RobotSculptor: Artist-Directed Robotic Sculpting of Clay

Zhao Ma
ETH Zurich & Disney Research
ma@arch.ethz.ch

Simon Duenser
ETH Zurich
simon.duenser@inf.ethz.ch

Christian Schumacher
Disney Research
christian.schumacher@disney.com

Romana Rust
ETH Zurich
rust@arch.ethz.ch

Moritz Bächer
Disney Research
moritz.baecher@disney.com

Fabio Gramazio
ETH Zurich
gramazio@arch.ethz.ch

Matthias Kohler
ETH Zurich
kohler@arch.ethz.ch

Stelian Coros
ETH Zurich
scoros@inf.ethz.ch

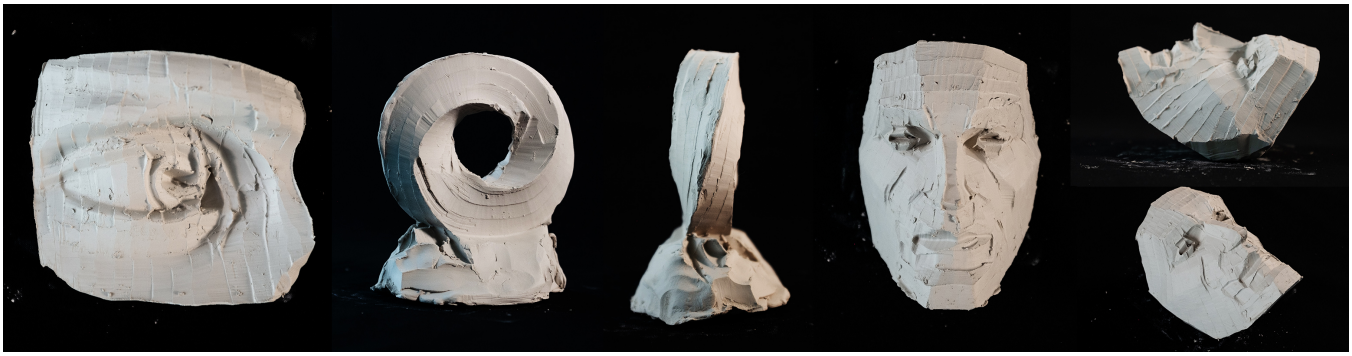

Figure 1: Examples of robot-sculpted clay models.

ABSTRACT

We present an interactive design system that allows users to create sculpting styles and fabricate clay models using a standard 6-axis robot arm. Given a general mesh as input, the user iteratively selects sub-areas of the mesh through decomposition and embeds the design expression into an initial set of toolpaths by modifying key parameters that affect the visual appearance of the sculpted surface finish. These parameters were identified and extracted through a series of design experiments, using a customized *loop tool* to cut the water-based clay material. The initialized toolpaths are fed into the optimization component of our system afterwards for optimal path planning, aiming to find the robotic sculpting motions that match the target surface, maintaining the design expression, and resolving collisions and reachability issues. We demonstrate the versatility of our approach by designing and fabricating different sculpting styles over a wide range of clay models.

SCF '20, November 5–6, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8170-3/20/11…$15.00
https://doi.org/10.1145/3424630.3425415

CCS CONCEPTS

• **Applied computing** → **Computer-aided manufacturing**; • **Computing methodologies** → **Computer graphics**.

KEYWORDS

Robotics, Digital Fabrication, Sculpting, Clay, Path Planning

ACM Reference Format:
Zhao Ma, Simon Duenser, Christian Schumacher, Romana Rust, Moritz Bächer, Fabio Gramazio, Matthias Kohler, and Stelian Coros. 2020. RobotSculptor: Artist-Directed Robotic Sculpting of Clay. In *Symposium on Computational Fabrication (SCF '20), November 5–6, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3424630.3425415

1 INTRODUCTION

Sculpture is one of the oldest forms of three-dimensional visual art. Amongst all the materials used for sculpture, clay is the most widespread and frequently used. Its malleability allows it to be formed into any shape imaginable and makes it suitable for both additive and subtractive processes. During a sculpting process, artists utilize a variety of techniques and employ different tools to reform a piece of clay until it satisfies the intent of their artistic expression.

In today's industrialized context, CNC milling is widely used for the manufacturing of three-dimensional sculptural artefacts and often substitutes traditional processes including stone carving or foam cutting. However, conventional CNC milling techniques are limited when applied to soft materials like water-based clay.

Highly ductile materials are notoriously difficult to cut mechanically, and the strong adhesive tendency of clay greatly hinders the removal of small shavings. A special technique called "cryogenic machining" uses low-temperature coolant to freeze soft or elastic materials (for instance, rubber) temporarily during the milling process [Dhokia et al. 2011], but has to our knowledge not been used for clay material.

On the other hand, human sculpting still holds a special place due to its close association with arts and crafts. Due to the non-linear nature of the design process, artists usually rely on an interactive process to think and create through minds and hands simultaneously. Compared to machining, manual clay sculpting satisfies this need, with its modifiable and superimposable layers of sculpting strokes. Additionally, the often imperfect surface finish records the working process of the artist and thus becomes a feature of artistic expression. Such patterns and textures are difficult to achieve and often not considered in CNC machining, and if required, they are typically only achieved by subsequent special surface treatments.

With the long term goal of endowing robots with human-level skill, we present *RobotSculptor*—a user-guided design and motion planning framework for robotic clay sculpting. By isolating those parameters related to the aesthetics of the sculpture from the fabrication process, we enable the user to define the style of the result. Our system automates the sculpting process by generating feasible motion trajectories that can be executed by robot arms.

In order to make the computational problem tractable, we focus on a sculpting process that only involves subtraction of material, using custom-shaped wire loop tools (Figure 3). The use of such customized tools poses two challenges: since the tools are directional, no off-the-shelf algorithm is available, as conventional path planning algorithms for CNC machining often treat the milling bit as axisymmetric. New planning algorithms are needed to suit our tool, i.e. manage all 6 Degrees of Freedom (DoFs). The other challenge is how to support a wide range of possible sculpting strokes that unleash the expression of the user's creativity while simultaneously complying with the above algorithms.

We open our investigation with a set of experiments aimed at identifying the primary parameters that affect the expression of the user's design intent. Building on the insights gained through these experiments, we then address the challenges listed above by separating the entire pipeline into two independent units:

- **User-Guided Initialization** exposes a set of parameters for the user to control interactively, and transfers the input style information into a series of initial tool positions, i.e. a toolpath, that matches their design intent. This part aims to provide the user the freedom to design the sculpting strokes;
- **Path Planning** takes the initial toolpath as input to an optimization and computes the complete robot trajectories. This part aims to find a high fidelity approximation of the input that balances accuracy and design expression. It further resolves any collisions and respects all other workspace constraints of the fabrication robot.

We demonstrate the versatility of our computational approach on a set of examples of increasing complexity, and finally fabricate these objects with a *Universal Robot UR5* (5 kg-payload version), to assess the degree to which simulated results translate to the real world.

1.1 Overview

As illustrated in Figure 2, our pipeline proceeds as follows:

(1) Given a mesh representation of the target model as input, the user sketches free strokes on the model to define preferred independent areas for further processing.
(2) For each disconnected stroke group, the system computes a decomposed patch.
(3) With a minimal set of user input on directional guidance, our system generates the initial sculpting paths for each patch.
(4) After the toolpaths have been initialized, the optimization adjusts the toolpaths to eliminate collisions and smoothen sharp corners, while still maintaining the artistic expression.
(5) Finally, once the toolpath has been successfully optimized, the user can preview the simulated result, or directly execute the trajectory information computed from the optimization to obtain a physical artefact of the "stylized" target geometry in clay.

We organize this paper as follows: Section 2 covers work related to our research. Section 3 shows a series of design experiments we conducted to understand how the material deforms with specific fabrication parameters. Section 4 explains how we build our design system based on the important parameters extracted from those design experiments. Section 5 presents the optimization formulation that helps to transform design intent into collision-free, feature-preserving robot trajectories. We demonstrate the capacity of our system on a set of physically fabricated examples in Section 6, and discuss the limitations and future work in Section 7. See also the accompanying video, which showcases the pipeline along with the results.

