followed by a procedure to compute the final model, preparing it for 3D printing while defining a robust solution for generating internal pipes to connect the sensor grid on the surface with the touch controllers placed outside the object. The 3D printed prototype is equipped with sensor conductors through a manual procedure and connected to touch-sensing hardware to enable the computation of touch points in the continuous space defined by the triangle mesh surface. Tests on the touch sensing quality, conducted on physical prototypes, demonstrate a robust multi-touch detection with high SNR values and spatial accuracy. The average touch position error over all the tests falls in the 0.37 - 2.37 mm range. Areas with the worst SNR and spatial accuracy are sharp features and highly concave regions where better sensor placement and advanced interpolation procedures could further improve the results.

Acknowledgments

The authors thank Alessandro Muntoni for the help with the material jetting 3D printing. The models used in the paper are courtesy of the Stanford 3D Scanning Repository, the AIM@SHAPE Shape Repository, and Keenan Crane. This research was partially funded by the German Research Foundation (DFG project 425869111 within the Priority Program SPP 2199 Scalable Interaction Paradigms for Pervasive Computing Environments).

References

Moritz Bächer, Benjamin Hepp, Fabrizio Pece, Paul G. Kry, Bernd Bickel, Bernhard Thomaszewski, and Otmar Hilliges. 2016. DefSense: Computational Design of Customized Deformable Input Devices. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (San Jose, California, USA). ACM, New York, USA, 3806–3816. https://doi.org/10.1145/2858036.2858354

Gary Barrett and Ryomei Omote. 2010. Projected-capacitive touch technology. *Information Display* 26, 3 (2010), 16–21.

Jesse Burstyn, Nicholas Fellion, Paul Strohmeier, and Roel Vertegaal. 2015. Printput: Resistive and capacitive input widgets for interactive 3D prints. In *IFIP Conference on Human-Computer Interaction*. Springer International Publishing, Cham, 332–339. https://doi.org/10.1007/978-3-319-22701-6_25

Marcel Campen. 2017. Partitioning Surfaces Into Quadrilateral Patches: A Survey. *Computer Graphics Forum* 36, 8 (2017), 567–588. https://doi.org/10.1111/cgf.13153

Marcel Campen, David Bommes, and Leif Kobbelt. 2015. Quantized Global Parametrization. *ACM Trans. Graph.* 34, 6, Article 192 (nov 2015), 12 pages. https://doi.org/10.1145/2816795.2818140

Giorgio Cannata, Marco Maggiali, Giorgio Metta, and Giulio Sandini. 2008. An embedded artificial skin for humanoid robots. In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*. IEEE, USA, 434–438. https://doi.org/10.1109/MFI.2008.4648033

Tingyu Cheng, Koya Narumi, Youngwook Do, Yang Zhang, Tung D. Ta, Takuya Sasatani, Eric Markvicka, Yoshihiro Kawahara, Lining Yao, Gregory D. Abowd, and HyunJoo Oh. 2020. Silver Tape: Inkjet-Printed Circuits Peeled-and-Transferred on Versatile Substrates. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 4, 1 (March 2020), 1–17. https://doi.org/10.1145/3381013

Burke Davison. 2010. *Techniques for robust touch sensing design*. Technical Report. AN1334 Microchip Technology Inc. 53 pages.

David Eppstein, Michael T. Goodrich, Ethan Kim, and Rasmus Tamstorf. 2008. Motorcycle Graphs: Canonical Quad Mesh Partitioning. *Computer Graphics Forum* 27, 5 (2008), 1477–1486. https://doi.org/10.1111/j.1467-8659.2008.01288.x

Jun Gong, Olivia Seow, Cedric Honnet, Jack Forman, and Stefanie Mueller. 2021. MetaSense: Integrating Sensing Capabilities into Mechanical Metamaterial. In *The 34th Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 1063–1073. https://doi.org/10.1145/3472749.3474806

Timo Götzelmann and Christopher Althaus. 2016. TouchSurfaceModels: Capacitive Sensing Objects through 3D Printers. In *Proceedings of the 9th ACM International Conference on PErvasive Technologies Related to Assistive Environments*. ACM, New York, NY, USA, 1–8. https://doi.org/10.1145/2910674.2910690

Tony Gray. 2019. *Projected Capacitive Touch*. Springer International Publishing, Cham. https://doi.org/10.1007/978-3-319-98392-9

