Autocomplete 3D Sculpting

MENGQI PENG, University of Hong Kong
JUN XING, University of Hong Kong and USC Institute for Creative Technologies
LI-YI WEI, University of Hong Kong and Adobe Research

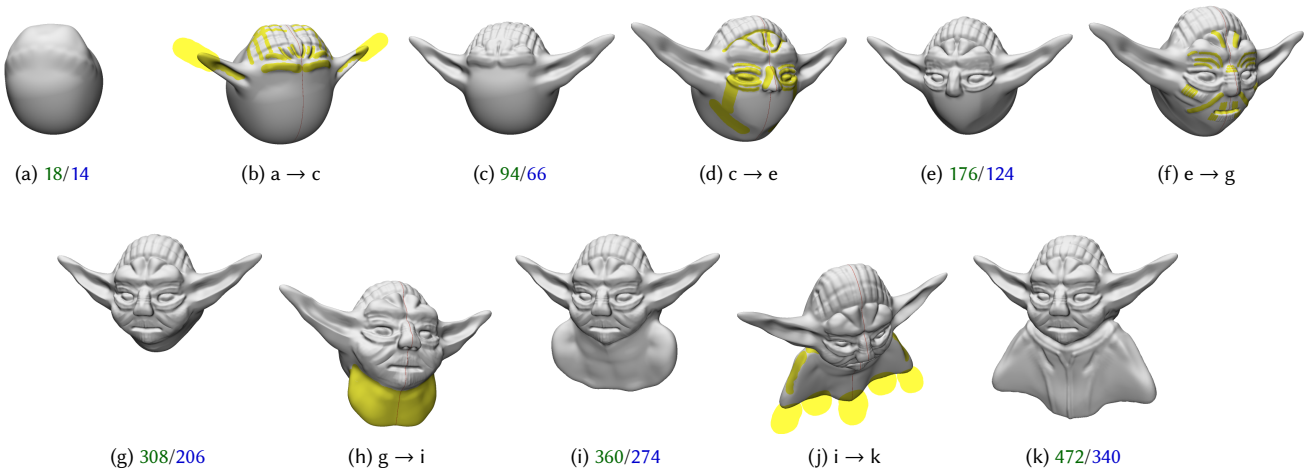


Fig. 1. *Our autocomplete 3D sculpting system.* With an authoring interface similar to common sculpting tools, our system detects potential repetitions and provides suggestions for which the users can accept, partially accept, or ignore. Shown here is an editing sequence with the number of cumulative manual and autocomplete strokes indicated below each stage, and in-between steps with some hints visualized in yellow colors, including overlapped/non-overlapped strokes for both surface and freeform types. Other functionalities include workflow clone and automatic camera view control. Please refer to the accompanying videos for recorded actions, including all edits, hints, and camera views.

Digital sculpting is a popular means to create 3D models but remains a challenging task. We propose a 3D sculpting system that assists users, especially novices, in freely creating models with reduced input labor and enhanced output quality. With an interactive sculpting interface, our system silently records and analyzes users' workflows including brush strokes and camera movements, and predicts what they might do in the future. Users can accept, partially accept, or ignore the suggestions and thus retain full control and individual style. They can also explicitly select and clone past workflows over output model regions. Our key idea is to consider *how* a model is authored via dynamic workflows in addition to *what* is shaped in static geometry. This allows our method for more accurate analysis of user intentions and more general synthesis of shape structures than prior workflow or geometry methods, such as large overlapping deformations. We evaluate our method via user feedbacks and authored models.

CCS Concepts: • **Human-centered computing** → **User interface design**; • **Computing methodologies** → **Shape modeling**;

Additional Key Words and Phrases: workflow, autocomplete, clone, beautification, sculpting, modeling, user interface

Authors' addresses: Mengqi Peng, University of Hong Kong; Jun Xing, University of Hong Kong, USC Institute for Creative Technologies; Li-Yi Wei, University of Hong Kong, Adobe Research.

ACM Reference Format:
Mengqi Peng, Jun Xing, and Li-Yi Wei. 2018. Autocomplete 3D Sculpting. *ACM Trans. Graph.* 37, 4, Article 132 (August 2018), 16 pages. https://doi.org/10.1145/3197517.3201297

1 INTRODUCTION

Digital sculpting is a popular means to create 3D models with diverse styles and organic shapes. The authoring practices often involves repetitive applications of large, overlapping deformations interleaved with small, detailed strokes. Such cumulative process can demand significant expertise and efforts, and be particularly daunting for novice users.

Significant research has been devoted to analyzing and synthesizing 3D geometry based on shapes [Chaudhuri and Koltun 2010; Funkhouser et al. 2004; Mitra et al. 2013; Takayama et al. 2011] or procedures [Emilien et al. 2015; Nishida et al. 2016]. These methods mainly focus on *what* the model is shaped instead of *how* it is authored, and thus cannot readily apply to iterative, dynamic sculpting of general shapes. The processes of how models are authored by users, termed workflows, contain rich information that can facilitate a variety of tasks such as visualization [Denning and Pellacini 2013; Denning et al. 2015], view selection [Chen et al. 2014],

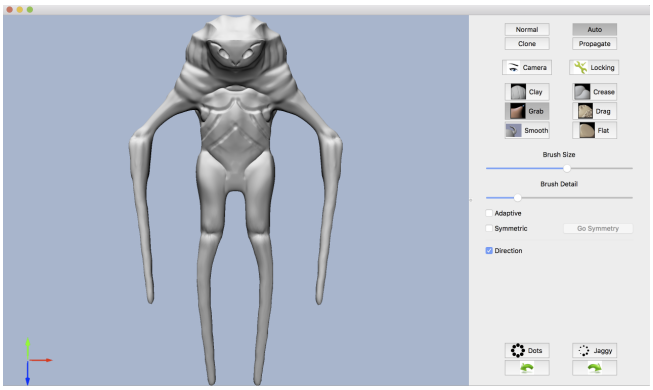


Fig. 2. *User interface of our system.* The interface consists of a sculpting canvas (left) and a widget panel (right). The widget panel provides the usual sculpting tools, brush parameters such as size, and mode controls unique to our autocomplete system.

and collaboration [Calabrese et al. 2016; Salvati et al. 2015]. However, it remains unclear whether and how such workflows can help automate interactive sculpting of 3D models.

We propose a 3D sculpting system that assists users in creating organic models with reduced input workload and enhanced output quality. With a brushing interface similar to existing sculpting tools, our system analyzes what users have done in the past and predicts what they might do in the future. The predictions are visualized as suggestions over the output model without disrupting user practices. Users can choose to accept, partially accept, or ignore the suggestions, and thus maintain full control. They can also select prior workflows from the model and clone over other regions.

The rich information contained in the workflows allows our method to outperform prior methods based on only geometry, such as handling overlapping strokes, automating vertex locking, reapplying clones, and maintaining intermediate strokes relationships even after significant geometry deformations by later strokes. Similar to existing sculpting tools, our interface provides *surface* strokes for local details such as displacements and *freeform* strokes for large scale changes such as extrusions. Our system is intended for users with varying levels of expertise and models of different types without requiring pre-existing geometry database or procedural rules. In addition to sculpting strokes, our method also considers camera movements, which are often repetitive, predictable, and correlate well with the underlying shapes and brush strokes [Chen et al. 2014].

Inspired by recent works on predictive user interfaces that analyze repetitions and workflows to assist 2D drawings [Kazi et al. 2012; Xing et al. 2014] and animations [Xing et al. 2015], our premise is that 3D sculpting often consists of repetitive user actions and shape structures, and thus both the input process and the output structure can be predictable.

However, there are several key differences between 2D sketching and 3D sculpting that prevent trivial extensions of prior methods [Xing et al. 2014, 2015] for our applications. 2D sketching operates on a simple 2D static planar canvas, the user strokes do not interact with one another, and remain there once placed. 3D sculpting dynamically deforms the base object shape, the later strokes can overlap and alter earlier ones including both visible/explicit mesh shapes and invisible/implicit stroke relationships, and similar input strokes can end up with dissimilar output structures due to intervening edits. To address these challenges, we synchronize all intermediate workflows with dynamic domain changes, and analyze *how* existing geometry shapes and dynamic workflows relate to one another to predict *what* the future workflows would be like, including both sculpting strokes and camera movements.

We conduct a pilot user study to show that our system can help users on both objective performance and subjective satisfaction for a variety of output models.

In summary, the main contributions of this paper are:

- A method that combines dynamic workflows and static geometry to reduce input labor and enhance output quality for interactive 3D modeling;
- An autocomplete user interface for 3D sculpting with features including hint, edit propagation, workflow clone, workflow lock, and camera control;
- Algorithms that analyze and synthesize sculpting workflows over general 3D shapes while adapting to dynamically changing base geometry and surrounding contexts.

2 PREVIOUS WORK

Data-driven and procedural modeling. Creating models from scratch is challenging, but similar objects or parts often already exist. Analyzing existing model geometry for novel synthesis has been an active area of research [Funkhouser et al. 2004; Liu et al. 2017; Mitra et al. 2013] encompassing a variety of topics, such as suggestion [Chaudhuri and Koltun 2010], repetition [Bokeloh et al. 2011], symmetry [Mitra et al. 2006], style [Lun et al. 2016], fabrication [Schulz et al. 2014], simulation [De Goes and James 2017], and functional interaction [Hu et al. 2016]. Common model structures can also be abstracted into procedural rules for offline or interactive modeling [Ebert et al. 2002; Emilien et al. 2015; Nishida et al. 2016].

The outputs of these methods are naturally limited by the scopes of the underlying data and procedures. Our method, in contrast, aims to assist users to explore and create models [Cohen-Or and Zhang 2016] in their individual styles and preferences.

Modeling interface. Popular modeling interfaces have been designed for mechanic models (e.g. AutoCAD) and organic shapes (e.g. ZBrush). Following the line of suggestive interfaces for 3D drawing and sketching [Fan et al. 2013; Igarashi and Hughes 2001; Tsang et al. 2004; Yue et al. 2017], our system aims for a traditional sculpting interface with enhancement in analyzing, predicting, and suggesting workflows. Inspired by Teddy [Igarashi et al. 1999], we design our system to be interactive and friendly to novice users.

Workflow-assisted authoring. Workflows [Nancel and Cockburn 2014] have been investigated for various authoring tasks in 2D image editing [Chen et al. 2011, 2016], sketching [Xing et al. 2014], and animation [Xing et al. 2015], as well as in 3D modeling such as texturing [Suzuki et al. 2017], visualization [Denning et al. 2011, 2015], revision control [Denning and Pellacini 2013], view selection [Chen et al. 2014], and collaboration [Salvati et al. 2015]. These

workflows can be recorded during authoring, or inferred a posteriori [Fu et al. 2011; Hu et al. 2013; Tan et al. 2015].

As reported in [Santoni et al. 2016], even for complex objects, the workflows often consist of predictable brush choices and operations. Our method analyzes 3D sculpting workflows to autocomplete potential repetitions.

Our method is mainly inspired by prior 2D autocomplete methods for sketches and animations [Xing et al. 2014, 2015]. However, these 2D autocomplete methods cannot be directly extended for 3D sculpting due to several major differences in design and mechanisms:

- Strokes: All 2D sketching and animation strokes lay on a plane. 3D sculpting strokes can lay on the surface (e.g. small-scale details) or extend inward/outward towards the surrounding space (e.g. large-scale freeform deformations).
- Domains: The base domains for sketching and animations remain a static 2D plane. The base domains for 3D sculpting, which are the current shapes, coevolve with the editing strokes.
- Inputs and outputs: In 2D sketching, input strokes exactly form the output content thus repetitions are easier to identify. In contrast, the 3D sculpting stroke types (freeform or surface), behaviors (overlap or separate), and relationships (under different shape changes) are more complex and thus more difficult to analyze.

Therefore, despite conceptual similarity to 2D autocomplete, our system is the first to address the above challenges unique to 3D sculpting for autocompletes.

3 DESIGN GOALS

Among the four design principles suggested by Schneiderman [2007] for creative support tools, "provide rich history-keeping" and "design with low thresholds, high ceilings, and wide walls" are most relevant to our context of 3D sculpting. To follow these principles, a core problem of our work is the relationship between user workflows and final outputs. Our premise is that how users create the models can provide helpful cues to design flexible and friendly sculpting user interfaces. To better understand the validity of the premise, and collect insights for our work design, we observe sculpting processes, including multiple online videos/tutorials and onsite observations, of users with different levels of expertise, ranging from novices to professionals. The insights and discussions led us to the following design goals of our sculpting system.

$\mathcal{D}_1$: Leverage common sculpting practices and preserve individual styles

Based on our observations, users, either novices or professionals, tend to sculpt in a spatial and temporal coherent fashion. They like to focus on one local region at a time, showing a preference for short strokes. These observations are consistent with prior studies such as [Chen et al. 2014; Santoni et al. 2016].