2 RELATED WORK

Clay Fabrication. As a representative material for geometry forming, clay sculpting has been widely used in arts and crafts fields as a hand modelling process [Faraut and Faraut 2013; Philippe Faraut and Charisse Faraut 2009]. Recently, its economical and malleable characteristics have also led to increasing popularity in 3D printing and multi-axis robotic applications. These applications typically employ a customized tool attached to a common 3-axis CNC machine [Nan et al. 2016] or a 6-axis industrial robot arm [Bechthold 2016] and manufacture artefacts through additive, subtractive, or formative processes. Additive processes usually deposits clay either in layers to create sealed surface geometries, or in a woven style [Friedman et al. 2014; Rosenwasser et al. 2017] to create patterns. Deposition processes start either from a non-planar base geometry [Dunn et al. 2016; Ko et al. 2019] or a planar base that is gradually transformed to a non-horizontal fabrication plane [Bhooshan et al. 2019; Trilsbeck et al. 2019]. Taking advantage of the material's malleability, digitally controlled throwing of the clay has also been studied in order to erect large-scale building structures over distance [Dörfler et al. 2014]. Subtractive and formative processes, however, generally start with an initial block of clay which is shaped by either applying pressure to deform the material [Tan and Dritsas 2016], or cutting off material [Schwartz and Prasad 2013].

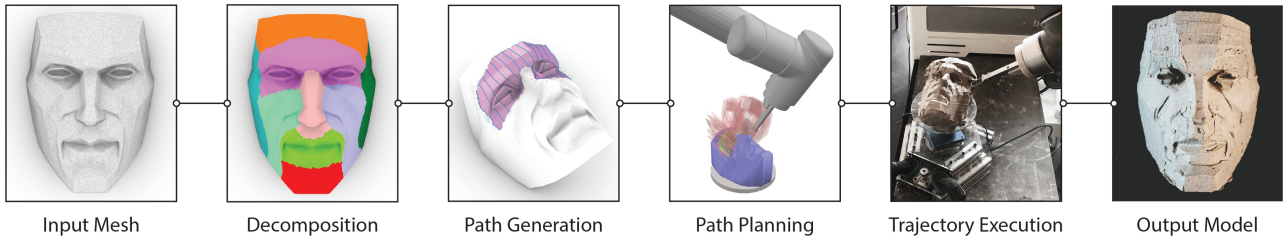

Figure 2: System overview. It takes four steps to design & sculpt a given model with specific styles: 1) the system takes a general triangle mesh as input and decomposes it based on the drawn strokes. 2) The user specifics sculpting styles based on the patch-level parameters and generate a set of initial toolpaths using our system. 3) Using the initialized toolpaths as input, the optimization will compute robot trajectories while maintain style information and resolve collisions simultaneously. 4) The trajectories are executed on a *UR5* to sculpt a physical clay model that match the optimized results.

Weichel et al. [2015] combined additive and subtractive processes (i.e., milling) using two distinct tools.

Design Input. While engineering problems are often well-defined and have clear objectives to optimize towards, design questions are usually open-ended and require different approaches to resolve. For instance, in order to increase the design variety and expressiveness, designers usually prefer a forward design process where they can apply modifications interactively, based on simulated or even fabricated results—and typically do not settle for rigid objectives before a final result is deemed adequate [Clifford et al. 2014; Schwartz and Prasad 2013]. Interactive approaches are employed to enlarge the variety of the resulting appearance through specifically designed rapid-prototyping systems. These systems provide instant feedback or guidance during the fabrication process [Braumann and Brell-Çokcan 2015; Johns 2017; Peng et al. 2018; Zoran and Paradiso 2013] and embrace the imprecise mapping between the digital models and the physical results.

Unlike the interactive fabrication approaches mentioned above, our approach still automatically fabricates the target, but intends to facilitate the artistic design process through an interactive toolpath initialization, and compute feasible toolpaths that balance the user's design inputs and model accuracy through an optimization process.

We must identify the factors that affect the translation of manually-created designs into machine commands. In our context, the distribution of toolpaths vitally influences the appearance of the final surface. Kontovourkis and Tryfonos [2018] and Rael and Fratello [2017] demonstrated potential applications in this direction, but it still remains open to what extent a robot can achieve subtractive clay sculpting. This leads us to develop style-oriented toolpath generation techniques that have rarely been touched.

Path Planning. While toolpath planning plays an essential part in our work, we frame it in a broader context of path generation problems where a large body of work is available in CNC machining [Chiou and Lee 2002; Feng and Li 2002; Jun et al. 2003; Sullivan et al. 2012; Tournier and Duc 2005; Zhu and Lee 2004]. One main difference of our work lies in the customized tool, which requires specially designed path planning algorithms to manage all of its 6 DoFs, compared to a usual 5-DoF milling bit. Dragomatz and Mann [1997] surveyed general path generation methods used in CNC milling, and Elber and Cohen [1994] summarized two main approaches, *isocurves* and *contours*, and their strengths and weaknesses. Our method is similar to the *isocurve* approach, but allows for more flexibility in toolpath generation as we care more about the design expression than machining time or toolpath length. To increase the variety of surface styles, we further employ a "divide-and-conquer" concept for toolpath generation by splitting the target geometry into several sub-sections. Similar approaches have also been applied to multi-axis milling path generation [Muntoni et al. 2018; Zhao et al. 2018].

The general problem of robotic path planning has been studied extensively in the past, and respective software is readily available. For example, the Open Motion Planning Library [Şucan et al. 2012] provides a collection of sampling-based algorithms to plan a feasible path between two points, subject to optimality conditions. The *Descartes* package of the ROS-Industrial project [Edwards and Lewis 2012] implements a tree search to find a robot trajectory that matches a suite of prescribed tool positions. Unlike these applications, however, toolpath planning for our application calls for long trajectories with very dense sampling, in order to accurately follow the fine-scaled details of the target shape. Further, the entire length of the cut path is heavily restricted by collision constraints and computationally expensive optimization criteria. Such conditions are inherently challenging for sampling based approaches. Notably, De Maeyer et al. [2017] report that already for 50 trajectory points, memory usage starts to become a matter of concern for the tree search used by *Descartes*. In our application we routinely exceed this number ten-fold. We therefore choose to rely on iterative optimization, namely Newton's method, to handle the large number of parameters—albeit at the expense of global optimality.

Robotic manipulation of a wire-like tool has recently been studied in Duenser et al. [2020], where an elastically deformable, heated rod cuts through blocks of polystyrene foam. That work focused on trajectory optimization for a small number of individual cuts using a comparably large tool, rather than on a global cutting strategy. In contrast, the tools we employ are much smaller in size, such that a global strategy for path generation is necessary. Nevertheless, we draw inspiration from their work for our path planning step, and optimize for feasible robot trajectories in a similar fashion.

3 DESIGN FACTOR EXTRACTION

Instead of developing a fully automated system similar to existing software for Computer-Aided Manufacturing (CAM), we intend to provide the user with control over those aspects of the fabrication process that are relevant for the design and appearance of an object. Thus, we need to first understand what factors affect the fabrication result of a sculpting process so as to abstract them into parameters that can be built into our system.

In order to reveal the most important design parameters, we conducted a series of experiments involving the interaction between the tool and the clay material. These experiments were designed to help us in three aspects:

- Understand the relationship between the material deformation and sculpting velocity;
- Guide the selection of a suitable shape and size of the customized tools;
- Decide on a minimal set of parameters exposed to the user to exert control on the toolpath generation.

Before discussing the details of these experiments, we briefly introduce our tool designs.

3.1 Customized Loop Tool

A conventional *loop tool* (Figure 3) for cutting clay consists of a handle and a planar "loop", a piece of steel wire or a thin, narrow metal strip bent into rectangular, triangular, or circular profiles to fulfill different cutting needs (size, angle, texture effects, level of detail, etc.). While the sculptor uses their hands for the modelling (additive) and formative process, such loop tools are usually used for the subtractive process—cutting a strip of clay off by moving the tool along a desired path.

Figure 3: Left: different manual loop tools used by professional sculptors; Right: our customized loop tool that can be attached to a *UR5* as the robot end effector.

We use similar customized tools with replaceable "loops" (Figure 6) and a handle that can be attached to the robot. Compared to a conventional milling bit, one important benefit is the non-axisymmetry of the tool, which allows it to cut off clay strips of different widths and sizes by simply rotating around its axis. While this additional flexibility is trivial for human users to control, it adds significant complexity to the planning algorithm—the additional degree of freedom needs to be managed and exploited.

3.2 Parameter Extraction Experiments

We categorize the experiments into two classes: patch-level parameters and path-level parameters. The patch-level parameters affect the selected areas ("patches") of the mesh on which the preferred sculpting styles are applied; the path-level parameters affect the toolpaths generated on each patch. The selected patch can be either a portion of the whole mesh or the mesh itself.