Daniel Groeger and Jürgen Steimle. 2018. ObjectSkin: Augmenting Everyday Objects with Hydroprinted Touch Sensors and Displays. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies* 1, 4 (Jan. 2018), 1–23. https://doi.org/10.1145/3161165

Tobias Grosse-Puppendahl, Yannick Berghoefer, Andreas Braun, Raphael Wimmer, and Arjan Kuijper. 2013. OpenCapSense: A rapid prototyping toolkit for pervasive interaction using capacitive sensing. In *IEEE International Conference on Pervasive Computing and Communications (PerCom)*. IEEE Computer Society, Washington, DC, USA, 152–159. https://doi.org/10.1109/PerCom.2013.6526726

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 SIGCHI Conference on Human Factors in Computing Systems* (Denver, USA). ACM, New York, NY, USA, 3293–3315. https://doi.org/10.1145/3025453.3025808

LLC Gurobi Optimization. 2018. Gurobi Optimizer Reference Manual. http://www.gurobi.com

Jefferson Y. Han. 2005. Low-cost Multi-touch Sensing Through Frustrated Total Internal Reflection. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology* (Seattle, WA, USA). ACM, New York, NY, USA, 115–118. https://doi.org/10.1145/1095034.1095054

Chris Harrison, Hrvoje Benko, and Andrew D. Wilson. 2011a. OmniTouch: Wearable Multitouch Interaction Everywhere. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA). ACM, New York, NY, USA, 441–450. https://doi.org/10.1145/2047196.2047255

Chris Harrison, Julia Schwarz, and Scott E. Hudson. 2011b. TapSense: enhancing finger interaction on touch surfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM, New York, NY, USA, 627. https://doi.org/10.1145/2047196.2047279

Liang He, Jarrid A. Wittkopf, Ji Won Jun, Kris Erickson, and Rafael Tico Ballagas. 2022. ModElec: A Design Tool for Prototyping Physical Computing Devices Using Conductive 3D Printing. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 5, 4, Article 159 (dec 2022), 20 pages. https://doi.org/10.1145/3495000

Freddie Hong, Connor Myant, and David E Boyle. 2021. Thermoformed Circuit Boards: Fabrication of Highly Conductive Freeform 3D Printed Circuit Boards with Heat Bending. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan). ACM, New York, NY, USA, Article 669, 10 pages. https://doi.org/10.1145/3411764.3445469

Yoshihiro Kawahara, Steve Hodges, Benjamin S. Cook, Cheng Zhang, and Gregory D. Abowd. 2013. Instant Inkjet Circuits: Lab-Based Inkjet Printing to Support Rapid Prototyping of UbiComp Devices. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Zurich, Switzerland). ACM, New York, NY, USA, 363–372. https://doi.org/10.1145/2493432.2493486

Arshad Khan, Joan Sol Roo, Tobias Kraus, and Jürgen Steimle. 2019. Soft Inkjet Circuits: Rapid Multi-Material Fabrication of Soft Circuits Using a Commodity Inkjet Printer. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA). ACM, New York, NY, USA, 341–354. https://doi.org/10.1145/3332165.3347892

Konstantin Klamka, Raimund Dachselt, and Jürgen Steimle. 2020. Rapid Iron-On User Interfaces: Hands-on Fabrication of Interactive Textile Prototypes. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA). ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376220

Gierad Laput, Eric Brockmeyer, Scott E. Hudson, and Chris Harrison. 2015. Acoustruments: Passive, Acoustically-Driven, Interactive Controls for Handheld Devices. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems* (Seoul, Republic of Korea). ACM, New York, NY, USA, 2161–2170.

https://doi.org/10.1145/2702123.2702414

Andrea Lodi, Silvano Martello, and Michele Monaci. 2002. Two-dimensional packing problems: A survey. *European Journal of Operational Research* 141, 2 (2002), 241–252. https://doi.org/10.1016/S0377-2217(02)00123-6

Andrea Lodi, Silvano Martello, and Daniele Vigo. 1999. Heuristic and Metaheuristic Approaches for a Class of Two-Dimensional Bin Packing Problems. *INFORMS Journal on Computing* 11, 4 (1999), 345–357. https://doi.org/10.1287/ijoc.11.4.345

James J. Fitzgibbon Michael C. Brenner. 1985. Surface acoustic wave touch panel system. US Patent US4644100A.