On the other hand, sculpting also exhibits high degrees of freeform and personal styles. Our observations show that individual users have their own styles and preferences, including stroke size, length, direction, shape, and pressure. In a nutshell, 3D sculpting is a process of accumulating strokes, thus individual preference significantly influences final output style.

Our assisted system, therefore, should leverage general sculpting practices while preserving individual styles.

$\mathcal{D}_2$: Reduce workload from repetitive operations

We observed that sculpting often contain repetitive stroke behaviors, either overlapped or independent ones, including larger scale base mesh formation and details additions, and the sculpting sessions often contain repetitive camera movements. Most users achieve these through manual operations. It is also common for users to repeat batch strokes from one region to other regions.

Some sculpting tools, such as ZBrush [Pixologic 2015] and Mudbox [Autodesk 2014], provide data-driven features like stamping and stenciling to place repetitive patterns. However, these might not be easy for non-experts to understand and control, compared to direct sculpting strokes. Users also need to find the intended patterns from existing data.

To reduce repetitive manual operations, our system should analyze user behaviors and intentions from their workflows.

$\mathcal{D}_3$: Provide fluid and intuitive user control

Complex 3D user interface control/configuration is one of the major reasons that make 3D sculpting challenging for most users, which generally requires years of experience to master. However, there are an increasing number of users joining the sculpting community. We observed that most users will find it difficult to learn sculpting if starting from tools which provide powerful but complex features and controls.

Therefore, another design goal is to provide our new features under a familiar framework and friendly user interface.

4 USER INTERFACE

To satisfy the design goals in Section 3, the user interface (Figure 2) of our prototype sculpting system follows the brush model as in popular digital sculpting tools such as Blender [Blender Foundation 2016], Sculptris [Pixologic 2011], Zbrush [Pixologic 2015], MeshMixer [Schmidt and Singh 2010], etc. Our system supports *surface* strokes for local details, and *freeform* strokes for large scale deformations. Users can sculpt as usual while our system silently records and analyzes the sculpting workflows. All stroke operations can be combined with the main functions: hint, workflow clone, camera control, and other features such as workflow lock.

4.1 Hint

Our system automatically analyzes users' sculpting workflows on the fly and predicts what they might sculpt in the near future. These predictions are suggested as hints on our user interface. Different from 2D sketching, 3D sculpting has more diverse stroke effects depending on types (*surface* or *freeform*) and behaviors (*independent* or *overlapped*/*connected*). All combinations of hints (type × behavior) are supported by our system.

Figure 3 shows an example where the user is sculpting *independent* detailed strokes on an object. As more strokes are added, our system automatically analyzes the past strokes and predicts what strokes the user might want to perform next, as shown in Figure 3b. The suggestions are shown transparently over the object surface. Users can ignore the suggestions and continue sculpting as usual,

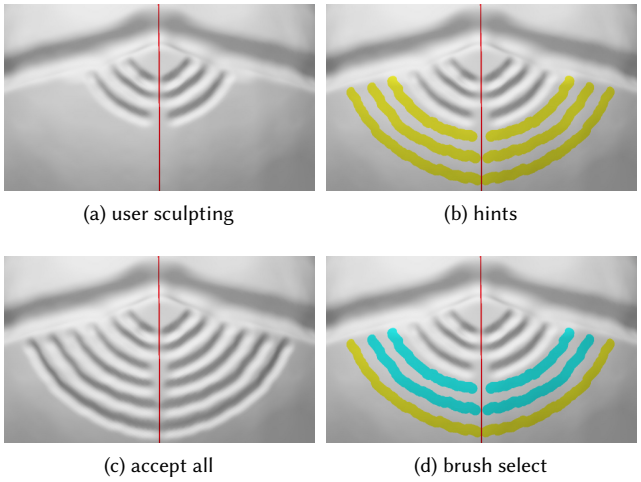


Fig. 3. *Independent hint examples.* During user sculpting (a), our system suggests potential future repetitions as displayed in transparent yellow (b). Users can ignore the hints and continue sculpting, accept all hints via a hot key (c), or partially select a subset of the *independent* hints (d) with a selection brush as shown in transparent blue.

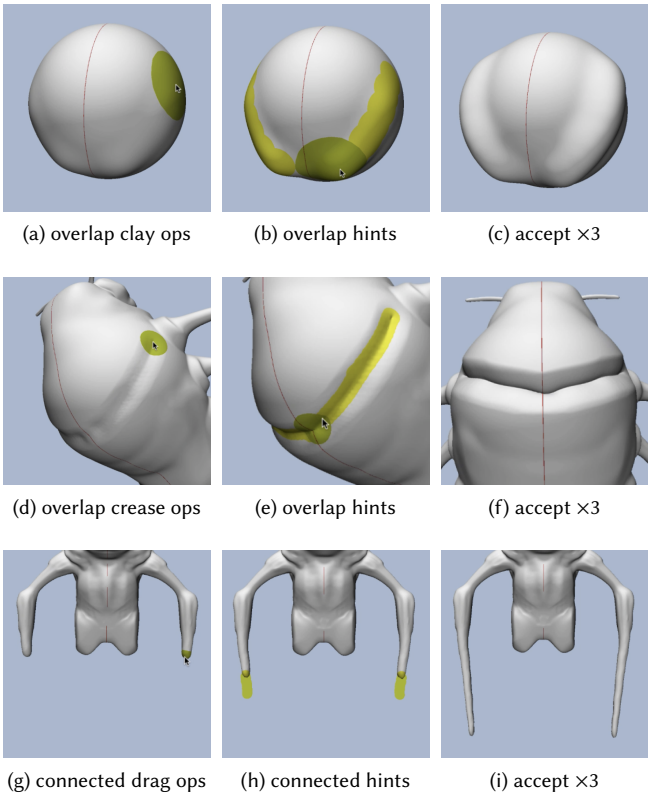


Fig. 4. *Overlapped and connected hint examples.* For such *overlapped* surface hints and *connected* long freeform stroke hints, user can select the number of hints by pressing the number of continuous hints to be accepted. For example, users can achieve the effect in (c), (f), and (i) at one go with selected number of three. The accept hot key is the same as in Figure 3c.

accept all the suggestions via a hot key (Figure 3c), or partially select the suggestions (Figure 3d).

As a main difference from 2D sketching, sculptors often iterate over the same area [Santoni et al. 2016], including multiple *overlapped* strokes for large deformation (Figure 4a) or refinement (Figure 4d), as well as connected (Figure 4g) strokes for long freeform feature formation. We also provide such hints as shown in Figure 4. Users can customize the number of continuous overlapped repetitions to be accepted for faster feature creation, such as three in Figure 4. The suggestions are continuously updated in real-time according to user inputs.

Our hint function can also propagate a single source edit to multiple targets with individual adaptation by considering the contextual relationships among surface or freeform strokes. It is similar to the "find and replace" features of modern IDEs and graphical illustrations [Kurlander and Bier 1988; Kurlander and Feiner 1992a,b]. Examples are shown in Figure 5. Users can turn this feature on or off independent from local hints.

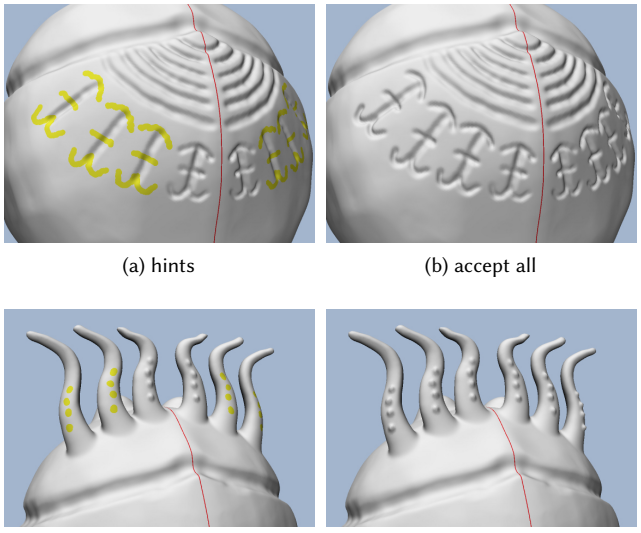
4.2 Workflow Clone

The clone tool is common among interactive content authoring systems. Prior methods mostly clone output content such as illustrations [Kazi et al. 2014, 2012], images [Pérez et al. 2003], textures [Sun et al. 2013], or geometry [Takayama et al. 2011]. The methods in [Xing et al. 2014] can clone intermediate sketch workflows. Our system allows users to clone sculpting workflows with more information and options than cloning static geometry, with clone effects that adapt better to the user intentions and geometry shapes. Via our brushing interface, users can select source and target regions, and parameters such as positions, sizes, and directions. Similar to prior clone tools [Kloskowski 2010], our system previews the clone outcomes for which users can accept or ignore. An example is shown in Figure 6. Furthermore, our workflow clone can be applied to already cloned regions, which is difficult to achieve via geometry clone.

4.3 Camera Control

Camera control is an integral part of 3D content authoring such as drawing [Ortega and Vincent 2014] and sculpting [Chen et al. 2014], which can also be quite tedious and repetitive in addition to brush strokes. Fortunately, similar to sculpting operations, camera controls also tend to be predictable [Chen et al. 2014].

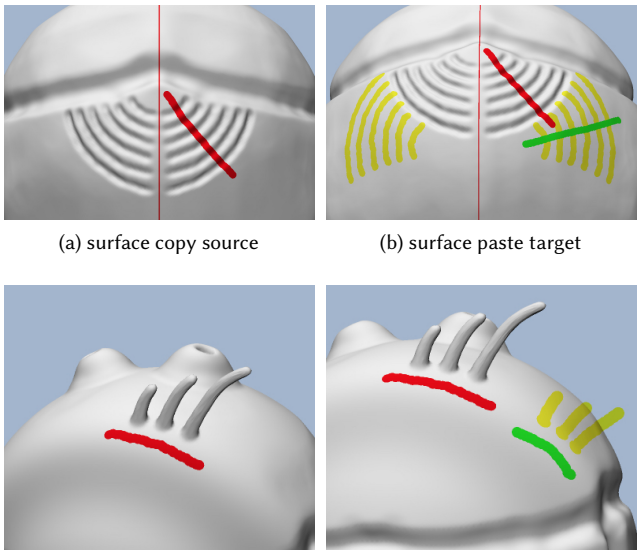
We provide a basic camera control mode that automatically moves the viewpoints along with the suggested hints (Section 4.1). This choice is inspired by the observation in [Chen et al. 2014] that users often view the target sculpting regions frontal-parallel, and thus more natural and less disorienting for users than other forms of camera control. Users can quickly press a hotkey to return to the original viewpoint. They can also turn this mode on or off depending on their preferences. One example is shown in Figure 7.



(a) hints (b) accept all

(c) hints (d) accept all

Fig. 5. *Edit propagation examples.* The hints (yellow) can appear in multiple compatible targets from a single editing source (left), from which the users can accept, ignore, or brush select. This applies to both surface (top) and freeform (bottom) strokes.

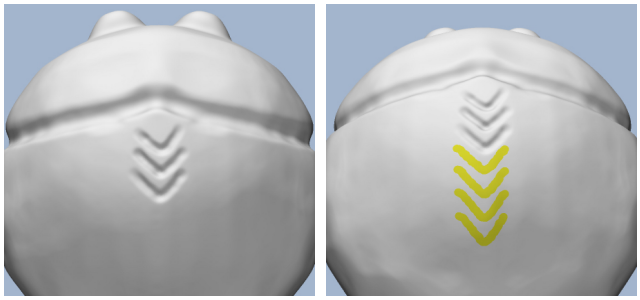


(a) surface copy source (b) surface paste target

(c) freeform copy source (d) freeform paste target

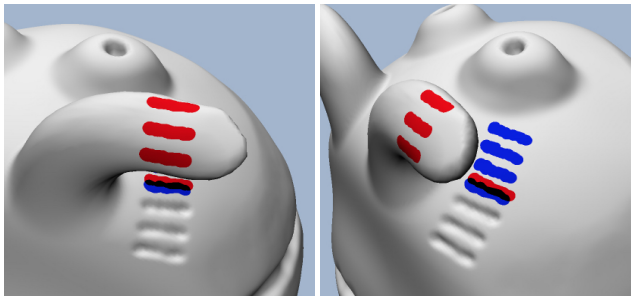
Fig. 6. *Workflow clone examples.* The red/green strokes mark the clone sources/targets. The clone results are previewed as yellow in (b) and (d) for the users to partially or fully accept. The clone can be applied to both surface (top) and freeform (bottom) strokes, and can adapt to different clone source/target sizes/lengths and different source/target base geometry.