Patch geometry (patch-level). This parameter is directly related to how an input model is decomposed. It defines the area and shape to which a particular style can be applied, and can be created with various methods. We implemented a sketch-based method for the interactive design process in our system (Section 4). Note that simple cases, manual decomposition with any mesh operation software may suffice.

Patch overlap (patch-level). We observed undesired material aggregation near the seams between individual patches, leading to a clear visual separation. This is caused by the high ductility of the material—when the tool enters or exits the clay, at the material interface, it carries forward some material by pushing or pulling, rather than causing a clean separation. This effect is most visible when entry and exit locations accumulate in the same spot. We found that we could reduce this effect sufficiently by introducing an overlap between adjacent patches, as illustrated in Figure 4, and thus eliminating the accumulation.

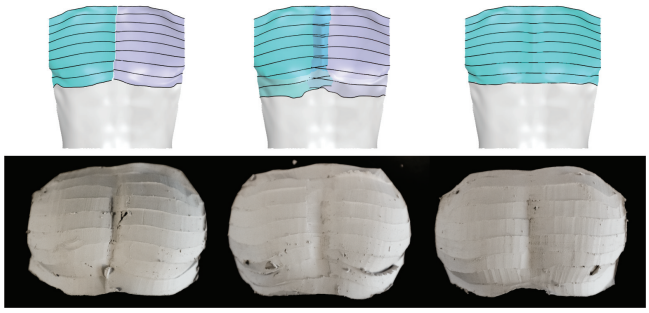

Figure 4: Seam comparison models. Left: sculpt paths intersecting at seam area without overlapping; Middle: sculpt paths intersecting at seams with overlapping; Right: continuous sculpt paths across the whole surface.

Toolpath length (path-level). The above-mentioned material property has a similar impact on the toolpaths generated for a specific patch. Regardless of the generation methods, a toolpath will start/end in three circumstances: 1) at the start/end point of another toolpath (toolpaths connected), 2) at the middle of another toolpath (toolpaths overlapped), 3) at an empty area (toolpath disconnected).

Our experiments showed that for a specific patch, 1) and 2) will always create leftover material at the intersection, and 3) will result in a area not sculpted in the target geometry. As the number of intersection locations is largely decided by the number of toolpaths, we favor long toolpaths to reduce this aggregation. Two extreme cases are shown in Figure 5, where one contains randomly generated short toolpaths on various directions and the other contains only aligned toolpaths across the entire surface.

Toolpath direction (path-level). This parameter affects the toolpath generation process, and is the most important one for defining

Figure 5: Left: surface sculpted with 100 toolpaths of 18 mm length in random direction, generated from 100 randomly sampled points; Right: the same surface sculpted with 15 parallel toolpaths across the whole width of the patch.

the artistic style the user wishes to achieve. It affects the visual effects of the sculpted stripe patterns on the final surface as well as the cutting depth into the clay. We use a Laplacian-based algorithm to generate evenly-distributed parallel-aligned toolpaths on top of each path, and the details are explained in Section 4.3.

Tool direction (path-level). As shown in Figure 10, the three rotation parameters define the local pose of the tool. Our experiments confirmed that the *aligning direction* affects the precision of the target surface, and the *facing direction* affects the amount of material cut by each toolpath. These parameters together also affect the final surface quality (Section 4.4).

Secondary parameters. Besides the parameters described above, we also experimented with several other parameters, though these were found to be less effective in influencing the design and fabrication results. For completeness, we list them below:

- *Density of toolpaths*: This parameter needs to ensure that the sculpted area covers the whole patch. Beyond that, increasing the density only increase optimization time with little gain for a selected tool. However, this parameter is still exposed to the user to compensate for any change in the tool.
- *Inclining direction*: This parameter does not affect the result as much as the other two listed in the *Tool direction* categories, as long as it does not cause any collisions.
- *Tool shape*: As shown in Figure 6, we experimented with various tool shapes. However, we do not allow for tool changes during a sculpting task, so we exclude this parameter from the design stage. Note that the same optimization pipeline is applicable to different tool shapes.

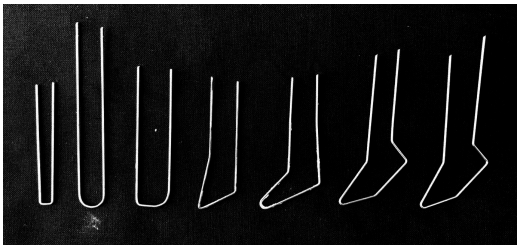

Figure 6: Customized loop tool heads of different shapes.

3.3 Material Properties

As briefly mentioned in Section 3.2, we discovered that the viscosity and plasticity of the clay affect, on a local scale, how the clay behaves when the tool enters, sculpts, and exits it, which then directly affects the final appearance of the sculpted model. There are two main effects: 1) The clay sufficiently soft to be pushed by the tool during the sculpting process. Although we use a thin 1 mm steel wire to reduce this effect as much as possible, leftover clay is still noticeable along the moving paths of the tool. 2) Due to viscosity, the forces introduced by the tool cause a "pulling" effect when leaving the clay, and can even cause failure to detach when the remaining material is unable to withstand these forces. This effect mainly happens between the clay subtracted by the tool and the clay model, resulting in small accumulations on the target surface.

A complete modelling of the clay is extremely challenging, as its material properties change over time when the contained water evaporates gradually. However, for a thin (1 mm) tool made of steel wire, we found that by limiting the cutting speed to within 3 cm/s to 8 cm/s we could reduce these visible defects to an acceptable level. We thus decided to conduct the fabrication under these settings and formulate the optimization using a purely geometric approach, resolving robot motion and collision issues without involving any simulation of the material behaviour.

3.4 Style as an Aesthetic Feature

One of the core contributions of our work is to deviate from a conventional path planning task by bringing the designer into the loop—to embed the user's design expression as the sculpting styles in an automated robotic process. It provides a different perspective to robotic processes by adding manually-controlled elements into the "design-to-fabrication" process, and provides users with more freedom and control over their design expressions. Although not designed to be a computer-human interaction system, our system embeds design preferences and choices in a predefined manner.

While conventional CNC milling prefers precision, our system favours the possibility of creating various visual styles with a minimum amount of effort. It identifies a set of design parameters abstracted from fabrication experiments, and transfers the designed styles from the digital environment to physical artefacts with ease. As evaluating the aesthetics of a sculpt is difficult and inherently subjective, we leave it to the user to realize their creativity and intention by providing them with a considerable amount of freedom to explore this design space.

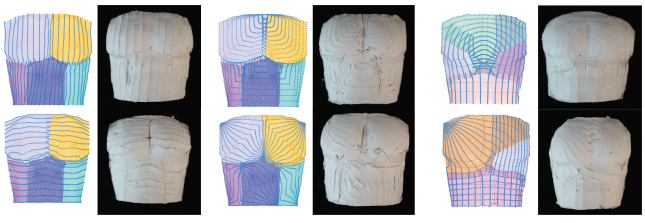

Figure 7: Sculpting styles created by *RobotSculptor* with different decomposition schemes and toolpaths.

Figure 7 shows the sculpting style variations of a torso model created by different decomposition schemes or toolpaths. The visual styles formed by the sculpting toolpaths define a unique feature of the sculpting process. We believe these style variations provide new opportunities to explore new ways of robot control and open up discussions in human-robot interaction. More details are discussed in Section 4 and Section 5.

4 USER-DRIVEN TOOLPATH GENERATION

4.1 Design Parameters

One important goal of our research is to embed human design choices and expressions as "styles" into the automated robotic fabrication. This requires the system to maintain a certain magnitude of precision, and at the same time deviate from the homogeneous look typical for the results of CNC milling. We rely on the key parameters selected based on the design experiments described in Section 3.2 to allow users to generate toolpaths creatively and transfer essential features into the fabrication process. Following the pipeline described in Section 1, we assume the input mesh as conceptually quadrilateral (for ease of geometry processing, we only define 4 corners + 4 edges to the mesh, whatever the true shape is) and interpret the selected 5 parameters into variables that the user can access and modify in the GUI:

(1) Locations of free strokes drawn for model decomposition;
(2) Offset distance at the overlapping area between patches;
(3) Locations on patch boundaries as conceptual "corners";
(4) Distribution of start and end points of the toolpaths;
(5) Number of toolpaths generated on each patch;

(1), (2) are *patch-level parameters* and relate to the *Decomposition* process; (3), (4), (5) are *path-level parameters* and relate to the *Toolpath Initialization* process. We will explain both of them in detail below.