Microchip. 2012. Sensor Design Guidelines. Retrieved April 25, 2024 from http://ww1.microchip.com/downloads/en/DeviceDoc/FAQs%20-%20Sensor%20Design%20Guidelines.pdf.

Toshiharu Mukai, Masaki Onishi, Tadashi Odashima, Shinya Hirano, and Zhiwei Luo. 2008. Development of the Tactile Sensor System of a Human-Interactive Robot "RI-MAN". *IEEE Transactions on Robotics* 24, 2 (2008), 505–512. https://doi.org/10.1109/TRO.2008.917006

Ken Museth. 2013. VDB: High-resolution sparse volumes with dynamic topology. *ACM Trans. Graph.* 32, 3, Article 27 (jul 2013), 22 pages. https://doi.org/10.1145/2487228.2487235

Ashish Myles, Nico Pietroni, Denis Kovacs, and Denis Zorin. 2010. Feature-Aligned T-Meshes. *ACM Trans. Graph.* 29, 4, Article 117 (jul 2010), 11 pages. https://doi.org/10.1145/1778765.1778854

Simon Olberding, Michael Wessely, and Jürgen Steimle. 2014. PrintScreen: fabricating highly customizable thin-film touch-displays. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*. ACM, Honolulu, Hawaii, USA, 281–290. https://doi.org/10.1145/2642918.2647413

Gianpaolo Palma, Sara Perry, and Paolo Cignoni. 2021. Augmented Virtuality Using Touch-Sensitive 3D-Printed Objects. *Remote Sensing* 13, 11 (2021), 20 pages. https://doi.org/10.3390/rs13112186

Daniele Panozzo, Enrico Puppo, Marco Tarini, Nico Pietroni, and Paolo Cignoni. 2011. Automatic Construction of Quad-Based Subdivision Surfaces Using Fitmaps. *IEEE Transactions on Visualization and Computer Graphics* 17, 10 (oct 2011), 1510–1520. https://doi.org/10.1109/TVCG.2011.28

Thiago Pereira, Szymon Rusinkiewicz, and Wojciech Matusik. 2014. Computational Light Routing: 3D Printed Optical Fibers for Sensing and Display. *ACM Trans. Graph.* 33, 3, Article 24 (jun 2014), 13 pages. https://doi.org/10.1145/2602140

Nico Pietroni, Stefano Nuvoli, Thomas Alderighi, Paolo Cignoni, and Marco Tarini. 2021. Reliable Feature-Line Driven Quad-Remeshing. *ACM Trans. Graph.* 40, 4, Article 155 (jul 2021), 17 pages. https://doi.org/10.1145/3450626.3459941

Nico Pietroni, Enrico Puppo, Giorgio Marcias, Roberto Roberto, and Paolo Cignoni. 2016. Tracing Field-Coherent Quad Layouts. *Computer Graphics Forum* 35, 7 (oct 2016), 485–496. https://doi.org/10.1111/cgf.13045

Narjes Pourjafarian, Marion Koelle, Fjolla Mjaku, Paul Strohmeier, and Jürgen Steimle. 2022. Print-A-Sketch: A Handheld Printer for Physical Sketching of Circuits and Sensors on Everyday Surfaces. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA). ACM, New York, NY, USA, Article 270, 17 pages. https://doi.org/10.1145/3491102.3502074

Faniry H. Razafindrazaka and Konrad Polthier. 2017. Optimal base complexes for quadrilateral meshes. *Computer Aided Geometric Design* 52-53 (2017), 63–74. https://doi.org/10.1016/j.cagd.2017.02.012

Faniry H. Razafindrazaka, Ulrich Reitebuch, and Konrad Polthier. 2015. Perfect Matching Quad Layouts for Manifold Meshes. *Computer Graphics Forum* 34, 5 (2015), 219–228. https://doi.org/10.1111/cgf.12710

Munehiko Sato, Ivan Poupyrev, and Chris Harrison. 2012. Touché: Enhancing Touch Interaction on Humans, Screens, Liquids, and Everyday Objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Austin, Texas, USA). ACM, New York, NY, USA, 483–492. https://doi.org/10.1145/2207676.2207743

Valkyrie Savage, Ryan Schmidt, Tovi Grossman, George Fitzmaurice, and Björn Hartmann. 2014. A series of tubes: adding interactivity to 3D prints using internal pipes. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology* (Honolulu, Hawaii, USA). ACM, New York, NY, USA, 3–12. https://doi.org/10.1145/2642918.2647374