4.4 Additional Features



(a) before (b) after

Fig. 7. *Camera control example.* The camera automatically adjusts the viewpoint from (a) according to the suggested strokes in (b).

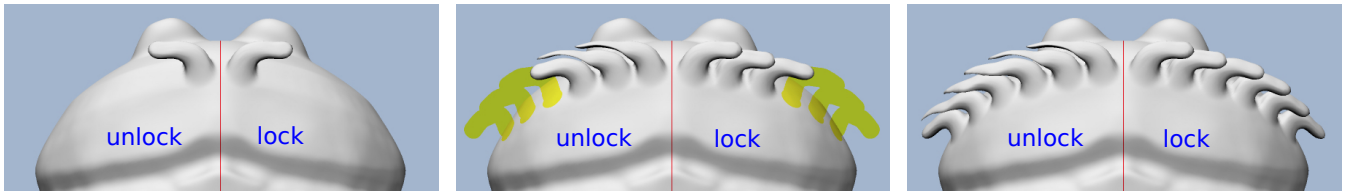


(a) sculpting viewpoint (b) shifted viewpoint

Fig. 8. *Occlusion example.* The blue and red strokes are predictions from the three previous sculpting strokes (bottom), with and without considering occlusion. (a) is the original sculpting view, and (b) shifts the view to show the predicted strokes underneath the occlusion.

Occlusion. Occlusion handling is another major difference between 2D and 3D content authoring such as painting [Fu et al. 2010]. Our system considers geometry occlusion for sculpting strokes predictions and suggestions. As exemplified in Figure 8, with suggestions predicted on the view-plane by [Xing et al. 2014, 2015], even simple surface propagation may reside on the wrong part of the occluding geometry, as shown in the red strokes. Our system considers the underlying shape and propagates the predictions more accurately, as shown in the blue strokes.

Lock. Locking is an option under some digital sculpting tools (e.g. [Blender Foundation 2016]) to keep some parts of the model fixed while users manipulating other parts. This feature, while useful, can be tedious and difficult for users, especially novices. In addition to future operations, our system can also predict what might need to be locked based on workflows, as exemplified in Figure 9b; such scenarios can be challenging to analyze via geometry only. This locking mechanism can be applied to not only existing geometry as described above but also future geometry predicted by our system which cannot be manually locked, as shown in Figure 9c. Users can easily enable or disable the locking effects via a hot key and thus maintain full control.



(a) initial manual stroke (b) more manual strokes and hints (c) after accepting hints

Fig. 9. *Workflow lock.* After a manual stroke in (a) under the symmetry mode, the user went on to place two more strokes in (b). The yellow parts indicate suggested hints. For comparison, the left side has no workflow lock; notice how earlier strokes can be unintentionally deformed by the later strokes. Our workflow lock can prevent this from happening for both existing geometry and accepted hints, as shown on the right side of (c). Note that the predicted strokes (yellow) are always correct, with or without workflow lock. However, when the user accepted the hints, they will play out in the workflow order as in manual strokes. Thus, without workflow lock, later hint strokes can still deform earlier hint strokes.

Table 1. *Notations used in our algorithm descriptions.*

Symbol	Meaning
$\mathbf{b}$	brush stroke
s	a sample of a brush stroke $\mathbf{b}$
$\mathbf{u}(s)$	all parameters of sample s
$\mathbf{p}(s)$	3D position of sample s
$\ddot{\mathbf{p}}(s)$	locally parameterized position of sample s
$\mathbf{a}(s)$	appearance parameters of sample s
$\mathbf{t}(s)$	temporal parameters of sample s
$\mathbf{n}(s)$	spatial-temporal neighborhood of sample s
$\mathbf{n}(\mathbf{b})$	spatial-temporal neighborhood of brush stroke $\mathbf{b}$
$\mathbf{b}'$	a neighborhood stroke of brush stroke $\mathbf{b}$
s'	a neighborhood sample of sample s
$\hat{\mathbf{u}}(s', s)$	differential of $\mathbf{u}(s')$ and $\mathbf{u}(s)$
$\mathbf{I}$	input sequences of sculpting strokes
$\mathbf{b}_o$	next predicted stroke
$\mathbf{b}_{o,i}$	initialization for $\mathbf{b}_o$
$\overline{\mathbf{u}}(\mathbf{b}_o)$	contextual prediction for $\mathbf{b}_o$

5 METHOD

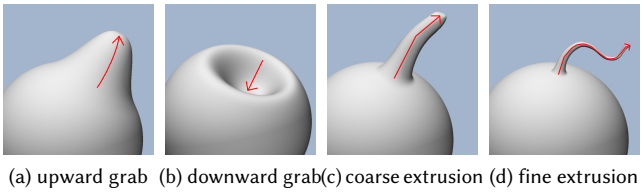
We describe our algorithms behind the autocomplete sculpting interface in Section 4. Notations are summarized in Table 1 for easy reference.

5.1 Representation

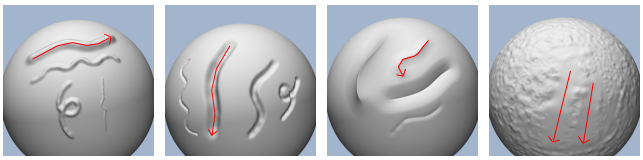
Stroke. Our system supports two main types of brush strokes as in common digital sculpting systems: *surface* strokes for small scale displacements (e.g. clay, crease, smooth, flatten), and *freeform* strokes for larger scale shape deformation (e.g. drag and grab).

Users sculpt via a tactile interface (e.g. pen/touch tablet) with different tool types and parameter settings. Each sculpting brush will determine the affected regions on the object surface to undergo various local transformations, including freeform upward/downward grabbing, coarse/fine dragging, and on-surface relief, engraving, crease, smoothing, flattening, etc. These main sculpting effects are supported by our system as shown in Figure 10.

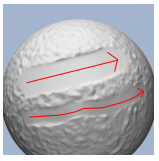
Unlike 2D sketching where input strokes directly form the final content with the underlying plane being static, 3D sculpting is a dynamic transformation process with the object shapes created via various operations with significant different effects.



(a) upward grab (b) downward grab (c) coarse extrusion (d) fine extrusion

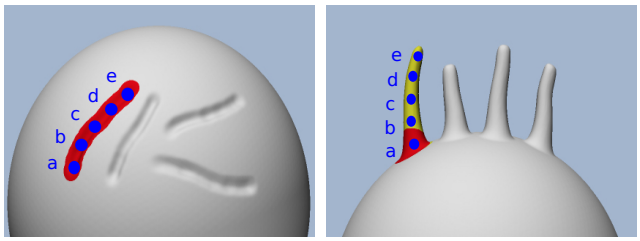


(e) relief (f) engrave (g) crease (h) smooth



(i) flatten

Fig. 10. *3D sculpting stroke effects and user controls.* We support a rich toolset similar to those popular sculpting tools. The red lines indicate user input stroke paths for the respective effects. Notice the relationships between invisible input strokes (red lines with arrows indicating the directions) and actual visible shape changes.



(a) surface stroke (b) freeform stroke

Fig. 11. *Brush stroke types.* A surface stroke (a) has all samples on the object surface, such as the 5 surface samples s_a, s_b, s_c, s_d, and s_e. A freeform stroke (b) has the first sample s_a on the object but the rest for 3D movements such as extrusion $s_a \rightarrow s_b \rightarrow s_c \rightarrow s_d \rightarrow s_e$.

Sample. We represent each brush stroke $\mathbf{b}$ as a collection of point samples $\{s\}$. Each sample s is associated with a set of attributes $\mathbf{u}$:

$$\mathbf{u}(s) = (\mathbf{p}(s), \mathbf{a}(s), \mathbf{t}(s)), \tag{1}$$

where $\mathbf{p}(s)$ is the 3D position of s, $\mathbf{a}(s)$ is a set of appearance parameters (such as size, type, and pressure) and geometry signatures (such as normal and curvature), $\mathbf{t}(s)$ indicates temporal parameters that include the global time stamp and a sample-id for the relative position within the stroke.

As shown in Figure 11, for a surface stroke $\mathbf{b}$, its samples' positions $\mathbf{p}(\mathbf{b}) = \{\mathbf{p}(s)\}_{s\in\mathbf{b}}$ all lay on the object surface; while for a freeform stroke, $\mathbf{p}(\mathbf{b})$ consists of two parts: the first sample is on the surface, and the rest ($\|\mathbf{p}(\mathbf{b})\|-1$) samples are in 3D free space with movement directions controlled by the users.

Mesh. We adopt a mesh-based representation with two operators, sculpt $\mathbf{c}$ and mesh $\mathbf{m}$, to support geometry and topology changes. A mesh $\mathcal{M}$ is represented by a set of elements including vertices, edges, and faces. Each sculpt operator $\mathbf{c}$ in Figure 10 applies specific geometry transformation to mesh elements, such as vertex positions, within a finite support defined by the brush radius, as in traditional sculpting brush pipeline [Calabrese et al. 2017]. A mesh operator $\mathbf{m}$ can change the underlying mesh resolution and topology by adding or removing mesh elements. The result of each brush stroke over $\mathcal{M}$ is the joint effect of $\mathbf{c}$ with $\mathbf{m}$:

$$\mathcal{M} \leftarrow (\mathbf{c} \otimes \mathbf{m})(\mathcal{M}), \tag{2}$$

where $\otimes$ combines $\mathbf{c}$ and $\mathbf{m}$ to achieve Blender-Dyntopo-like or Sculptris-like adaptive tessellation effect [Hernandez 2011], as shown in Figure 12.

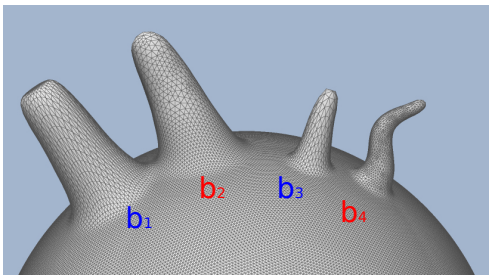


Fig. 12. *Mesh sculpting effects.* A sculpt operator $\mathbf{c}$ such as drag with inactive $\mathbf{m}$ can influence mesh geometry but not topology as shown in $\mathbf{b}_1$ and $\mathbf{b}_3$ with different brush radii. The joint effect of $\mathbf{c}$ with active $\mathbf{m}$ can change mesh resolution and connectivity as shown in $\mathbf{b}_2$ and $\mathbf{b}_4$.

5.2 Measurement

A core component for our method is to measure similarity between 3D brush strokes based on their spatial-temporal neighborhoods. This similarity in turn enables our method to detect repetitions, suggest future edits, clone workflows, and auto-lock strokes. However, unlike [Xing et al. 2014, 2015] where the underlying domain is a fixed 2D plane (drawing canvas), our base domain is a 3D object under dynamic modification. Thus, all definitions of neighborhood and similarity must be conditioned on 3D object surfaces.

Neighborhood. We define the neighborhood $\mathbf{n}(s)$ of a sample s as the set of all samples within its spatial-temporal vicinity analogous to the spatial-temporal neighborhoods in [Ma et al. 2013]. Each spatial neighborhood is oriented with respect to a local frame $\mathbf{o}$ associated with s. The temporal neighborhood is causal and contains only samples placed before s.

Brush strokes are composed of samples and can capture the high-level relationships between one another. Thus we use brush strokes as the fundamental units for sculpting workflow analysis and synthesis. The neighborhood of a stroke $\mathbf{b}$ is defined as the union of its sample neighborhoods:

$$\mathbf{n}(\mathbf{b}) = \bigcup_{s\in\mathbf{b}} \mathbf{n}(s) \tag{3}$$

Similarity. For each neighborhood sample $s' \in \mathbf{n}(s)$, we define its differential with respect to s as:

$$\hat{\mathbf{u}}(s', s) = \begin{pmatrix} w_{\mathbf{p}}\hat{\mathbf{p}}(s', s) \\ w_{\mathbf{a}}\hat{\mathbf{a}}(s', s) \\ w_{\mathbf{t}}\hat{\mathbf{t}}(s', s) \end{pmatrix}, \tag{4}$$

where $\hat{\mathbf{p}}$, $\hat{\mathbf{a}}$, and $\hat{\mathbf{t}}$ represent the sample pair differentials in position $\mathbf{p}$, appearance $\mathbf{a}$, and temporal parameters $\mathbf{t}$ defined in Equation (1), and $w_{\mathbf{p}}$, $w_{\mathbf{a}}$, $w_{\mathbf{t}}$ are the corresponding scalar weightings.