For both CNC milling and our system, one necessary step of the toolpath generation is to develop toolpaths that can cover the whole surface of the input model. While common milling tasks use widely applied strategies including *parallel*, *scallop*, *radial* and *flow-line*, we require a different procedure to generate toolpaths as the robot end effector (i.e. the customized loop tool) is not axisymmetrical, as normal milling bits are. The normal of the cutting plane must be aligned towards the cutting direction for an effective cut (though it doesn't need to be aligned fully), so standard strategies would be insufficient.

Therefore, we developed a global-to-local strategy that decomposes the input model into small patches that can incorporate different sculpting intent. Treating each patch individually, we generate toolpaths based on the isolines of a scalar field, which in turn is defined through user-provided boundary conditions for each patch. If no decomposition is given, the system will treat the whole mesh as a single patch, and conducts the toolpath generation over the whole area.

4.2 Decomposition

The *Decomposition* aims to allow the user to select different areas that can be treated separately for the toolpath generation. We developed a GUI to facilitate this task. The user can draw strokes on the model using a mouse, and the system will compute a distance field for each disconnected stroke. This field measures the distance between mesh vertices and the strokes, and later helps to compute separate surface patches using a priority queue based on the measured distances. Once the result is visualized, the user can accordingly decide to either draw additional isolated strokes to create more patches, or to intersect existing strokes with new stroke(s) to modify the shape of the corresponding patches (Figure 8). See also the supporting video.

Figure 8: Left: distance field calculated from the drawn strokes; Middle: decomposed patches without overlapping boundaries; Right: decomposed patches with overlapping boundaries of 15 additional triangle loops.

Once the *patch-geometry* has been defined, the user can modify the overlapping areas around the borders where patches intersect. As the dimension of the input model may vary, this is achieved by adjusting the number of facets in the overlap areas (Figure 8). This adjustment aims to prevent the aggregation of entry/exit locations of the loop tool, which will produce inferior surface quality due to the material behaviour discussed in Section 3.3.

4.3 Initialization

The *Initialization* process aims to provide intuitive toolpath generation for each decomposed patch. We treat each patch as a "quad-like" patch and ask the user to provide four "cutting" points near the boundary of each patch. These points are used to segment the closed boundary curve into four segments, i.e. two facing pairs. We assign the vertices of the two segments of one of the pairs with the value 0 and 1 respectively and assign that of the other pair with values interpolated from 0 to 1.

To generate the isolines, we use a technique similar to those described in Ma et al. [2020] and [Pereira et al. 2014]. For each surface patch with n vertices, we compute a scalar field by solving the common Laplacian equation with boundary constraints:

$$\begin{cases} \mathbf{L}\mathbf{z}(x) = \mathbf{0}, & x \in \Omega \\ \mathbf{z}(x) = \mathbf{z}_0(x), & x \in \partial\Omega \end{cases} \tag{1}$$

where $\mathbf{L}$ is the $n \times n$ discrete Laplacian and $\mathbf{x}$ are the coordinates of the mesh vertices. Variables $\mathbf{z}(\mathbf{x})$ and $\mathbf{z_0}(\mathbf{x})$ are vectors of per-vertex values of all the vertices and boundary vertices, respectively. Those elements of $\mathbf{z}$ that corresponds to the interior vertices are unknown, while the elements corresponding to the boundary vertices are given as constraints. We allow the user to modify the path direction and orientation by adjusting the position of the cutting points, the distribution of the assigned values, and the number of toolpaths (Figure 9).

We then interpolate a series of isolines from the scalar field. The user can set parameters interactively to find a path initialization

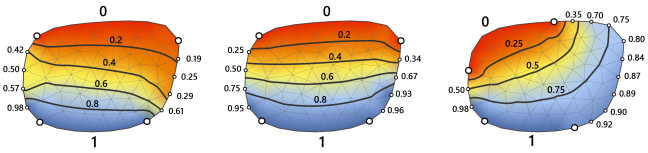

Figure 9: Toolpath initialization: Left & Middle: same cutting point location, different distributions of assigned boundary values; Right: different cutting point location, distribution of assigned boundary value and density of paths.

that matches his vision. We found that an overlap of more than 30% of the tool width between adjacent paths is needed to allow for the optimization to modify the paths sufficiently in order to avoid collisions or match the target geometry more closely.

4.4 Tool Direction Modification

Although the *Decomposition* and *Initialization* processes successfully help in transferring the design intention to initial toolpaths, we can further improve our initialization through local adjustments of the tool direction. While we can generally rely on the optimization to compute the results, experiments showed that better initialization would often lead to better surface quality and faster optimization, especially in high curvature areas where local minima occur.

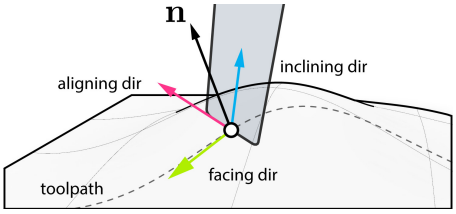

Figure 10: During a sculpting movement, the tool pose is defined by three vectors: the facing direction, the aligning direction, and the inclining direction.

As the loop tool has 6 DoF, we define the 3 directions that are not constrained by a given toolpath (in fact, a series of tool positions) as *facing direction, aligning direction* and *inclining direction* (Figure 10). A milling bit has no facing direction as it always cuts at the width of the tool's diameter. For the loop tool, the cutting profile depends on the projection of the tool profile to the material along the toolpath direction and can be adjusted by its relative angle to the tangent direction of the toolpath.

For a sampled tool location along a toolpath, we initialize the *inclining direction* using the normal direction of the patch, and project the tangent direction of the toolpath to the tangent plane of the patch at the referenced point to initialize the *facing direction.*

Additionally, we re-align some of the tool's *facing directions* perpendicular to the averaged principal curvature [Meyer et al. 2003] directions near high curvature areas:

$$\mathbf{n}_f = \frac{1}{N} \sum_{r<r_{near}} \mathbf{n}_p \tag{2}$$

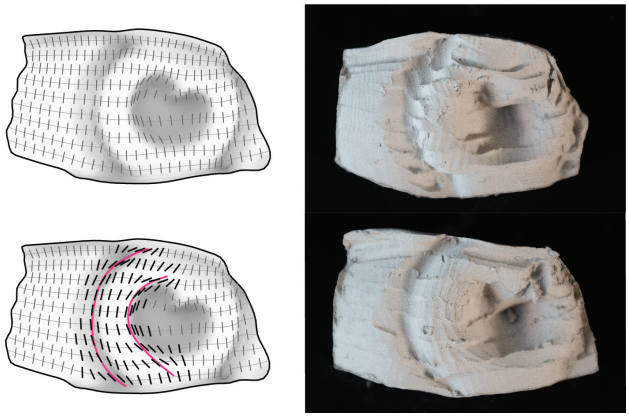

Figure 11: Local adjustment of the *facing direction* using curvature information. The fabrication results illustrate noticeable improvements of the surface quality.

where N is the number of samples of the principal curvature $\mathbf{n}_p$ within a pre-defined sphere of radius r_{near} around the tool location. We illustrate the benefits of this post-processing step in Figure 11.

With the above procedures, we obtain a general initialization of both toolpaths and tool directions. However, there is no guarantee that these results can be executed with a specific robot without any collision or reachability problems. It would thus require the user to manually modify the paths iteratively for a specific robot in use to resolve all collision issues, or use a simplified version / allow certain collisions (Figure 15 middle, Figure 16 upper right)—this one of the main reasons that led us to develop the optimization process described in Section 5.

5 OPTIMAL PATH PLANNING

The toolpath generation in the previous sections defines a path that sweeps the target surface closely and expresses the aesthetic preferences of the user. It is, however, not guaranteed to be feasible, in the sense that it can be executed by a given robot without causing collisions or exceeding the robot's reach. Therefore, given a patch of the target surface and the associated toolpaths (collectively referred to as *input toolpath* in this section), we need to find a robot trajectory that 1) is feasible, 2) produces a cut surface that best approximates the target surface and 3) maintains the overall aesthetics of the cut surface implied by the input toolpath.

We follow an approach similar to the one proposed by Duenser et al. [2020] for computing cut trajectories for an elastically deformable tool, manipulated by a two-armed robot. At the core of this approach lies the formulation of an optimization problem which matches the surface swept by the tool during movement (*toolsurface*) with the surface of the input model (*target shape*). In particular, we use similar formulations for the physical model of the system, the final *primary objective*, the *constraint objectives* and the last two of the *secondary objectives*, as introduced below.