Nico Schertler, Daniele Panozzo, Stefan Gumhold, and Marco Tarini. 2018. Generalized Motorcycle Graphs for Imperfect Quad-Dominant Meshes. *ACM Trans. Graph.* 37, 4, Article 155 (jul 2018), 16 pages. https://doi.org/10.1145/3197517.3201389

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* (Charlotte, NC, USA). ACM, New York, NY, USA, 253–258. https://doi.org/10.1145/2807442.2807503

Martin Schmitz, Jürgen Steimle, Jochen Huber, Niloofar Dezfuli, and Max Mühlhäuser. 2017. Flexibles: Deformation-Aware 3D-Printed Tangibles for Capacitive Touchscreens. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA). ACM, New York, NY, USA, 1001–1014. https://doi.org/10.1145/3025453.3025663

Martin Schmitz, Martin Stitz, Florian Müller, Markus Funk, and Max Mühlhäuser. 2019. Trilaterate: A Fabrication Pipeline to Design and 3D Print Hover-, Touch-, and Force-Sensitive Objects. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk). ACM, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300684

Paul Strohmeier, Jarrod Knibbe, Sebastian Boring, and Kasper Hornbæk. 2018. zPatch: Hybrid Resistive/Capacitive eTextile Input. In *Proceedings of the Twelfth International Conference on Tangible, Embedded, and Embodied Interaction* (Stockholm, Sweden). ACM, New York, NY, USA, 188–198. https://doi.org/10.1145/3173225.3173242

Mathias Sundholm, Jingyuan Cheng, Bo Zhou, Akash Sethi, and Paul Lukowicz. 2014. Smart-Mat: Recognizing and Counting Gym Exercises with Low-Cost Resistive Pressure Sensing Matrix. In *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Seattle, Washington). ACM, New York, NY, USA, 373–382. https://doi.org/10.1145/2632048.2636088

Marco Tarini, Enrico Puppo, Daniele Panozzo, Nico Pietroni, and Paolo Cignoni. 2011. Simple Quad Domains for Field Aligned Mesh Parametrization. *ACM Trans. Graph.* 30, 6 (dec 2011), 1–12. https://doi.org/10.1145/2070781.2024176

Marc Teyssier, Brice Parilusyan, Anne Roudaut, and Jürgen Steimle. 2021. Human-Like Artificial Skin Sensor for Physical Human-Robot Interaction. In *IEEE International Conference on Robotics and Automation*. IEEE, USA, 3626–3633. https://doi.org/10.1109/ICRA48506.2021.9561152

Tito Pradhono Tomo, Massimo Regoli, Alexander Schmitz, Lorenzo Natale, Harris Kristanto, Sophon Somlor, Lorenzo Jamone, Giorgio Metta, and Shigeki Sugano. 2018. A New Silicone Structure for uSkin—A Soft, Distributed, Digital 3-Axis Skin Sensor and Its Integration on the Humanoid Robot iCub. *IEEE Robotics and Automation Letters* 3, 3 (2018), 2584–2591. https://doi.org/10.1109/LRA.2018.2812915

Daiki Tone, Daisuke Iwai, Shinsaku Hiura, and Kosuke Sato. 2020. FibAR: Embedding Optical Fibers in 3D Printed Objects for Active Markers in Dynamic Projection Mapping. *IEEE Transactions on Visualization and Computer Graphics* 26, 5 (2020), 2030–2040. https://doi.org/10.1109/TVCG.2020.2973444

Nobuyuki Umetani and Ryan Schmidt. 2017. SurfCuit: Surface-Mounted Circuits on 3D Prints. *IEEE Computer Graphics and Applications* 37, 3 (2017), 52–60. https://doi.org/10.1109/MCG.2017.40

Guanyun Wang, Fang Qin, Haolin Liu, Ye Tao, Yang Zhang, Yongjie Jessica Zhang, and Lining Yao. 2020. MorphingCircuit: An Integrated Design, Simulation, and Fabrication Workflow for Self-Morphing Electronics. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 4, 4, Article 157 (dec 2020), 26 pages. https://doi.org/10.1145/3432232

Tianyi Wang, Ke Huo, Pratik Chawla, Guiming Chen, Siddharth Banerjee, and Karthik Ramani. 2018. Plain2Fun: Augmenting Ordinary Objects with Interactive Functions by Auto-Fabricating Surface Painted Circuits. In *Proceedings of the Designing Interactive Systems Conference* (Hong Kong, China). ACM, New York, NY, USA, 1095–1106. https://doi.org/10.1145/3196709.3196791