We compute the sample position differentials $\hat{\mathbf{p}}(s', s)$ via:

$$\hat{\mathbf{p}}(s', s) = \ddot{\mathbf{p}}(s') - \ddot{\mathbf{p}}(s), \tag{5}$$

where $\ddot{\mathbf{p}}(s)/\ddot{\mathbf{p}}(s')$ is the local position of s/s' with frame $\mathbf{o}(s)$ as described in Section 5.3 and relates to the global $\mathbf{p}(s)$ via a coordinate transformation.

From Equation (4), we define the differential between two strokes $\mathbf{b}'$ and $\mathbf{b}$ via their constituent samples:

$$\hat{\mathbf{u}}(\mathbf{b}', \mathbf{b}) = \{\hat{\mathbf{u}}(s', s) | s' = m(s) \in \mathbf{b}', s \in \mathbf{b}\}, \tag{6}$$

where m is the matching sample computed via the Hungarian algorithm as in [Ma et al. 2013; Xing et al. 2015].

From Equation (6), we can compute the distance between two stroke neighborhoods $\mathbf{n}(\mathbf{b}_o)$ and $\mathbf{n}(\mathbf{b}_i)$ as follows:

$$\begin{aligned} \|\mathbf{n}(\mathbf{b}_o) - \mathbf{n}(\mathbf{b}_i)\|^2 &= \|\hat{\mathbf{u}}(\mathbf{b}_o, c_o) - \hat{\mathbf{u}}(\mathbf{b}_i, c_i)\|^2 \\ &+ \sum_{\mathbf{b}'_o\in\mathbf{n}(\mathbf{b}_o), \mathbf{b}'_i\in\mathbf{n}(\mathbf{b}_i)} \|\hat{\mathbf{u}}(\mathbf{b}'_o, \mathbf{b}_o) - \hat{\mathbf{u}}(\mathbf{b}'_i, \mathbf{b}_i)\|^2, \end{aligned} \tag{7}$$

where the first term measures the distance between the two strokes $\mathbf{b}_o$ and $\mathbf{b}_i$ with respect to their central samples c_o and c_i:

$$\hat{\mathbf{u}}(\mathbf{b}, c) = \{\hat{\mathbf{u}}(s, c), s \in \mathbf{b}\}, \tag{8}$$

and the second term computes the distances between their neighborhood strokes with respect to $\mathbf{b}_o$ and $\mathbf{b}_i$. Thus, Equation (7) is computed in relative instead of absolute coordinates. The stroke pairs $\mathbf{b}'_o$ and $\mathbf{b}'_i$ are matched via the Hungarian algorithm as well.

5.3 Parameterization

Surface stroke parameterization. We extend the stroke parameterization method in [Schmidt 2013] for our surface strokes. Each surface stroke is parameterized by the surface normal as the z-direction and the stroke path as the y-direction measured by the geodesic arc-length t. The x-direction is measured by the geodesic distance d. We then apply the single-pass forward propagation [Schmidt 2013]

to estimate the parametrization for any sample s within distance r of the stroke, as illustrated in Figure 13.

$$\ddot{\mathbf{p}}(s) = \mathcal{P}_s(s) = (t_s, d_s) \tag{9}$$

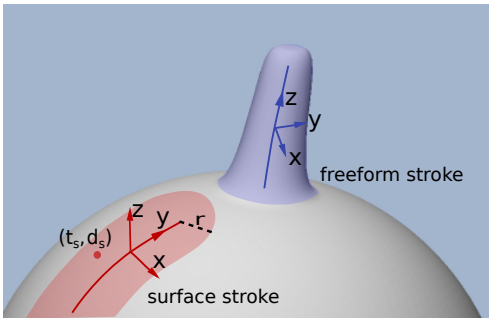


Fig. 13. *Stroke parameterizations.*

Freeform stroke parameterization. Unlike the surface strokes, freeform strokes do not adhere to the object surfaces. Thus the method in [Schmidt 2013] cannot directly apply. However, we can extend it into the freeform space as follows. We use the stroke path as the z-direction similar to the y-direction for the surface strokes, parameterized by arc-length. The cross-product of the z-direction and the camera look-at direction (non-parallel to z-direction for sculpting) for each stroke sample point forms the y-direction, and the x-direction is the cross product of y and z directions. This is illustrated in Figure 13. Unlike surface stroke parameterization which is 2D, the freeform stroke parameterization is 3D:

$$\ddot{\mathbf{p}}(s) = \mathcal{P}_f(s) = (x_s, y_s, z_s) \tag{10}$$

5.4 Synthesis

In order to synthesize the predictions interactively, we extend the texture optimization methodology [Kwatra et al. 2005; Ma et al. 2013; Xing et al. 2014] with particular focus on speed and geometry. With $\mathbf{I}$ as the current sequences of strokes ordered by their time-stamps, we synthesize the next stroke $\mathbf{b}_o$ by minimizing the following energy:

$$\mathbf{E}(\mathbf{b}_o; \mathbf{I}) = \min_{\mathbf{b}_i \in \mathbf{I}} |\mathbf{n}(\mathbf{b}_o) - \mathbf{n}(\mathbf{b}_i)|^2 + \Psi(\mathbf{b}_o) + \Theta(\mathbf{b}_o) \tag{11}$$

The first term identifies the existing $\mathbf{b}_i$ with similar neighborhood to $\mathbf{b}_o$, as measured in Equation (7). The second term corresponds to the context constraint predictions $\overline{\mathbf{u}}(\mathbf{b}_o)$ calculated from our workflow analysis in Section 5.5. The last term denotes optional, applicant-dependent specifications that can be supplied by the users.

As summarized in Algorithm 1, to synthesize the next predicted stroke $\mathbf{b}_o$ with good quality, our solver goes through three main steps: initialization in Section 5.4.1 provides a set of candidate stroke $\{\mathbf{b}_{o,i}\}$; then each $\mathbf{b}_o \in \{\mathbf{b}_{o,i}\}$ goes through the search step in Section 5.4.2 and the assignment step in Section 5.4.3. The one which has the least energy in Equation (11) will be considered as most suitable and selected to be the predicted stroke.

To predict multiple instead of only one stroke as described above, similar to other suggestive systems like [Xing et al. 2014], we can include predicted strokes into $\mathbf{I}$ for incremental continuous iterations.

function $\mathbf{b}_o \leftarrow$ *PredictNextStroke*($\mathbf{I}$)
  // $\mathbf{b}_o$: *predicted next stroke*
  // $\mathbf{I}$: *input sequence of strokes ordered by time-stamps*
  $\{\mathbf{b}_{o,i}\} \leftarrow$ Initialize($\mathbf{I}$) // *Section 5.4.1*
  foreach $\mathbf{b}_o \in \{\mathbf{b}_{o,i}\}$ **do**
    $\{\mathbf{b}_i\} \leftarrow$ Search($\mathbf{b}_o$, $\mathbf{I}$) // *Section 5.4.2*
    $\overline{\mathbf{u}}(\mathbf{b}_o) \leftarrow$ ContextAnalysis($\{\mathbf{b}_i\}$, $\mathbf{b}_o$) // *Section 5.5*
    $\Psi(\mathbf{b}_o), \Theta(\mathbf{b}_o) \leftarrow$ Assign($\overline{\mathbf{u}}(\mathbf{b}_o)$, $\mathbf{b}_o$) // *Section 5.4.3*
  end
  $\mathbf{b}_o \leftarrow \arg\min_{\mathbf{b}_o \in \{\mathbf{b}_{o,i}\}} \mathbf{E}(\mathbf{b}_o; \mathbf{I})$ // *Equation* (11)
  return $\mathbf{b}_o$
end function

function $\{\mathbf{b}_{o,i}\} \leftarrow$ *Initialize*($\mathbf{I}$) // *Section 5.4.1*
  $\mathbf{b}'_o \leftarrow$ last sculpting stroke in $\mathbf{I}$
  $\{\mathbf{b}'_i\} \leftarrow$ candidate matching strokes of $\mathbf{b}'_o$
  foreach $\mathbf{b}'_i \in \{\mathbf{b}'_i\}$ **do**
    $\mathbf{b}_i \leftarrow$ next stroke of $\mathbf{b}'_i$
    $\hat{\mathbf{u}}(\mathbf{b}_{o,i}, \mathbf{b}'_o) \leftarrow \hat{\mathbf{u}}(\mathbf{b}_i, \mathbf{b}'_i)$
    $\mathbf{b}_{o,i} \leftarrow \arg\min_{\mathbf{b}_{o,i}} \mathbf{E}(\mathbf{b}_{o,i}; \mathbf{b}'_o, \mathbf{b}_i, \mathbf{b}'_i)$ // *Equation* (13)
  end
  return $\{\mathbf{b}_{o,i}\}$
end function

function $\{\mathbf{b}_i\} \leftarrow$ *Search*($\mathbf{b}_o$, $\mathbf{I}$) // *Section 5.4.2*
  $\{\mathbf{b}^t_i\} \leftarrow$ temporal matching $\mathbf{b}_o$, $\mathbf{I}$
  $\{\mathbf{b}_i\} \leftarrow$ spatial filtering on $\{\mathbf{b}^t_i\}$
  return $\{\mathbf{b}_i\}$
end function

function *Assign*($\overline{\mathbf{u}}(\mathbf{b}_o)$, $\mathbf{b}_o$) // *Section 5.4.3*
  if $\overline{\mathbf{u}}(\mathbf{b}_o)$ **then** // *explicit or implicit context detected*
    $\Psi(\mathbf{b}_o) \leftarrow$ Equation (15)
  endif
  if extra sculpting effects configured **then**
    $\Theta(\mathbf{b}_o) \leftarrow$ optional specifications $\mathbf{c}(\mathbf{b}_o)$
  endif
end function

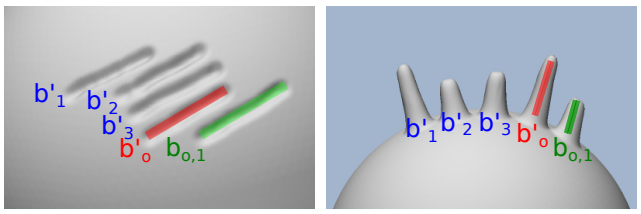
Pseudocode 1. *Overview of synthesis algorithm.*

5.4.1 Initialization. We initialize future strokes based on local similarity with the existing strokes. For the last sculpted stroke $\mathbf{b}'_o$, we identify a candidate set of matching strokes $\{\mathbf{b}'_i\}$ by measuring the local neighborhood similarity. Each $\mathbf{b}'_i$ provides an initialization $\mathbf{b}_{o,i}$ via its next stroke $\mathbf{b}_i$:

$$\hat{\mathbf{u}}(\mathbf{b}_{o,i}, \mathbf{b}'_o) = \hat{\mathbf{u}}(\mathbf{b}_i, \mathbf{b}'_i) \tag{12}$$

Each $\mathbf{b}_{o,i}$ is computed depending on the brush stroke type – surface or freeform, due to their different parameterizations as described in Section 5.3. For surface strokes, Equation (12) is computed on the local surface parameterization. For freeform strokes, Equation (12) is computed by a two-step process: deciding the starting point on the surface, followed by the freeform space movement from the starting point. More specifically, each $\mathbf{b}'_i$ provides relative

sample position differential with its next stroke $\mathbf{b}_i$ per its brush type as in Equation (5). Then within the local parameterization space of $\mathbf{b}'_o$, these relative differential will be transformed to compute initialized sample positions of $\mathbf{b}_{o,i}$. This is visualized in Figure 14.



(a) surface stroke initialization (b) freeform stroke initialization

Fig. 14. *Synthesis initialization.* For both (a) and (b), the three blue strokes $\mathbf{b}'_{1,2,3}$ are placed in order, before the current stroke $\mathbf{b}'_o$ shown in red. Each of $\mathbf{b}'_{1,2,3}$ can provide a prediction based on its next stroke $\mathbf{b}_1 = \mathbf{b}'_2$, $\mathbf{b}_2 = \mathbf{b}'_3$, $\mathbf{b}_3 = \mathbf{b}'_o$, and $\mathbf{b}'_o$ via Equation (12). For example, the green stroke $\mathbf{b}_{o,1}$ is predicted from $\mathbf{b}'_1$ via $\hat{\mathbf{u}}(\mathbf{b}_{o,1}, \mathbf{b}'_o) = \hat{\mathbf{u}}(\mathbf{b}_1 = \mathbf{b}'_2, \mathbf{b}'_1)$.