Model description. The robot trajectory is represented through a sequence of robot poses (*trajectory points*), each defined by the set of joint angles $\boldsymbol{q}_i$, collectively forming the full trajectory $\boldsymbol{q} = (\boldsymbol{q}_i)$.

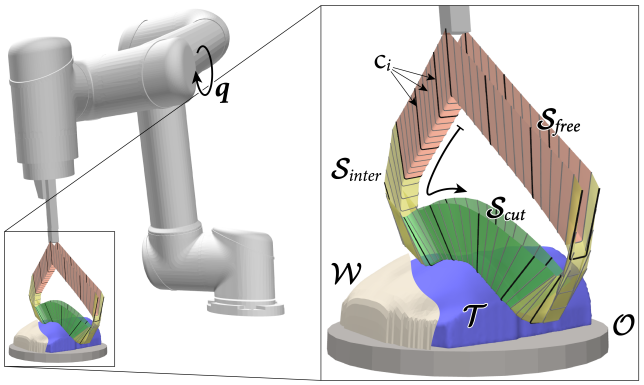

Figure 12: An overview of the main components of the optimization model. The robot is shown in its rest pose, from where it traverses towards the workpiece (toolpath $\mathcal{S}_{free}$ and $\mathcal{S}_{inter}$) and performs the cut ($\mathcal{S}_{cut}$). The robot then moves back to its rest pose—although typically it would loop around and perform a number of successive cuts, optimized simultaneously, to carve out the entirety of a given surface patch.

In case a turntable is used, we simply view it as an additional robot joint and include its orientation in $\boldsymbol{q}$. The tool is rigidly attached to the robot end effector and modeled by its center line $\boldsymbol{c}_i$, such that the path swept by the tool forms the *toolsurface* $\mathcal{S}$. Between the discrete steps of the trajectory we approximate this surface as piecewise linear. Using a kinematic model for the robot, the toolsurface is then fully defined through the joint angles as $\mathcal{S} = \mathcal{S}(\boldsymbol{q})$. For a full description of the setup, we further consider the target shape $\mathcal{T}$ and its currently processed subsection $\mathcal{T}^*$, the current shape of the workpiece $\mathcal{W}$, as well as any other obstacles $\mathcal{O}$ in the scene, such as the turntable. If the target mesh is split into several patches, the shape of the workpiece is updated after applying each of the corresponding cuts. See Figure 12 for an overview of the simulated setup.

Optimization problem. Similarly to Duenser et al. [2020], we formulate an unconstrained optimization problem of the form

$$\min_{\boldsymbol{q}} \quad E(\boldsymbol{q}) = E_{prime} + E_{constr} + E_{sec}, \tag{3}$$

where all physical constraints are enforced through penalty terms, collectively denoted E_{constr}. The principal design objective E_{prime} defines a cost for the distance between the toolpath and its target, while E_{sec} collects several secondary objectives, as laid out in more detail below. We solve this minimization problem using Newton's method with line search and a Levenberg-Marquardt type regularization.

The trajectory we optimize, and correspondingly the toolsurface, consists of several distinct, predefined subsections: One or more cut portions $\mathcal{S}_{cut}$, in accordance with individual cuts of the input toolpath, which are designated to carve out the target shape. Transitional portions $\mathcal{S}_{free}$, which describe the free movement in-between individual cuts, as well as from and to a fixed robot rest pose. And finally, intermediate portions $\mathcal{S}_{inter}$, which are short connecting sections at the interface between $\mathcal{S}_{cut}$ and $\mathcal{S}_{free}$. While the toolsurface of these sections may take part in cutting through the material, it is not optimized to match the target shape.

5.1 Primary Objective and Constraints

Surface Matching. The primary objective E_{prime} measures the closeness between the toolpath and the given target. We view this as a non-rigid surface registration problem and match the target surface $\mathcal{T}^*$ with the toolsurface $\mathcal{S}_{cut}$. Starting from a dense set of sample points on the target surface $\mathcal{T}^*$, we penalize the absolute distances to their respective closest points on the toolsurface $\mathcal{S}_{cut}$. In principle, a simple quadratic penalty could be used for this. Although in a case where portions of $\mathcal{T}^*$ can not feasibly be cut, this choice can lead to an undesirable overemphasis on these regions. Instead, we turn to a smooth step function of the form

$$H_\tau(d) = \begin{cases} 3\left(\frac{d}{\tau}\right)^2 - 2\left(\frac{d}{\tau}\right)^3 & 0 \le d < \tau \\ 1 & d \ge \tau. \end{cases} \tag{4}$$

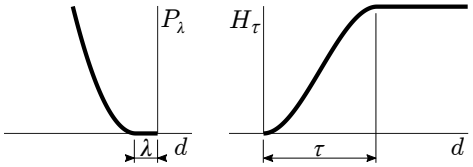

Figure 13: Penalty functions on distance used for collision avoidance (left) and surface matching (right).

This function acts similar to a quadratic penalty for a distance d close to zero, but smoothly transitions to a constant penalty over a transitional region of size τ (Figure 13, right). Thereby, regions that are definitely uncuttable, i.e. with a distance larger than τ, are simply ignored.

Initialization Procedure. Due to the relatively fine-scaled geometry of the toolsurface and its very low rigidity, the outlined surface matching is prone to a large number of undesirable local minima. It therefore relies on a fairly good initialization, for which we use the input toolpath. To this end, we split the optimization process into two distinct stages. During the first, we do not apply the surface matching objective as the primary objective. Rather, we match the cut portion of the toolpath to the input toolpath, with regards to the position and orientation of the tool, using a quadratic penalty. Once a toolpath is found which resembles the input path as close as possible but has a feasible trajectory, we gradually drop this initial objective and apply surface matching instead. Using the input toolpath as initialization also establishes the desired global path layout, and in our experiments we found that this layout was generally well preserved during the surface matching stage, even once the initial objective had been removed entirely. At the same time, matching only the toolsurface provides a larger degree of freedom for the robot trajectory, allowing it to gracefully avoid collisions even in challenging situations.

Physical Limits. The constraints we consider are the robot's limitations on joint angles, as well as collisions of the robot and the tool. These collisions are namely: 1) self-collisions of the robot, 2) collisions between the robot and the workpiece $\mathcal{W}$ and obstacles O, 3) collisions between the toolsurface $\mathcal{S}$ and the obstacles O, 4) collisions between $\mathcal{S}_{free}$ and the workpiece $\mathcal{W}$, and 5) penetration of $\mathcal{S}_{cut}$ and $\mathcal{S}_{inter}$ into the target shape $\mathcal{T}$.

For the implementation of robot collisions, the robot model is equipped with a number of spherical collision primitives, typically eight per link. From each collision primitive the signed distance is computed to all of the other collision spheres, as well as to the closest point on each of the objects in the scene. The latter are accurately represented through triangle meshes. A negative sign of the distance thereby signifies penetration. Similarly, proximity of the toolsurface $\mathcal{S}$ is evaluated on a dense set of sample points on the surface, for each of which the smallest distance to the relevant objects is computed. We then penalize these distances with the one-sided quadratic function

$$P_\lambda(d) = \begin{cases} (d-\lambda)^2 & d < \lambda \\ 0 & d \geq \lambda, \end{cases} \tag{5}$$

where λ is a safety margin or tolerance (Figure 13, left). The same type of penalty is applied directly to the joint angles of the robot. The weighted sum of all penalties constitutes the full constraint objective E_{constr}, whereby the weights are chosen to be large compared to any of the remaining objectives, such that the constraints are enforced rigidly.

5.2 Secondary Objectives

We identified several additional criteria for the quality and practicability of a toolpath, enforced through additional objectives E_{sec}.