Michael Wessely, Ticha Sethapakdi, Carlos Castillo, Jackson C. Snowden, Ollie Hanton, Isabel P. S. Qamar, Mike Fraser, Anne Roudaut, and Stefanie Mueller. 2020. Sprayable User Interfaces: Prototyping Large-Scale Interactive Surfaces with Sensors and Displays. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu HI USA, 1–12. https://doi.org/10.1145/3313831.3376249

Karl Willis, Eric Brockmeyer, Scott Hudson, and Ivan Poupyrev. 2012. Printed optics: 3D printing of embedded optical elements for interactive devices. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology* (Cambridge, Massachusetts, USA). ACM, New York, NY, USA, 589–598. https://doi.org/10.1145/2380116.2380190

Andrew D. Wilson. 2010. Using a depth camera as a touch sensor. In *ACM International Conference on Interactive Tabletops and Surfaces* (Saarbrücken, Germany). ACM, New York, NY, USA, 69–72. https://doi.org/10.1145/1936652.1936665

Raphael Wimmer and Patrick Baudisch. 2011. Modular and Deformable Touch-sensitive Surfaces Based on Time Domain Reflectometry. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA). ACM, New York, NY, USA, 517–526. https://doi.org/10.1145/2047196.2047264

Jarrid A. Wittkopf, Kris Erickson, Paul Olumbummo, Aja Hartman, Howard Tom, and Lihua Zhao. 2019. 3D Printed Electronics with Multi Jet Fusion. *NIP & Digital Fabrication Conference* 35, 1 (2019), 29–33. https://doi.org/10.2352/ISSN.2169-4451.2019.35.29

Kui Wu, Marco Tarini, Cem Yuksel, James McCann, and Xifeng Gao. 2022. Wearable 3D Machine Knitting: Automatic Generation of Shaped Knit Sheets to Cover Real-World Objects. *IEEE Transactions on Visualization and Computer Graphics* 28, 9 (2022), 3180–3192. https://doi.org/10.1109/TVCG.2021.3056101

Sen Zhang, Hui Zhang, and Jun-Hai Yong. 2016. Automatic Quad Patch Layout Extraction for Quadrilateral Meshes. *Computer-Aided Design and Applications* 13, 3 (2016), 409–416. https://doi.org/10.1080/16864360.2015.1114399

Yang Zhang and Chris Harrison. 2018. Pulp Nonfiction: Low-Cost Touch Tracking for Paper. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Montreal, QC, Canada,). ACM, New York, NY, USA, 1–11. https://doi.org/10.1145/3173574.3173691

Yang Zhang, Gierad Laput, and Chris Harrison. 2017. Electrick: Low-Cost Touch Sensing Using Electric Field Tomography. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA). ACM, New York, NY, USA, 1–14. https://doi.org/10.1145/3025453.3025842

Qingnan Zhou, Eitan Grinspun, Denis Zorin, and Alec Jacobson. 2016. Mesh arrangements for solid geometry. *ACM Trans. Graph.* 35, 4, Article 39 (jul 2016), 15 pages. https://doi.org/10.1145/2897824.2925901

Junyi Zhu, Lotta-Gili Blumberg, Yunyi Zhu, Martin Nisser, Ethan Levi Carlson, Xin Wen, Kevin Shum, Jessica Ayeley Quaye, and Stefanie Mueller. 2020a. CurveBoards: Integrating Breadboards into Physical Objects to Prototype Function in the Context of Form. In *Proceedings of the CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA). ACM, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376617

Junyi Zhu, Yunyi Zhu, Jiaming Cui, Leon Cheng, Jackson Snowden, Mark Chounlakone, Michael Wessely, and Stefanie Mueller. 2020b. MorphSensor: A 3D Electronic Design Tool for Reforming Sensor Modules. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. ACM, New York, NY, USA, 541–553. https://doi.org/10.1145/3379337.3415898

Thomas G. Zimmerman, Joshua R. Smith, Joseph A. Paradiso, David Allport, and Neil Gershenfeld. 1995. Applying Electric Field Sensing to Human-computer Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA). ACM/Addison-Wesley, New York, NY, USA, 280–287. https://doi.org/10.1145/223904.223940