We then improve each $\mathbf{b}_{o,i}$ further via neighborhood optimization, as follows. For clarity, we use $\mathbf{b}_o$ as the variable to optimize $\mathbf{b}_{o,i}$ for each matching $\mathbf{b}'_i$ of $\mathbf{b}'_o$, by minimizing the energy:

$$\mathbf{E}(\mathbf{b}_o; \mathbf{b}'_o, \mathbf{b}_i, \mathbf{b}'_i) = \sum_{s_o \in \mathbf{b}_o} \sum_{s'_o \in \mathbf{b}'_o} \kappa(s_i, s'_i) \left| \mathbf{u}(s_o) - \mathbf{u}(s'_o) - \hat{\mathbf{u}}(s_i, s'_i) \right|^2$$
$$s_i = m(s_o) \in \mathbf{b}_i, \;\; s'_i = m(s'_o) \in \mathbf{b}'_i \tag{13}$$

where $\kappa(s_i, s'_i)$ is a weighting parameter inspired by [Ma et al. 2013] for Gaussian falloff with σ_p set to 10:

$$\kappa(s_i, s'_i) = \exp\left(\frac{-\left| \mathbf{p}(s_i) - \mathbf{p}(s'_i) \right|^2}{\sigma_p} \right) \tag{14}$$

5.4.2 Search. This step aims to minimize the first term in Equation (11). For each initialization $\mathbf{b}_o \in \{\mathbf{b}_{o,i}\}$ obtained during the initialization step, within its local spatial-temporal window, we search for the matching strokes $\{\mathbf{b}_i\}$ whose neighborhood are similar to $\mathbf{n}(\mathbf{b}_o)$ by measuring the neighborhood similarity in Equation (7).

Similar to [Xing et al. 2014], we perform temporal matching followed by spatial matching instead of matching with the whole temporal-spatial neighborhood to accelerate the searching process. In the first step, we conduct temporal matching to search the candidate strokes. Instead of selecting only one best match, for more robust optimization we search for multiple candidates $\{\mathbf{b}^t_i\}$ whose neighborhood dissimilarity $|\mathbf{n}(\mathbf{b}_o) - \mathbf{n}(\mathbf{b}_i)|^2$ is lower than $2\,|\mathbf{n}(\mathbf{b}_o) - \mathbf{n}(\mathbf{b}'')|^2$, where $\mathbf{b}''$ has the lowest dissimilarity value. From this candidate set, similarly, we use spatial neighborhood for further filtering/matching to get the finalized $\{\mathbf{b}_i\}$.

5.4.3 Assignment. This step finds the optimal $\mathbf{b}^*_o \in \{\mathbf{b}_{o,i}\}$ that minimizes the energy in Equation (11), given the $\{\mathbf{b}_i\}$ identified for each $\mathbf{b}_o \in \{\mathbf{b}_{o,i}\}$ for the first term in the search step.

The second term in Equation (11) considers the prediction $\bar{\mathbf{u}}(\mathbf{b}_o)$ for the properties of $\mathbf{b}_o$ after performing context analysis (Section 5.5) on the candidate strokes $\{\mathbf{b}_i\}$ extracted in the search step. We utilize $\bar{\mathbf{u}}(\mathbf{b}_o)$ as an extra constraint for synthesis:

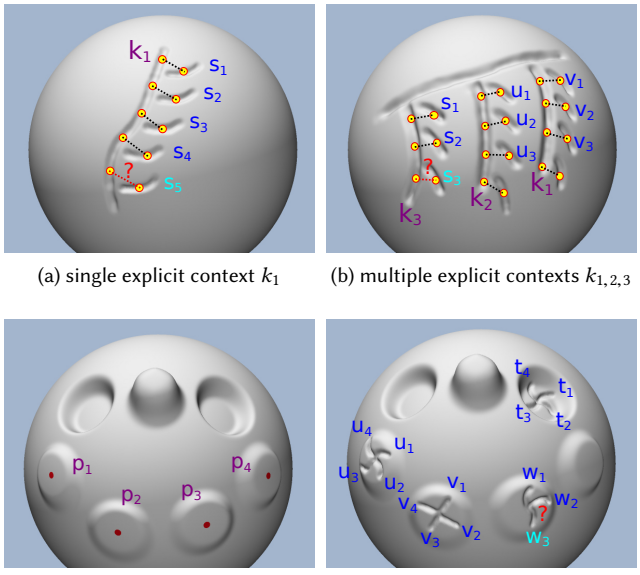
$$\Psi(\mathbf{b}_o) = \sum_{s_o \in \mathbf{b}_o} |\mathbf{u}(s_o) - \bar{\mathbf{u}}(s_o)|^2 \tag{15}$$

The last term, Θ, allows users to configure various parameter settings for various effects such as dots as in [Pixologic 2015]. This can be achieved by adding constraints $\mathbf{c}$ for various sample attributes:

$$\Theta(\mathbf{b}_o) = \sum_{s_o \in \mathbf{b}_o} |\mathbf{u}(s_o) - \mathbf{c}(s_o)|^2 \tag{16}$$

Together, the search and assignment steps decide the next stroke $\mathbf{b}^*_o$ via minimizing Equation (11) with expansions in Equations (7), (15) and (16).

5.5 Context Analysis



(a) single explicit context k_1 (b) multiple explicit contexts $k_{1,2,3}$

(c) implicit context correspondence (d) implicit contexts

Fig. 15. *Context analysis examples.* Our system considers two kinds of contexts: explicit (top) and implicit (bottom) based on workflow and geometry. In (a), the strokes s_1 to s_4 are aligned to a common long stroke k_1, which serves as a shared explicit context to help predict s_5. Intuitively, s_5 relative to k_1 is the average of s_1 to s_4 relative to k_1. In (b) there are multiple long context strokes from which s_3 can be predicted from u_3 and v_3. (c) shows implicit geometry correspondences, such as the freeform extrusions $p_{1,2,3,4}$ with similar geometry. This deduced implicit contexts are then used for prediction in (d), where w_3 may come from u_3 (similar pattern and context) but not v_3 (different pattern) or t_3 (different context).

Sculpting is a process of strokes accumulation. Strokes with different types, lengths, directions can overlap and interact one another via explicit stroke-level or implicit geometry-level correlations. This is one major difference between 2D sketching and 3D sculpting. In

our system, we consider these two kinds of contexts to help prediction, as illustrated in Figure 15. For each predicted stroke $\mathbf{b}_o$, we perform contextual analysis on the set of strokes that are similar to $\mathbf{b}_o$. The strokes from these contexts provide the refined prediction $\overline{\mathbf{u}}(\mathbf{b}_o)$ for Equation (15).

The explicit/implicit contexts are orthogonal to surface/freeform strokes, and thus all four combinations of explicit/implicit × surface/freeform are possible. The explicit contexts are long brush strokes while the implicit contexts are large geometry changes relative to nearby detailed strokes.

Explicit context. As illustrated in Figures 15a and 15b, a stroke $\mathbf{b}_o$ is often correlated with and constrained by the nearby set of long strokes $\{\mathbf{b}'_o\}$, which we term the explicit context of $\mathbf{b}_o$. Via Equation (6), such context can be represented as $\{\hat{\mathbf{u}}(\mathbf{b}'_o, \mathbf{b}_o)\}$.

Within the temporal-spatial neighborhood of $\mathbf{b}_o$, a stroke will be considered as an explicit contextual stroke of $\mathbf{b}_o$ only when it is at least three times longer than $\mathbf{b}_o$. If no such long strokes are detected, we skip this explicit context analysis step.

Intuitively, to predict how $\mathbf{b}_o$ should be constrained by $\{\mathbf{b}'_o\}$, we analyze how its matching strokes $\{\mathbf{b}_i\}$ are constrained by their context strokes. For each context stroke $\mathbf{b}'_o \in \{\mathbf{b}'_o\}$ encoded by $\hat{\mathbf{u}}(\mathbf{b}'_o, \mathbf{b}_o)$, we perform statistics analysis for $\{\hat{\mathbf{u}}(\mathbf{b}'_i, \mathbf{b}_i)\}$, where each $\hat{\mathbf{u}}(\mathbf{b}'_i, \mathbf{b}_i)$ is a matching pair of $\hat{\mathbf{u}}(\mathbf{b}'_o, \mathbf{b}_o)$. Note that $\mathbf{b}'_i$ and $\mathbf{b}'_o$ can be the same or different strokes as shown in Figures 15a and 15b.

We analyze the statistics for $\{\hat{\mathbf{u}}(\mathbf{b}'_i, \mathbf{b}_i)\}$ based on the constituent matching samples pairs $\{\hat{\mathbf{u}}(s'_i, s_i)\}$ (Equation (6)). Specifically, for each sample pair $\{\hat{\mathbf{u}}(s'_o, s_o)\} \in \{\hat{\mathbf{u}}(\mathbf{b}'_o, \mathbf{b}_o)\}$, we extract its matching sample-pairs $\{\hat{\mathbf{u}}(s'_i, s_i)\}$ from $\{\hat{\mathbf{u}}(\mathbf{b}'_i, \mathbf{b}_i)\}$, as shown in the dash lines in Figures 15a and 15b. The sample differences are computed according to Equation (4). The average and standard deviation of $\{\hat{\mathbf{u}}(s'_i, s_i)\}$, $\{\overline{\mathbf{u}}(s'_i, s_i)\}$ and $\{\sigma(s'_i, s_i)\}$, serve as the prediction and stability for the explicit context constraints in Equation (17).

Implicit context. In sculpting, users often form larger base regions via freeform strokes followed by detailed strokes [Denning et al. 2015]. Since similar base regions are often decorated with similar detailed strokes, we consider the geometry base of each detail stroke $\mathbf{b}_o$ as its implicit context.

Within the temporal-spatial neighborhood of $\mathbf{b}_o$, the intersected geometry base of $\mathbf{b}_o$ will be considered as implicit context only when the freeform stroke radius leading to that base geometry transformation is at least three times larger than the radius of $\mathbf{b}_o$. If no underlying large geometry changes are detected for $\mathbf{b}_o$, we skip this implicit context analysis step.

Once repeated large geometry changes are detected (often caused by repeated large freeform strokes), we extract the central sample from each affected region surface, as shown in Figure 15c. We use the central samples of the corresponding base regions as the origin of local parameterization [Schmidt et al. 2006]. For a $\mathbf{b}_o$, we find the matching strokes $\{\mathbf{b}_i\}$ with the corresponding base regions, and use their constituent samples to compute the average and standard deviation, $\{\overline{\mathbf{u}}(s_i)\}$ and $\{\sigma(s_i)\}$, which serve as the predictions and stability for the implicit constraint predictions in Equation (17). One example is shown in Figure 15d.

In theory [Schmidt et al. 2006] is an isotropic parameterization and thus can cause larger distortion than the anisotropic version in [Schmidt 2013]. In practice, we have found it sufficient as the distortion remains similar for repeated strokes even for elongated deformation such as Figure 5d.

There are existing geometry-based methods (such as [Maximo et al. 2011; Zelinka and Garland 2004]) to detect similar regions, but we analyze workflows instead of geometry for better efficiency and adaptivity for dynamic model changes. For example, the method in [Maximo et al. 2011] is more suitable for static/finished mesh analysis, but not for real time sculpting with dramatic mesh modifications due to high computation cost.

Prediction. We obtain the contextual prediction $\overline{\mathbf{u}}(\mathbf{b}_o)$ by optimizing Equation (17), which combines the implicit and explicit constraints based on the std-weighted mean differences.

$$\begin{aligned}\mathbf{E}(\mathbf{u}(\mathbf{b}_o)) = \sum_{\mathbf{b}'_o \in \{\mathbf{b}'_o\}} \sum_{s_o \in \mathbf{b}_o} \sum_{s'_o \in \mathbf{b}'_o} \exp\left(\frac{-\sigma(s_i)}{\epsilon}\right) |\mathbf{u}(s_o) - \overline{\mathbf{u}}(s_i)|^2 \\ + \exp\left(\frac{-\sigma(s_i, s'_i)}{\epsilon}\right) \left|\mathbf{u}(s_o) - \mathbf{u}(s'_o) - \overline{\mathbf{u}}(s_i, s'_i)\right|^2\end{aligned} \tag{17}$$

$$\overline{\mathbf{u}}(\mathbf{b}_o) = \arg\min \mathbf{E}(\mathbf{u}(\mathbf{b}_o)) \tag{18}$$

5.6 Deployment

Based on the common framework in Section 5.4, we now describe how to support various modes and options in our system.

Hint. The predicted and accepted strokes are rendered in light transparent yellow and blue colors for visualization.

With the context analysis in Section 5.5, we can propagate edits concurrently from one region to other similar regions. The propagations are rendered in the same way as ordinary hints.

Normalization. For workflow clone, we normalize the stroke parameterization to support source and target regions specified with different stroke lengths. Specifically, a sample s in a surface stroke will be normalized to be within $t_s \in [-r/T, 1 + r/T], d_s \in [-1, 1]$ via:

$$t_s \leftarrow \frac{t_s}{T}, \quad d_s \leftarrow \frac{d_s}{r}, \tag{19}$$

where T is the stroke arc length and r is the parameterization width range, as illustrated in Figure 13.