Orthogonal tool orientation. For the fabrication process, it is favorable to keep the cutting direction orthogonal to the tool plane. While a cut can be produced when the tool plane is aligned with the cutting direction, this would produce only a narrow slit, often without fully removing a portion of clay from the workpiece. There is a high risk the clay will subsequently reattach, effectively undoing the cut. By only cutting orthogonal to the tool plane, long, narrow shavings are produced which can be removed immediately. Let $\boldsymbol{c}_{ij}$ be the sample point j of the tool of time step i. For each $\boldsymbol{c}_{ij}$ we penalize the deviation of the tool facing direction $\mathbf{u}_i$ from the local cut direction $\mathbf{v}_{ij}$, for those sample points on the tool engaged in the cutting, as

$$E^{ij}_{orth} = l_{ij}\ \sin^4(\angle(\mathbf{u}_i, \mathbf{v}_{ij}))\ H^*_{a,\tau}(d_{\mathcal{W},ij}). \tag{6}$$

The symbol $\angle(\cdot,\cdot)$ is the angle spanned by two vectors. The cut direction is computed as $\mathbf{v}_{ij} = 1/2\ (\hat{\mathbf{v}}_- + \hat{\mathbf{v}}_+)$, where $\mathbf{v}_- = \boldsymbol{c}_{i,j} - \boldsymbol{c}_{i-1,j}$, $\mathbf{v}_+ = \boldsymbol{c}_{i+1,j} - \boldsymbol{c}_{i,j}$, and $\hat{\cdot}$ represents a normalized vector. The associated step size $l_{ij} = 1/2\ (\|\mathbf{v}_-\| + \|\mathbf{v}_+\|)$ is used to weight the objective. Finally, the last term of the equation represents a weight in the range $[0,1]$ indicating whether the sample point is inside or close to the workpiece $\mathcal{W}$ and therefore is relevant for the cut. Herein $H^*_{a,\tau}(d) = 1 - H_\tau(d-a)$ is an inverted smooth step function shifted by a tolerance a, and $d_{\mathcal{W}}$ is the signed distance between the sample point and $\mathcal{W}$.

Smooth discrete toolpath. To ensure smoothness of the discretized toolpath we penalize the angle spanned by the piecewise linear path of a tool sample point at each time step through

$$E^{ij}_{smooth} = l_{ij}\ \alpha^2_{ij}\ H^*_{a,\tau}(d_{\mathcal{T},ij}), \tag{7}$$

where $\alpha_{ij} = \angle(\mathbf{v}_-, \mathbf{v}_+)$. This angle can essentially be viewed as the ratio between the local, approximated curvature of the toolpath (i.e. α_{ij}/l_{ij}) and the sampling density (given by $1/l_{ij}$). Thus, the objective does allow for an arbitrarily large curvature of the path, provided that the temporal resolution is adequate locally. As above, we weight the objective with the path length l_{ij}, and also according to the closeness $d_{\mathcal{T}}$ to the target shape, such that only portions of the cut are affected which may be visible in the final object.

Limited joint angle step size. While for the optimization we assume the toolpath is given by linear interpolation of the tool geometry at discrete time steps, during fabrication the robot trajectory is interpolated linearly in joint angle space. For the k^{th} joint of the robot, a step of $\beta_{i,k} = \boldsymbol{q}_{i,k} - \boldsymbol{q}_{i-1,k}$ in joint angle space induces a maximum interpolation error of

$$\epsilon_{i,j,k} = r_{i,j,k}\ \left(1 - \cos\left(\frac{\beta_{i,k}}{2}\right)\right), \tag{8}$$

where $r_{i,j,k}$ is the distance between a tool sample point $\boldsymbol{c}_{i,j}$ and the k^{th} robot axes. For simplicity, we assume a rough, fixed estimate $\tilde{r}_k$ for this distance for each joint angle, and penalize the corresponding approximation error through

$$E^{i,k}_{joint} = \left(\tilde{r}_k\ \left(1 - \cos\left(\frac{\beta_{i,k}}{2}\right)\right)\right)^2. \tag{9}$$

Limited tool step size. Collision avoidance of the toolsurface is based on a fixed number of sample points. In order to maintain an adequate sampling density, it is necessary to limit the step size of the tool. Again, we simply apply a one-sided quadratic penalty

$$E^{i,j}_{step} = P_{-\delta}\left(-\left\|\boldsymbol{c}_{i,j} - \boldsymbol{c}_{i-1,j}\right\|\right) \tag{10}$$

to roughly ensure an upper bound of δ.

Quadratic regularization. Finally, we apply a weak quadratic regularization to the tool step size, such that all portions of the toolpath which are not governed by any of the above objectives remain short and smooth:

$$E^{i,j}_{reg} = \left\|\boldsymbol{c}_{i,j} - \boldsymbol{c}_{i-1,j}\right\|^2. \tag{11}$$

6 RESULTS

To demonstrate the versatility of our system, we designed and fabricated four prototypes featuring different geometric characteristics. The decomposition of the input model by drawing strokes in the UI and generating toolpaths for each patch takes around 0.5 h on average, depending on the number of decomposed patches and the number of attempts made to match the user's intention. The optimization takes 1 h to 4 h on average for the models we present here (torso, eye, face, 3D Möbius ring). The fabrication takes around 1 h on average with a joint velocity of 1 rad/s for the leading axis (the *movej* command [Robots 2015]). After fabrication, the clay needs around one day to dry until its surface solidifies, and at least two days to be fully dried. The optimization framework is implemented in C++, making use of the Eigen library [Guennebaud et al.

2010] for matrix algebra. Searches for closest points on surfaces, as required for collision avoidance, are performed through an axis aligned bounding box tree, using the libigl library [Jacobson et al. 2018]. This operation accounts for the largest part of the computational costs in the procedure, with roughly 50%. Another 15%-20% of costs can be attributed the forward kinematics of the robot, and the respective first- and second order derivatives. For context, it should be noted that collisions between toolsurface and target surface are tested on 140 sample points per trajectory point, and the distance function for surface matching is evaluated with similar density. Computation times for all examples are reported in Table 1, obtained on a standard PC with a 3.4GHz Intel Core i7-3770 CPU.

We also refer to the supporting video, which shows the fabrication setup and the physical results.

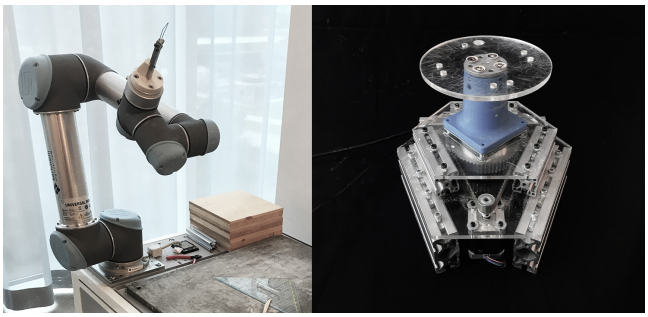

Figure 14: Left: our customized tool attached to the *UR5*; Right: the Arduino-controlled turntable.

6.1 Fabrication Setup

We designed a custom loop tool with a metal handle and a 3D printed ABS base. For the examples shown in this paper, we chose a rectangular profile with 10 mm cutting width and 30 mm cutting depth, made of 1 mm steel wire. The cutting depth is limited by the stiffness of the wire to avoid visible deformation during a cutting process. The loop is screwed onto the customized handle (10 mm × 10 mm cross-sectional area), which is attached to a *UR5* through the ABS base.

We use a custom turntable controlled by an *Arduino Uno* to compensate for the limited reach of the robot. It rotates in both directions with 1.8° resolution, and acts as the 7th axis of our system to rotate the model to a position within the robot's reach. (Figure 14).

6.2 Fabricated Models

Beside the torso model (Figure 7), the simplest of our examples is the eye model (Figure 15), which contains concave features that are nearly impossible to generate collision-free toolpaths for. We made several attempts through our CAD-modelling process, but fell back to use a smoothed version of the model as the collisions cannot be fully resolved. However, our optimization component from the *RobotSculptor* resolves all the collisions and generates toolpath trajectories that achieve fabricated results with reasonable

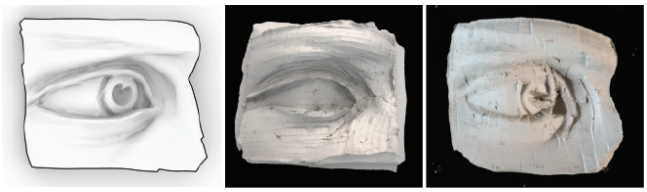

Figure 15: The *eye* model. Left: Input geometry; Middle: model by executing CAD-modelled toolpaths; Right: model by executing trajectories generated from *RobotSculptor*.

quality, even with a customized tool that is oversized for the details around the iris area.

We further use our interactive, user-guided design method to decompose and generate toolpaths for a face model that contains more challenging geometric features around the eye area (concave feature with large curvature) and the nose area (sharp edges). Similarly, CAD-modelled toolpaths failed to resolve collisions around the eye corner, but our *RobotSculptor* system successfully fabricates the different styles we desire(Figure 16).