We also normalize the sample-id to fall within $[0, 1]$, where 0 and 1 represent the starting and ending positions of stroke $\mathbf{b}$.

Workflow lock. The footprint of a stroke may unintentionally influence nearby geometry. As exemplified in Figure 16 under symmetry mode, the current stroke can influence everything inside the yellow sphere. Thus, the geometry created by the prior stroke $\mathbf{b}_{A2}$ can be undesirably deformed by the next stroke stroke $\mathbf{b}_{B2}$. This is another key difference from [Xing et al. 2014, 2015] where the synthesized sketching strokes are the final outputs.

While locking in common sculpting tools can address this issue, it requires manual user intervention. Automatically deducing which parts of the model to lock based on geometry alone can be challenging, as spatial information may not convey user intention. With workflows, our method can directly associate all brush strokes with

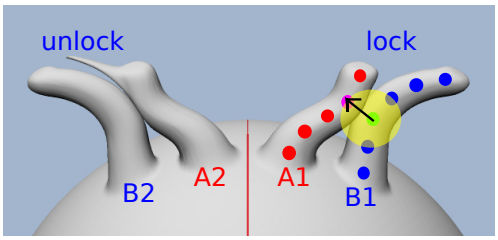
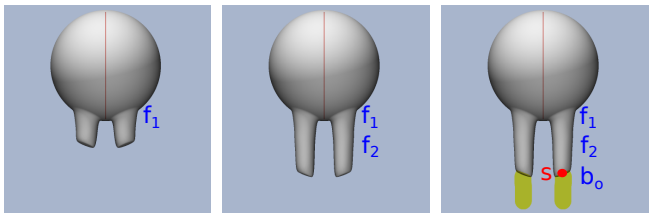


Fig. 16. *Workflow lock based on spatial-temporal neighborhood.* The strokes are placed in the order $\mathbf{b}_A \rightarrow \mathbf{b}_B$, and in a symmetry mode for illustration. The left side shows general sculpting effect without workflow lock, the right side is aided with workflow lock. Any samples of $\mathbf{b}_{A1}$ within the spatial-temporal neighborhood of $\mathbf{b}_{B1}$ will be automatically locked, as exemplified in the yellow region of the green sample in $\mathbf{b}_{B1}$.

the model geometry, and decide what to lock based on workflow similarity as described in Section 5.2. Based on our experiments, we adopt a simple strategy to lock all past workflows with a spatial-temporal neighborhood of the current brush stroke, as shown in Figure 16. This strategy works well when users sculpt in a spatially-temporally coherent fashion, as they often do.

Camera control. As described in [Chen et al. 2014; Santoni et al. 2016], user brush strokes tend to correlate with camera movements and thus can facilitate viewpoint selection. Therefore, different from [Bae et al. 2008, 2009] which consider biomechanics constraints on sketching curves and view-planes, our camera movement prediction considers past strokes only. Our system stores all camera states, including positions and orientations, as part of the sculpting workflows. Thus, our method is able to predict camera movements in addition to sculpting strokes as described in Section 5.4. In our experiments we have found that excessive camera automation can be disorienting to users. We thus expose only the basic mode of piloting the camera viewpoint along with the predicted next brush strokes.



(a) first stroke (b) second stroke (c) hint

Fig. 17. *Long freeform strokes.* When repetitions are detected for long freeform features as shown in (a) and (b), suggestions within such long feature are provided in (c). The starting sample of $\mathbf{b}_o$ rendered in red is connected with the ending sample of the previous stroke in (b).

Long freeform stroke. It is common to sculpt long freeform strokes for organic features such as limbs Figure 4i. However, instead of at one go, users often perform multiple connected or overlapped strokes for iterative refinement [Santoni et al. 2016]. When repetitions are detected for such long freeform features, we constrain the starting sample position of $\mathbf{b}_o$ to the ending sample position of just completed stroke to ensure their connection (see Figure 17).

Neighborhood and search window. We set r dynamically to be 4× the stroke radius. The spatial-temporal neighborhood of a brush stroke includes its two previous temporal strokes and nearby spatial strokes overlapping its parameterization region (Figure 13). For the search step in Section 5.4.2, we search within a local temporal-spatial window of 20 previous temporal strokes, and the same spatial neighborhood window as above.

Neighborhood acceleration. To improve quality and speed, we accelerate the neighborhood matching in Equation (7) by a two-tiered sampling process for the brush strokes. Specifically, we first place three samples uniformly over each stroke to select the most similar candidate strokes, and continue with all samples to select the best matches from the candidates.

3D acceleration. Interaction speed for sculpting is a major concern [Calabrese et al. 2016] compared to 2D applications and low-polygonal 3D hard-surface modeling [Salvati et al. 2015]. With more strokes, the original base domain in sculpting will undergo significant mesh changes with increasing mesh elements via adaptive subdivisions. The created models as shown in Figures 1 and 19 contain around 360k faces faces on average, comparable with [Calabrese et al. 2016]. To handle such high-resolution models unique in sculpting, we adopt the octree structure to support instant brush sphere and ray lookups, such structure ensures fast mesh updates, as mesh changes only needed to be made locally.

Hints number. For overlapped strokes, we set the predicted number of strokes to be 1, and users can customize the number of continuous overlapped strokes to be accepted as in Figure 4. For non-overlapped strokes, users can also customize the number of predictions;based on our experiments, setting the number of iterations to be within $[3, 6]$ works well in practice. By default, we set the iteration number to be 3 for freeform predictions and 5 for surface predictions; users can accept the hints via selection brush as in Figure 3.

To improve the hint visualization quality, instead of directly displaying the exact iteration number of hints, we optimize the hints visualization with extra filtering: (1) when context as in Section 5.5 is detected for last sculpted stroke $\mathbf{b}'_o$, we filter predicted strokes that are not constrained by contextual strokes; (2) when symmetry mode is enabled, we further filter predictions that cross the symmetry border.

Weights. For Equation (4), we set the position weighting $w_\mathbf{p}$ to be 1. We set $w_\mathbf{a}$ to be 1 if there is no Θ term in Equation (11), otherwise we set it to be 0.1 and 0.9 for the neighborhood and Θ terms. The $w_\mathbf{t}$ includes global time stamp $w_{\mathbf{t}_1}$ and sample-id $w_{\mathbf{t}_2}$. We set $w_{\mathbf{t}_2}$ to be 1, and $w_{\mathbf{t}_1}$ to be 100 for temporal neighborhood matching to enforce the same sculpting order, and 0 for spatial neighborhood matching. For Equation (17), we set ϵ to 10.

6 USER STUDY

We have conducted a preliminary user study to evaluate the usability and efficacy of our assisted sculpting system.

Setup. All tasks were conducted on a 13-in laptop with a Wacom tablet. The study contains three sessions: tutorial, open creation, and interview. The entire study took about one hour per participant.

Participants. Our participants include 2 experienced modelers and 6 novice users with different levels of modeling experiences, to help us gain insights during design iterations and gather feedbacks during evaluation.

Tutorial session. The tutorial session is designed to help participants familiarize with the sculpting interface and various functions of our system. The tasks consist of manipulating various sculpting parameters such as type, radius size, pressure, and direction; performing initial base shape formation via large-scale operations, adding details via different surface tools; and moving the user viewpoints via camera control. To accelerate the learning process, we play a recorded UI introduction video to participants, followed with more detailed introduction on how to use different functions via live demonstrations by the authors.

Target session. We have designed a target session to quantify the usability of our assisted system compared to traditional sculpting tools. Our collaborating artists created the initial input and reference output, and the participants started from the same initial input to reach the reference output. Each participant performed this task both with and without assisted functions. Details of this study and the results are reported in an earlier version of this work [Peng et al. 2017]. We have removed this study for this version and emphasized more on open-creation below, as our study participants commented that such closed-ended tasks might limit their creativity while open-creation tasks better simulate real-world sculpting and help fuller exploration of our system.

Open creation. The goal of this session is to observe participant behaviors and identity merits and issues of our systems. Since digital sculpting is a flexible art form to express various creativities, we do not enforce pre-defined requirements for this session. Participants are encouraged to explore various functions and perform open-ended sculpting using our system, with personalized and unconstrained degree of repetitions.

Interview. In the final session, we collect feedbacks from each participant on different aspects on our assisted interactive system, including individual functions, overall satisfaction, and open-ended feedbacks.

7 RESULTS

Subjective satisfaction. Figure 18 summarizes the subjective feedbacks about the individual features, easiness to use, and overall satisfaction. Overall, our prototype system has received positive opinions on satisfaction and easiness to use. Among various modes, hint, especially with context analysis, receives the highest rating; clone, lock, and propagation come next; camera control is considered least useful. More detailed study results are recorded in Appendix A.

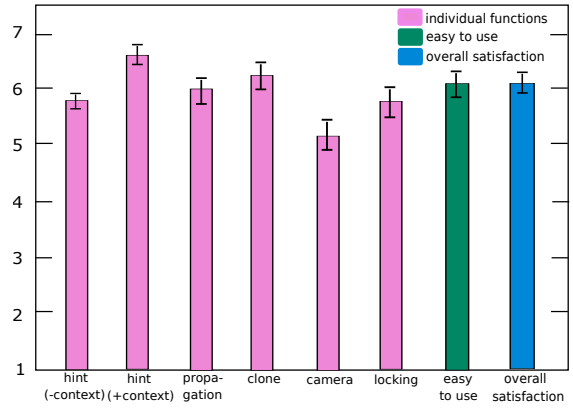


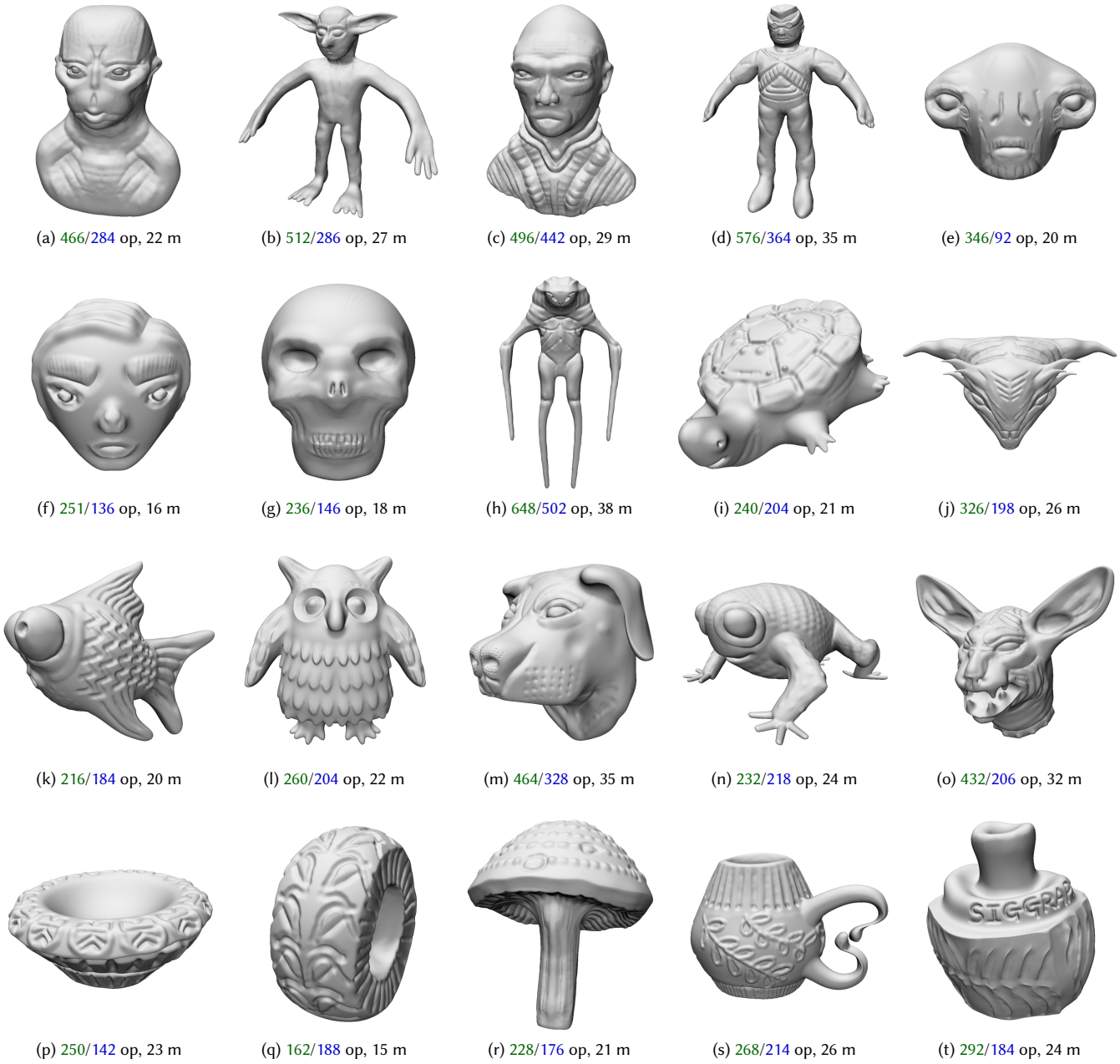
Fig. 18. *Subjective feedbacks from 8 participants in a 7-Likert scale on individual functions, easiness to use, and overall satisfaction, expressed as mean ± standard error.*

Sample outputs. Figure 19 show sample outputs from our user study participants. Our assisted system can help them produce 3D models with good quality and diverse styles. The ratio of autocomplete to total number of edits depends on the detectable amounts of repetitions, ranging from 21% for Figure 19e to 54% for Figure 19q. Please refer to our supplementary videos for recorded actions.

Stroke patterns. Our method also supports other stroke patterns such as self-intersections and aggregates, as shown in Figure 20. Even though in sculpting practice, such stroke patterns are less frequently used, we support such patterns inherently to help users sculpt some special patterns if needed.

Hierarchical editing. Many existing methods such as [Maximo et al. 2011; Zelinka and Garland 2004] can detect similar geometry. However, such type of approaches might not have sufficient speed for interactive suggestions and capability for dynamic changes. Our workflow-based method deals with key correspondences detection and geometry processing at the same time. An example is shown in Figure 21.

Workflow clone versus geometry clone. As sculpting is an accumulative process, it is common for users to perform additive cloning over existing shapes of the target regions. Capability-wise, our workflow clone keeps track of dynamic geometry changes, and supports additive cloning, such as detail additions or overlapping strokes. These are difficult if not impossible for pure geometry-based methods such as [Takayama et al. 2011], which assumes that the initial clone target geometry should be completely replaced by the copy source. Speed-wise, our interactions are in real-time while many pure geometry methods are unsuitable for instant interactions, e.g. the cost of re-stamping new source location in [Takayama et al. 2011] makes it difficult for continuous interactive sculpting. See Figure 22 for examples.



(a) 466/284 op, 22 m (b) 512/286 op, 27 m (c) 496/442 op, 29 m (d) 576/364 op, 35 m (e) 346/92 op, 20 m

(f) 251/136 op, 16 m (g) 236/146 op, 18 m (h) 648/502 op, 38 m (i) 240/204 op, 21 m (j) 326/198 op, 26 m

(k) 216/184 op, 20 m (l) 260/204 op, 22 m (m) 464/328 op, 35 m (n) 232/218 op, 24 m (o) 432/206 op, 32 m

(p) 250/142 op, 23 m (q) 162/188 op, 15 m (r) 228/176 op, 21 m (s) 268/214 op, 26 m (t) 292/184 op, 24 m

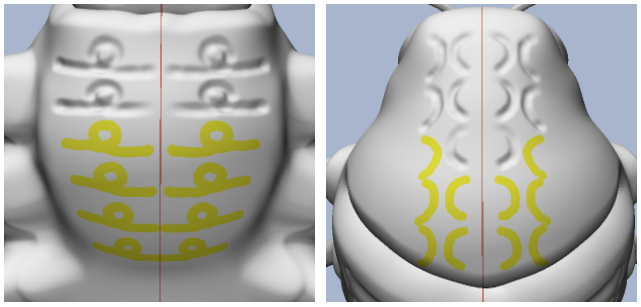
Fig. 19. *Sample outputs from our participants, all starting from a sphere.* Denoted with each output are the following statistics: number of manual sculpting strokes, number of autocomplete strokes, and total authoring time in minutes. Please refer to the supplementary videos for recorded modeling sessions.

8 LIMITATIONS AND FUTURE WORK

We present an autocomplete 3D sculpting system that can reduce input labor and enhance output quality, especially for novices, and demonstrate that the addition of dynamic workflows can effectively augment static geometry for interactive model creation. The autocomplete prototype targets repetitive operations, and falls back to traditional sculpting for non-repetitive operations.

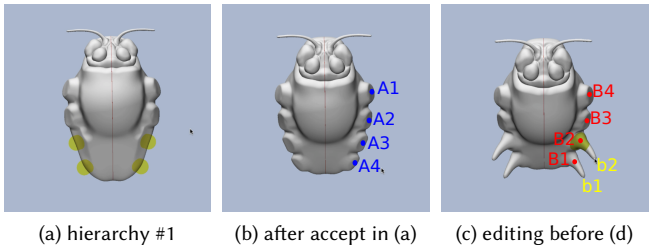
We propose an automatic camera control mode following workflow suggestions. This is a very basic mode and yet conservative enough to avoid user disorientation. Additional automatic camera controls are possible from the data and method perspectives, but warrant further user studies.

Our system can make aggressive predictions and let users decide whether or not to accept borderline hints, as exemplified in Figure 23.



(a) self-intersections (b) aggregates

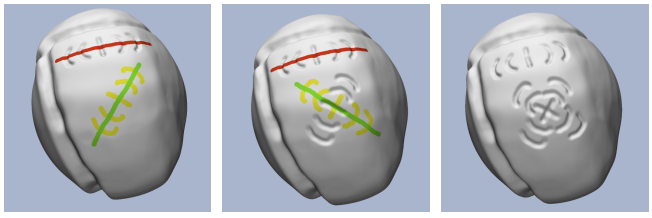
Fig. 20. *Stroke pattern variety.* Our method can also support stroke patterns such as self-intersections in (a) and aggregates in (b).



(a) hierarchy #1 (b) after accept in (a) (c) editing before (d)

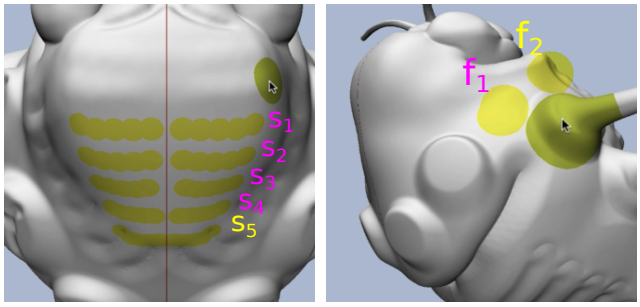
(d) hierarchy #2 (e) hierarchy #3 (f) hierarchy #4

Fig. 21. *Workflow-assisted hierarchical editing example.* Static geometry analysis can detect similarity in completed shapes such as points $A_{1,2,3,4}$ in (b), but not intermediate dynamically changes as (c). In contrast, our method can detect the similarity between b_1 and b_2 relative to B_1 and B_2 to help provide predictions over B_3 and B_4 in (d).



(a) first target (b) overlapped target (c) result

Fig. 22. *Clone re-application.* Via workflow clone, users can copy the same source in red over overlapping target regions for additive cloning.



(a) surface hints (b) freeform hints

Fig. 23. *Limitation cases.* Our predictions can be overly aggressive around areas with more geometry differences, such as the surface hints in (a) and the freeform hints in (b). Specificalaly, s_5 and f_2 are aggressively suggested over regions quite different from others ($s_{1,2,3,4}$ and f_1) to let users decide whether or not to accept.

We plan to explore other geometry and topology features for more accurate correlation between workflow and geometry. To further improve prediction accuracy, instead of manually crafted algorithms, we plan to investigate machine learning approaches that analyze user feedbacks (whether they accept, partially accept, or ignore the suggestions) for continuous training our prediction models.

Within the scope of this project we have focused on a single user within a single modeling session. The general ideas and specific methods can be extended to multiple users for crowdsourcing and tutorials as well as multiple sessions for individual stylization. A potential future work is to integrate multi-scale hierarchy structure in collaborative sculpting [Calabrese et al. 2016] for autocomplete across multiple users.

Our current system supports only single-layer surface mesh sculpting. A future direction is to explore alternative representations such as multi-layers for stratified [Calabrese et al. 2017] and voxels for volumetric [Wang and Kaufman 1995] materials, whose additional complexity can benefit even more from autocompletes than single-layer surface sculpting.

We focus on 3D sculpting as it is a very popular and flexible form of model creation. Digital sculpting is mainly for creating organic shapes [Calabrese et al. 2016; Chen et al. 2014; Denning et al. 2015]. We have applied our system for less organic, man-made objects as shown in Figures 19p to 19t, but the precision is less than hard-surface modeling tools such as AutoCAD. A future work is to extend autocomplete for other forms of 3D modeling, such as mechanical objects that require higher precisions in geometry, and VR brushing that operates more in 3D spaces than object surfaces.

ACKNOWLEDGMENTS

We would like to thank our user study participants for their time and feedbacks, and the anonymous reviewers for their valuable suggestions. This work has been partially supported by Hong Kong RGC general research fund 17202415.

REFERENCES

Autodesk. 2014. Mudbox. (2014). http://www.autodesk.com/products/mudbox/.

Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: As-natural-as-possible Sketching System for Creating 3D Curve Models. In *UIST '08*. 151–160.

Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2009. EverybodyLovesSketch: 3D Sketching for a Broader Audience. In *UIST '09*. 59–68.

Blender Foundation. 2016. Blender. (2016).

Martin Bokeloh, Michael Wand, Vladlen Koltun, and Hans-Peter Seidel. 2011. Pattern-aware Shape Deformation Using Sliding Dockers. *ACM Trans. Graph.* 30, 6, Article 123 (2011), 10 pages.

Claudio Calabrese, Marco Fratarcangeli, and Fabio Pellacini. 2017. sLayer: a System for Multi-Layered Material Sculpting.

Claudio Calabrese, Gabriele Salvati, Marco Tarini, and Fabio Pellacini. 2016. cSculpt: A System for Collaborative Sculpting. *ACM Trans. Graph.* 35, 4, Article 91 (2016), 8 pages.

Siddhartha Chaudhuri and Vladlen Koltun. 2010. Data-driven Suggestions for Creativity Support in 3D Modeling. *ACM Trans. Graph.* 29, 6, Article 183 (2010), 10 pages.

Hsiang-Ting Chen, Tovi Grossman, Li-Yi Wei, Ryan M. Schmidt, Björn Hartmann, George Fitzmaurice, and Maneesh Agrawala. 2014. History Assisted View Authoring for 3D Models. In *CHI '14*. 2027–2036.

Hsiang-Ting Chen, Li-Yi Wei, and Chun-Fa Chang. 2011. Nonlinear Revision Control for Images. *ACM Trans. Graph.* 30, 4, Article 105 (2011), 10 pages.

Hsiang-Ting Chen, Li-Yi Wei, Björn Hartmann, and Maneesh Agrawala. 2016. Data-driven Adaptive History for Image Editing. In *I3D '16*. 103–111.

Daniel Cohen-Or and Hao Zhang. 2016. From Inspired Modeling to Creative Modeling. *Vis. Comput.* 32, 1 (2016), 7–14.

Fernando De Goes and Doug L. James. 2017. Regularized Kelvinlets: Sculpting Brushes Based on Fundamental Solutions of Elasticity. *ACM Trans. Graph.* 36, 4, Article 40 (2017), 11 pages.

Jonathan D. Denning, William B. Kerr, and Fabio Pellacini. 2011. MeshFlow: Interactive Visualization of Mesh Construction Sequences. *ACM Trans. Graph.* 30, 4, Article 66 (2011), 8 pages.

Jonathan D. Denning and Fabio Pellacini. 2013. MeshGit: Diffing and Merging Meshes for Polygonal Modeling. *ACM Trans. Graph.* 32, 4, Article 35 (2013), 10 pages.

Jonathan D. Denning, Valentina Tibaldo, and Fabio Pellacini. 2015. 3DFlow: Continuous Summarization of Mesh Editing Workflows. *ACM Trans. Graph.* 34, 4, Article 140 (2015), 9 pages.

David S. Ebert, F. Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steven Worley. 2002. *Texturing and Modeling: A Procedural Approach* (3rd ed.). Morgan Kaufmann Publishers Inc.

Arnaud Emilien, Ulysse Vimont, Marie-Paule Cani, Pierre Poulin, and Bedrich Benes. 2015. WorldBrush: Interactive Example-based Synthesis of Procedural Virtual Worlds. *ACM Trans. Graph.* 34, 4, Article 106 (2015), 11 pages.

Lubin Fan, Ruimin Wang, Linlin Xu, Jiansong Deng, and Ligang Liu. 2013. Modeling by Drawing with Shadow Guidance. *Computer Graphics Forum (Proc. Pacific Graphics)* 23, 7 (2013), 157–166.

Chi-Wing Fu, Jiazhi Xia, and Ying He. 2010. LayerPaint: A Multi-layer Interactive 3D Painting Interface. In *CHI '10*. 811–820.