Figure 16: Sculpting results of the *face* model. Top-left: input geometry; Rest: initialized toolpaths and the results with different styles by executing robot trajectories generated from *RobotSculptor*.

Our system even allows the use of different parts of the tool for the sculpting process. In the 3D Möbius ring example (Figure 17-right), we use the *bottom blade* to sculpt the outer patches, and the *side blade* for the inner patches which are inaccessible to the *bottom blade* due to collision issues. However, we noticed two limitations: 1) Models with a thin connection to the base are likely to be deformed during the fabrication, which causes lower precision. In this example, we compensate it by manually supporting the model. 2) Sculpting with the *side blade*, the maximum cut depth naturally cannot exceed the length of the tool. This can become a limiting factor when cutting the innermost portion of the ring, and constraints the possible size of the model.

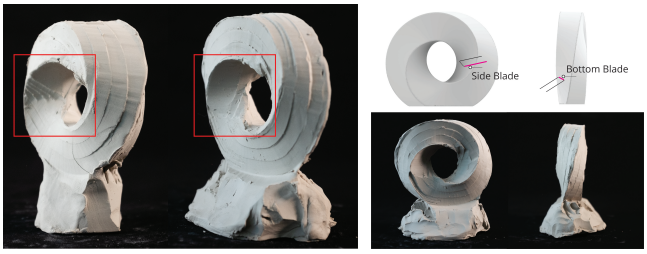

Figure 17: Left: Reachability limitation from inadequate *side blade* length. Right: Illustration of model areas cut by *side blade* or *bottom blade*.

Preference for using a specific edge of the tool can be set by simply choosing the appropriate tool-local frame used for the initialization phase. The subsequent path optimization on the other hand is agnostic to the notion of distinct blades. That is, it treats the entire tool as one blade. Similarly to the overall path layout, we find that any preferences implied by the initialization are typically well preserved.

As shown in Figure 17 (left), we verify the reachability limitation by fabricating two Möbius models with different thickness.

model	# patches	avg traj. pts/patch	opt. time	fab. time
Torso	5	334	1h 57m	7m
Face	6	445	4h 11m	26m
	7	473	4h 30m	31m
	9	412	5h 33m	33m
Eye	3	624	1h 21m	16m
Möbius	8	339	1h 39m	31m

Table 1: Statistics of presented examples.

7 CONCLUSION

We have presented an interactive design and fabrication system that allows users to design different styles for sculpting clay models with a 6-axis robot. We identified and extracted a set of key parameters from a series of sculpting experiments and exposed them to the users in an interactive user interface we developed. The interface allows the user to decompose the input mesh into desired patches by drawing free sketch strokes and embed their design expressions as different sculpting styles individually by generating a set of corresponding initial sculpting toolpaths.

After the toolpaths have been initialized, our system conducts optimal path planning to resolve robot collision and reachability issues while still maintaining a maximum match to the given input surface. To obtain a higher success rate, we also added an Arduino-controlled turntable and integrated it into the optimization pipeline. We have demonstrated the capacity of our system through a set of fabricated desktop-size clay models: torso, eye, face, and 3D Möbius ring. Moreover, as evidenced by the wide variety of styles for the same model, our system successfully enlarges the magnitude of expression incorporation during the design stage.

7.1 Limitation and Future Work

It is the combination of initialization and optimization that makes our system not only a robotic extension of the human hand, but a system with certain intelligence that fulfills certain design intentions. Yet we still have a long way to go to merge the system seamlessly in the human endeavours of design and creation. Many exciting questions are still left open for future work.

First, our system utilizes a subtractive strategy for the sculpting process assuming the clay to be rigid. This assumption works well when sculpting thick areas, but may cause imprecise results due to material deformation at thin sculpted areas (*e.g.* the nose area in the face model). We plan to investigate methods that incorporate material simulation during the optimization for a better prediction. This will contribute to controlling small accumulations caused by the material deformation, and the visual intensity of the styles.

Second, limited by both material properties and the optimization framework, our work only touches the sculpting styles in the manner of stroke density and directions for selected sub-areas, while many other possibilities exist for "artistic expressions". We plan to investigate different initialization strategies to enlarge our toolpath generation library to support more styles, and enrich the optimization component accordingly.

Third, our current system can only predict the sculpted geometry after running the toolpath optimization. As the optimization process is computationally demanding, the current pipeline cannot present a predicted representation of the final appearance to the users instantly. We plan to investigate different methods that can approximate the final appearance independent of the optimization so as to enhance the design process with instant feedback.

Fourth, compared to human artists who conduct a combination of various modelling techniques during an entire sculpting process (additive, subtractive, formative), our system only utilizes the subtractive process, using one type of tool. While combining both additive and subtractive techniques in the fabrication process is not difficult, predicting the material behaviour under formative processes (modelling, pushing) to fulfill the optimization tasks will require a simulation component, additional to the need mentioned in the first point above.

Fifth, we developed the customized UI for non-expert users to work on non-fired clay only. However, similar robotic processes have much larger application, such as foam wire cutting, wax cutting, or even fired clay, where the targeted user group may also extend to experts. As we decoupled the *Decomposition & Initialization* and the *Optimization* components, migrating the former into existing CAD software (for instance, *Rhinoceros*) and keeping the optimization (implemented in C++) as it is for computational efficiency is feasible. Applying the current path planning framework to other materials and processes would require additional physical tests and tool designs.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their helpful comments, and Espen Knoop for help with building the turntable and the customized metal pieces for the tool design.

REFERENCES

Martin Bechthold. 2016. Ceramic Prototypes – Design, Computation, and Digital Fabrication. *Informes de la Construcción* 68, 544 (Dec. 2016), 167. https://doi.org/10.3989/ic.15.170.m15

Shajay Bhooshan, Johannes Ladinig, Tom Van Mele, and Philippe Block. 2019. Function Representation for Robotic 3D Printed Concrete. In *Robotic Fabrication in Architecture, Art and Design 2018*, Jan Willmann, Philippe Block, Marco Hutter, Kendra Byrne, and Tim Schork (Eds.). Springer International Publishing, Cham, 98–109. https://doi.org/10.1007/978-3-319-92294-2_8

Johannes Braumann and Sigrid Brell-Çokcan. 2015. Adaptive Robot Control - New Parametric Workflows Directly from Design to KUKA Robots. In *Real Time - Proceedings of the 33rd eCAADe Conference - Volume 2*, B Martens, G Wurzer, T Grasl, WE Lorenz, and R Schaffranek (Eds.). CUMINCAD, 243–250.

Chuang-Jang Chiou and Yuan-Shin Lee. 2002. A Machining Potential Field Approach to Tool Path Generation for Multi-Axis Sculptured Surface Machining. *Computer-Aided Design* 34, 5 (April 2002), 357–371. https://doi.org/10.1016/S0010-4485(01)00102-6

Brandon Clifford, Nazareth Ekmekjian, Patrick Little, and Andrew Manto. 2014. Variable Carving Volume Casting. In *Robotic Fabrication in Architecture, Art and Design 2014*, Wes McGee and Monica Ponce de Leon (Eds.). Springer International Publishing, Cham, 3–15. https://doi.org/10.1007/978-3-319-04663-1_1

Jeroen De Maeyer, Bart Moyaers, and Eric Demeester. 2017. Cartesian path planning for arc welding robots: Evaluation of the descartes algorithm. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. 1–8.

Vimal G. Dhokia, Stephen T. Newman, Paul Crabtree, and Martin P. Ansell. 2011. A Process Control System for Cryogenic CNC Elastomer Machining. *Robotics and Computer-Integrated Manufacturing* 27, 4 (Aug. 2011), 779–784. https://doi.org/10.1016/j.rcim.2011.02.006

Kathrin Dörfler, Sebastian Ernst, Luka Piskorec, Jan Willmann, Volker Helm, Fabio Gramazio, and Matthias Kohler. 2014. Remote material deposition: exploration of reciprocal digital and material computational capacities. *What's the Matter: Materiality and Materialism at the Age of Computation, ed. Maria Voyatzaki* (2014), 361–77.