Hongbo Fu, Shizhe Zhou, Ligang Liu, and Niloy J. Mitra. 2011. Animated Construction of Line Drawings. *ACM Trans. Graph.* 30, 6, Article 133 (2011), 10 pages.

Thomas Funkhouser, Michael Kazhdan, Philip Shilane, Patrick Min, William Kiefer, Ayellet Tal, Szymon Rusinkiewicz, and David Dobkin. 2004. Modeling by Example. *ACM Trans. Graph.* 23, 3 (2004), 652–663.

Raul Fernandez Hernandez. 2011. Dynamic Subdivision Sculpting. (2011).

Ruizhen Hu, Oliver van Kaick, Bojian Wu, Hui Huang, Ariel Shamir, and Hao Zhang. 2016. Learning How Objects Function via Co-analysis of Interactions. *ACM Trans. Graph.* 35, 4, Article 47 (2016), 13 pages.

Shi-Min Hu, Kun Xu, Li-Qian Ma, Bin Liu, Bi-Ye Jiang, and Jue Wang. 2013. Inverse Image Editing: Recovering a Semantic Editing History from a Before-and-after Image Pair. *ACM Trans. Graph.* 32, 6, Article 194 (2013), 11 pages.

Takeo Igarashi and John F. Hughes. 2001. A Suggestive Interface for 3D Drawing. In *UIST '01*. 173–181.

Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 1999. Teddy: A Sketching Interface for 3D Freeform Design. In *SIGGRAPH '99*. 409–416.

Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, Shengdong Zhao, and George Fitzmaurice. 2014. Draco: Bringing Life to Illustrations with Kinetic Textures. In *CHI '14*. 351–360.

Rubaiat Habib Kazi, Takeo Igarashi, Shengdong Zhao, and Richard Davis. 2012. Vignette: Interactive Texture Design and Manipulation with Freeform Gestures for Pen-and-ink Illustration. In *CHI '12*. 1727–1736.

Matt Kloskowski. 2010. Cloning With a Preview in Photoshop CS4. (2010). http://www.photoshop.com/tutorials/4305.

David Kurlander and Eric A. Bier. 1988. Graphical Search and Replace. In *SIGGRAPH '88*. ACM, 113–120.

David Kurlander and Steven Feiner. 1992a. A History-based Macro by Example System. In *UIST '92*. ACM, 99–106.

David Kurlander and Steven Feiner. 1992b. Interactive Constraint-based Search and Replace. In *CHI '92*. ACM, 609–618.

Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra. 2005. Texture Optimization for Example-based Synthesis. *ACM Trans. Graph.* 24, 3 (2005), 795–802.

Jerry Liu, Fisher Yu, and Thomas A. Funkhouser. 2017. Interactive 3D Modeling with a Generative Adversarial Network. *CoRR* abs/1706.05170 (2017).

Zhaoliang Lun, Evangelos Kalogerakis, Rui Wang, and Alla Sheffer. 2016. Functionality Preserving Shape Style Transfer. *ACM Trans. Graph.* 35, 6, Article 209 (2016), 14 pages.

Chongyang Ma, Li-Yi Wei, Sylvain Lefebvre, and Xin Tong. 2013. Dynamic Element Textures. *ACM Trans. Graph.* 32, 4, Article 90 (2013), 10 pages.

André Maximo, Robert Patro, Amitabh Varshney, and R Farias. 2011. A robust and rotationally invariant local surface descriptor with applications to non-local mesh processing. *Graphical Models* 73, 5 (2011), 231–242.

Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. 2006. Partial and Approximate Symmetry Detection for 3D Geometry. *ACM Trans. Graph.* 25, 3 (2006), 560–568.

Niloy J. Mitra, Michael Wand, Hao Zhang, Daniel Cohen-Or, and Martin Bokeloh. 2013. Structure-Aware Shape Processing. In *Eurographics 2013 - State of the Art Reports*.

Mathieu Nancel and Andy Cockburn. 2014. Causality: A Conceptual Model of Interaction History. In *CHI '14*. 1777–1786.

Gen Nishida, Ignacio Garcia-Dorado, Daniel G. Aliaga, Bedrich Benes, and Adrien Bousseau. 2016. Interactive Sketching of Urban Procedural Models. *ACM Trans. Graph.* 35, 4, Article 130 (2016), 11 pages.

Michaël Ortega and Thomas Vincent. 2014. Direct Drawing on 3D Shapes with Automated Camera Control. In *CHI '14*. 2047–2050.

Mengqi Peng, Jun Xing, and Li-Yi Wei. 2017. Autocomplete 3D Sculpting. *CoRR* abs/1703.10405 (2017). arXiv:cs.GR/1703.10405

Patrick Pérez, Michel Gangnet, and Andrew Blake. 2003. Poisson Image Editing. *ACM Trans. Graph.* 22, 3 (2003), 313–318.

Pixologic. 2011. Sculptris. (2011).

Pixologic. 2015. ZBrush. (2015).

Gabriele Salvati, Christian Santoni, Valentina Tibaldo, and Fabio Pellacini. 2015. MeshHisto: Collaborative Modeling by Sharing and Retargeting Editing Histories. *ACM Trans. Graph.* 34, 6, Article 205 (2015), 10 pages.

Christian Santoni, Claudio Calabrese, Francesco Di Renzo, and Fabio Pellacini. 2016. SculptStat: Statistical Analysis of Digital Sculpting Workflows. *CoRR* abs/1601.07765 (2016).

Ryan Schmidt. 2013. Stroke Parameterization. *Computer Graphics Forum* 32, 2pt2 (2013), 255–263.

Ryan Schmidt, Cindy Grimm, and Brian Wyvill. 2006. Interactive Decal Compositing with Discrete Exponential Maps. *ACM Trans. Graph.* 25, 3 (2006), 605–613.

Ryan Schmidt and Karan Singh. 2010. Meshmixer: An Interface for Rapid Mesh Composition. In *SIGGRAPH '10 Talks*. Article 6, 1 pages.

Adriana Schulz, Ariel Shamir, David I. W. Levin, Pitchaya Sitthi-amorn, and Wojciech Matusik. 2014. Design and Fabrication by Example. *ACM Trans. Graph.* 33, 4, Article 62 (2014), 11 pages.

Ben Shneiderman. 2007. Creativity Support Tools: Accelerating Discovery and Innovation. *Commun. ACM* 50, 12 (2007), 20–32.

Qian Sun, Long Zhang, Minqi Zhang, Xiang Ying, Shi-Qing Xin, Jiazhi Xia, and Ying He. 2013. Texture Brush: An Interactive Surface Texturing Interface. In *I3D '13*. 153–160.

Ryo Suzuki, Tom Yeh, Koji Yatani, and Mark D. Gross. 2017. Autocomplete Textures for 3D Printing. *ArXiv e-prints* (2017). arXiv:cs.HC/1703.05700

Kenshi Takayama, Ryan Schmidt, Karan Singh, Takeo Igarashi, Tamy Boubekeur, and Olga Sorkine. 2011. Geobrush: Interactive mesh geometry cloning. In *Computer Graphics Forum*, Vol. 30. 613–622.

Jianchao Tan, Marek Dvorožňák, Daniel Sýkora, and Yotam Gingold. 2015. Decomposing Time-lapse Paintings into Layers. *ACM Trans. Graph.* 34, 4, Article 61 (2015), 10 pages.

Steve Tsang, Ravin Balakrishnan, Karan Singh, and Abhishek Ranjan. 2004. A Suggestive Interface for Image Guided 3D Sketching. In *CHI '04*. 591–598.

Sidney W. Wang and Arie E. Kaufman. 1995. Volume Sculpting. In *I3D '95*. 151–ff.

Jun Xing, Hsiang-Ting Chen, and Li-Yi Wei. 2014. Autocomplete Painting Repetitions. *ACM Trans. Graph.* 33, 6, Article 172 (2014), 11 pages.

Jun Xing, Li-Yi Wei, Takaaki Shiratori, and Koji Yatani. 2015. Autocomplete Hand-drawn Animations. *ACM Trans. Graph.* 34, 6, Article 169 (2015), 11 pages.

Ya-Ting Yue, Xiaolong Zhang, Yongliang Yang, Gang Ren, Yi-King Choi, and Wenping Wang. 2017. WireDraw: 3D Wire Sculpturing Guided with Mixed Reality. In *CHI '17*. 3693–3704.

Steve Zelinka and Michael Garland. 2004. Similarity-based Surface Modelling Using Geodesic Fans. In *SGP '04*. 204–213.

A DETAILED USER FEEDBACK

Table 2. *Subjective feedback from 8 participants in a 7-Likert scale on whether it is easy to use this assisted system, and overall satisfaction.*

user	easy to use	overall satisfaction
P_1	6	6
P_2	6.5	6
P_3	5.5	5.5
P_4	7	6.5
P_5	6	6.5
P_6	5.5	5.5
P_7	6	6.5
P_8	6	6

Table 3. *Subjective feedback from 8 participants in a 7-Likert scale for various modes of our system.*

user	hint		propa-gation	clone	camera	lock
	w/o context	w/ context				
P_1	5.5	6.5	5.5	6	5	6
P_2	6	7	6	6.5	5.5	5.5
P_3	6	6.5	5.5	5.5	6	4.5
P_4	6	7	6	6	5.5	6
P_5	5.5	6.5	6	6.5	5	6
P_6	5.5	6	6	5.5	4	6
P_7	6	7	7	6.5	5.5	6.5
P_8	6	6.5	6.5	7	5	6

Below are more detailed participant feedbacks for our user study.

Q: Do you think the assisted system easy to use? And what is your overall scoring for this system?

See Table 2.

Q: What is your score for each function we provided in the system?

See Table 3.

Q: Do you have prior modeling/sculpting experiences? What's your comments and suggestions for this autocomplete system, such as what you like and what you think can be improved? (If you have prior modeling/sculpting experience, feel free to compare the experiences via our system with your prior practice.)

Experienced users. P_1: The sculpting tools are similar to what are provided by Blender sculpting mode, but the autocomplete, propagation, cloning etc. are special here. Though being new, they are handy to grasp since I already know those effects such as grabbing and so on. The UI video tutorial is very helpful for me know what is the difference between yours with Blender. I like the propagation, copy and suggestion function the most. And they are easy to use/understand. In sum, the proposed system is helpful to reduce my efforts when creating models, and the yellow future ones can also stimulate more future edits. I think the camera function can be improved, it can move around with the hints now, but when there are no hints, can you also provide some other form of Autos? The sculpting brushes are similar to Blender, it will be nice if you provide a plug-in for Blender community.

P_2: (1) Comments and compare with my prior experience: The system you show is interesting, the sculpting is similar to other sculpting software, but it is also different. The auto future editing is an interesting idea, and useful for me to directly accept the guesses if I think they meet my goal. The clone and propagate functions can help complete patterns containing many strokes with a few edits. The lock tool can help protect previous structures. They are intuitive and useful. (2) Suggestions: Besides mesh sculpting, it will be nice if you also provide guesses for the final appearance editing. You can also provide more rendering modes to beautify the final models.

Novice users. P_3: I have limited 3D software experiences. But at the same time, your video is vivid to let me know how to use your system. What I like: I think the "hints", "cloning", and "propagating edits" are very helpful; they are shown to be transparent color so my own operations will not be influenced by your "hints". I think maybe you can provide some default profiles/models so I can combine them to create more complex models.

P_4: No, I do not know sculpting too much. The yellow autos are entertaining and are trying to predict what I want to do and have good quality, especially the hints and propagation functions you taught. In terms of this study, the tutorials you show help me to know what is sculpting and how to use your UI buttons and functions. Overall, I learned sculpting to create models and like this sculpting test.

P_5: Yes, I used Sculptris before during my courses. The brushes you gave me is similar to Sculptris, but the yellow suggestions are only available in your interface, and helps in particular for repetitive and continuous edits. You can provide other tools like Mask and more material default settings to make your own tool more powerful.

P_6: Yes, I used some modeling experiences before. The software you propose is easy to learn. The auto design concept is cool. I like the yellow autocompletes provided via context hints and propagations. It is helpful to generate shapes with many details.

P_7: I used Sketchfab to create architecture models before. The models are different from architecture models, the operations are softer and less mechanical. The auto functions you show are simple to operate and helpful, the predictions are just suggestions thus I can agree to accept or disagree to pass your suggestions. I like the Cloning the most, it is like copy and paste but I can adjust the target sizes and effects. Yellow suggestions with so-called "context" have higher quality. Camera is a heavy operation similar to Sketchfab, the flying with yellow strokes camera mode is intuitive but might have higher potential space for your future works to make it more advanced.

P_8: No, do not know modeling or sculpting much before. The hints/copy/propagation functions are easy to understand, I like the cloning the most, I can re-use my patterns for multiple times.