Donald Dragomatz and Stephen Mann. 1997. A Classified Bibliography of Literature on NC Milling Path Generation. *Computer-Aided Design* 29, 3 (March 1997), 239–247. https://doi.org/10.1016/S0010-4485(96)00060-7

Simon Duenser, Roi Poranne, Bernhard Thomaszewski, and Stelian Coros. 2020. RoboCut: Hot-wire Cutting with Robot-controlled Flexible Rods. *ACM Trans. Graph.* 39, 4, Article 98 (July 2020), 15 pages. https://doi.org/10.1145/3386569.3392465

Kate Dunn, Dylan Wozniak O'Connor, Marjo Niemelä, and Gabriele Ulacco. 2016. Free Form Clay Deposition in Custom Generated Molds. In *Robotic Fabrication in Architecture, Art and Design 2016*, Dagmar Reinhardt, Rob Saunders, and Jane Burry (Eds.). Springer International Publishing, Cham, 316–325. https://doi.org/10.1007/978-3-319-26378-6_25

Shaun Edwards and Chris Lewis. 2012. Ros-industrial: applying the robot operating system (ros) to industrial applications. In *IEEE Int. Conference on Robotics and Automation, ECHORD Workshop*.

Gershon Elber and Elaine Cohen. 1994. Toolpath Generation for Freeform Surface Models. *Computer-Aided Design* 26, 6 (June 1994), 490–496. https://doi.org/10.1016/0010-4485(94)90070-1

Philippe Faraut and Charisse Faraut. 2013. *Figure Sculpting: Planes & Construction Techniques in Clay*. Number v. 1. PCF Studios.

Hsi-Yung Feng and Huiwen Li. 2002. Constant Scallop-Height Tool Path Generation for Three-Axis Sculptured Surface Machining. *Computer-Aided Design* 34, 9 (Aug. 2002), 647–654. https://doi.org/10.1016/s0010-4485(01)00136-1

Jared Friedman, Heamin Kim, and Olga Mesa. 2014. Experiments in Additive Clay Depositions. In *Robotic Fabrication in Architecture, Art and Design 2014*, Wes McGee and Monica Ponce de Leon (Eds.). Springer International Publishing, Cham, 261–272. https://doi.org/10.1007/978-3-319-04663-1_18

Gaël Guennebaud, Benoît Jacob, et al. 2010. Eigen v3. http://eigen.tuxfamily.org.

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. http://libigl.github.io/libigl/.

Ryan Luke Johns. 2017. Augmented Materiality: Modelling with Material Indeterminacy. In *Fabricate 2014* (dgo - digital original ed.). UCL Press, 216–223. https://doi.org/10.2307/j.ctt1tp3c5w.30

Cha-Soo Jun, Kyungduck Cha, and Yuan-Shin Lee. 2003. Optimizing Tool Orientations for 5-Axis Machining by Configuration-Space Search Method. *Computer-Aided Design* 35, 6 (May 2003), 549–566. https://doi.org/10.1016/S0010-4485(02)00077-5

Minjae Ko, Donghan Shin, Hyunguk Ahn, and Hyungwoo Park. 2019. InFormed Ceramics: Multi-Axis Clay 3D Printing on Freeform Molds. In *Robotic Fabrication in Architecture, Art and Design 2018*. Springer International Publishing, Cham, 297–308. https://doi.org/10.1007/978-3-319-92294-2_23

Odysseas Kontovourkis and George Tryfonos. 2018. Integrating Parametric Design with Robotic Additive Manufacturing for 3D Clay Printing: An Experimental Study. *ISARC Proceedings* (July 2018), 918–925.

Zhao Ma, Alex Walzer, Christian Schumacher, Romana Rust, Fabio Gramazio, Matthias Kohler, and Moritz Bächer. 2020. Designing Robotically-Constructed Metal Frame Structures. *Computer Graphics Forum* 39, 2 (2020), 411–422. https://doi.org/10.1111/cgf.13940

Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H Barr. 2003. Discrete differential-geometry operators for triangulated 2-manifolds. In *Visualization and mathematics III*. Springer, 35–57.

Alessandro Muntoni, Marco Livesu, Riccardo Scateni, Alla Sheffer, and Daniele Panozzo. 2018. Axis-Aligned Height-Field Block Decomposition of 3D Shapes. *ACM Trans. Graph.* 37, 5 (Oct. 2018), 169:1–169:15. https://doi.org/10.1145/3204458

Ioana Cristina Nan, Charlie Patterson, and Remo Pedreschi. 2016. Digital Materialization: Additive and Robotical Manufacturing with Clay and Silicone.

Huaishu Peng, Jimmy Briggs, Cheng-Yao Wang, Kevin Guo, Joseph Kider, Stefanie Mueller, Patrick Baudisch, and François Guimbretière. 2018. RoMA: Interactive Fabrication with Augmented Reality and a Robotic 3D Printer. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10/ghb9kd

Thiago Pereira, Szymon Rusinkiewicz, and Wojciech Matusik. 2014. Computational Light Routing. *ACM Transactions on Graphics* 33, 3 (June 2014), 1–13. https://doi.org/10.1145/2602140 arXiv:1204.6216v2

Philippe Faraut and Charisse Faraut. 2009. *Mastering Portraiture: Advanced Analyses of the Face Sculpted in Clay*. PCF Studios, Incorporated.

Ronald Rael and Virginia San Fratello. 2017. Clay Bodies: Crafting the Future with 3D Printing. *Architectural Design* 87, 6 (2017), 92–97. https://doi.org/10.1002/ad.2243

Universal Robots. 2015. The URScript programming language. *Universal Robots A/S, version* 3 (2015).

David Rosenwasser, Sonya Mantell, and Jenny Sabin. 2017. Clay Non-Wovens: Robotic Fabrication and Digital Ceramics. In *ACADIA 2017: DISCIPLINES & DISRUPTION (Proceedings of the 37th Annual Conference of the Association for Computer Aided Design in Architecture (ACADIA))*. CUMINCAD, 502–511.

Mathew Schwartz and Jason Prasad. 2013. RoboSculpt. In *Rob | Arch 2012*, Sigrid Brell-Çokcan and Johannes Braumann (Eds.). Springer Vienna, 230–237.

Ioan A. Şucan, Mark Moll, and Lydia E. Kavraki. 2012. The Open Motion Planning Library. *IEEE Robotics & Automation Magazine* 19, 4 (December 2012), 72–82. https://doi.org/10.1109/MRA.2012.2205651 https://ompl.kavrakilab.org.

Alan Sullivan, Huseyin Erdim, Ronald N. Perry, and Sarah F. Frisken. 2012. High Accuracy NC Milling Simulation Using Composite Adaptively Sampled Distance Fields. *Computer-Aided Design* 44, 6 (June 2012), 522–536. https://doi.org/10.1016/j.cad.2012.02.002

Rachel Tan and Stylianos Dritsas. 2016. Clay Robotics: Tool Making and Sculpting of Clay with a Six-Axis Robot. In *Living Systems and Micro-Utopias: Towards Continuous Designing, Proceedings of the 21st International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA 2016)*. CUMINCAD, 579–588.

Christophe Tournier and Emmanuel Duc. 2005. Iso-Scallop Tool Path Generation in 5-Axis Milling. *The International Journal of Advanced Manufacturing Technology* 25, 9 (May 2005), 867–875. https://doi.org/10.1007/s00170-003-2054-7

Matthew Trilsbeck, Nicole Gardner, Alessandra Fabbri, Matthias Hank Haeusler, Yannis Zavoleas, and Mitchell Page. 2019. Meeting in the Middle: Hybrid Clay Three-Dimensional Fabrication Processes for Bio-Reef Structures. *International Journal of Architectural Computing* 17, 2 (June 2019), 148–165. https://doi.org/10.1177/1478077119849655

Christian Weichel, John Hardy, Jason Alexander, and Hans Gellersen. 2015. ReForm. *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology - UIST '15* (2015), 93–102. https://doi.org/10.1145/2807442.2807451

Haisen Zhao, Hao Zhang, Shiqing Xin, Yuanmin Deng, Changhe Tu, Wenping Wang, Daniel Cohen-Or, and Baoquan Chen. 2018. DSCarver: Decompose-and-Spiral-Carve for Subtractive Manufacturing. *ACM Trans. Graph.* 37, 4 (July 2018), 137:1–137:14. https://doi.org/10.1145/3197517.3201338

Weihang Zhu and Yuan-Shin Lee. 2004. Five-Axis Pencil-Cut Planning and Virtual Prototyping with 5-DOF Haptic Interface. *Computer-Aided Design* 36, 13 (Nov. 2004), 1295–1307. https://doi.org/10.1016/j.cad.2004.01.013

Amit Zoran and Joseph A. Paradiso. 2013. FreeD: A Freehand Digital Sculpting Tool. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. Association for Computing Machinery, New York, NY, USA, 2613–2616. https://doi.org/10/ghbzgn