3DFlow: Continuous Summarization of Mesh Editing Workflows

Jonathan D. Denning* Valentina Tibaldo† Fabio Pellacini†
*Taylor University †Sapienza University of Rome

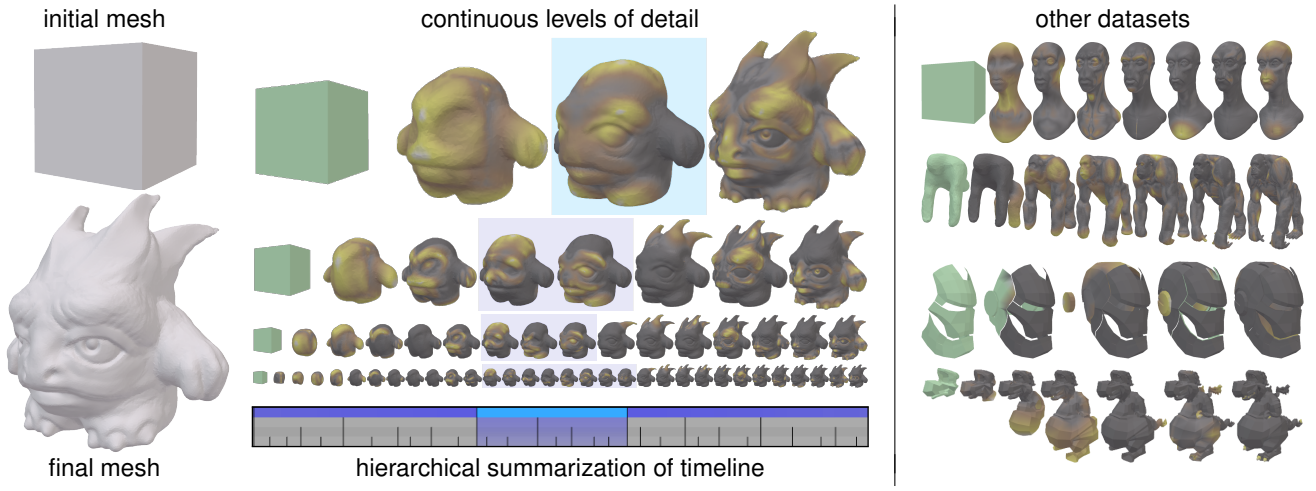


Figure 1: *Continuous levels of details automatically constructed from a 30 minute digital sculpting session of a professional artist. The artist sculpted the cube (top-left) into a monster (bottom-left) in 797 strokes using dynamic remeshing techniques. The center column shows the sequence summarized in 4, 8, 16, and 32 steps (top) and the corresponding timeline (bottom). The mesh is colored green to indicate created geometry and golden to indicate the strength of change from the previous mesh. Blue highlighting and vertical black lines indicate the hierarchical summarization. Four additional sequences are shown at different levels of detail on the right.*

Abstract

Mesh editing software is improving, allowing skilled artists to create detailed meshes efficiently. For a variety of reasons, artists are interested in sharing not just their final mesh but also their whole workflow, though the common media for sharing has limitations. In this paper, we present *3DFlow*, an algorithm that computes continuous summarizations of mesh editing workflows. *3DFlow* takes as input a sequence of meshes and outputs a visualization of the workflow summarized at any level of detail. The output is enhanced by highlighting edited regions and, if provided, overlaying visual annotations to indicated the artist's work, e.g. summarizing brush strokes in sculpting. We tested *3DFlow* with a large set of inputs using a variety of mesh editing techniques, from digital sculpting to low-poly modeling, and found *3DFlow* performed well for all. Furthermore, *3DFlow* is independent of the modeling software used because it requires only mesh snapshots, and uses the additional information only for optional overlays. We release *3DFlow* as open source for artists to showcase their work and release all our datasets so other researchers can improve upon our work.

1 Introduction

3D artists commonly showcase their workflows using time-lapse videos, as screen capturing software is simple to use and requires very little interruption in the artist's work. Even for relatively simple 3D models, though, mesh editing workflows are long, ranging from tens of minutes to several hours of work, and involve thousands of operations. Time-lapse videos are not very effective at these lengths as the artist must make a trade-off between presenting the details of their workflow and keeping the presentation as short as possible. Choosing either details or brevity impacts the effectiveness of the video. Motivated by this concern, recent research has explored ways to visualize and navigate lengthy recordings of artists at work, for modeling as well as image editing. While the previous work studied ways to provide levels of interactivity beyond static images and fixed video, their systems are still limited both by the input and the output.

In this paper, we present *3DFlow*, an algorithm for producing continuous summarizations of mesh editing workflows. Our approach does not require using instrumented software, can summarize digital sculpting workflows, and is agnostic to the 3D software and editing methods used—the input needs only to be a sequence of meshes. Figure 1 shows at different levels of detail the summaries of several workflows, including low-poly modeling and sculpting sessions using dynamic or uniform remeshing. *3DFlow* is inspired by two prior works. As in *Video Tapestries* [Barnes et al. 2010], we support continuous levels of summaries to allow arbitrary temporal zooming of the editing sequence. As in *MeshFlow* [Denning et al. 2011], we add visual annotations to highlight important changes and summarize the artist's edits.

Given a sequence of meshes, *3DFlow* first detects the changes made between subsequent meshes, called *mesh deltas*, and generates a dependency graph for these deltas, called a *depgraph*, to capture the spatial and temporal dependencies of the edits. Then *3DFlow* summarizes the depgraph by repeatedly contracting the edge of least weight, computed by a cost function over the strength and distance of changes in the spatial and temporal dimensions, and merging the corresponding deltas. When only one delta remains, *3DFlow* splits the merged deltas in reverse contracting order to produce continuous levels of details.

3DFlow visualizes the workflow at any level of detail in several forms: as a video, as an image sequence or tapestry, or within an interactive viewer. Within the interactive viewer, the data can be filtered to certain regions of interest. In this paper and the supplemental materials, we demonstrate an extensive variety of rendering techniques and visual annotations, all of which are computed automatically or derived from instrumentation output. For example, *3DFlow* can highlight changes on mesh and in the timeline to emphasize the magnitude of the edit, overlay sculpting strokes as line segments, use text or images to indicate tool usage, and view the workflow from an arbitrary point of view or from the artist's orientation (if optional edit details are available).

This work contributes to the growing research space on workflow visualization in a few different ways. We propose an algorithm that robustly summarizes a sequence of meshes based on simple surface analysis regardless of the used recording method, software, length of workflow, artistic style, and editing toolsets. Our method considers both the spatial and temporal dimensions, possibly lending to more succinct summaries. *3DFlow* produces output at any level of detail, from the original sequence length down to a single step. The contributions listed above are not only helpful to artists by interactively visualizing their workflows, but also benefit the scientific community. For example, summarization aids in the understanding of workflow from a high-level, a necessary component to analyzing workflows for patterns. Furthermore, we believe that our approach is general and principled, providing a framework for summarizing workflows in other domains.

We quantitatively and qualitatively validate in two user studies the idea that *3DFlow* can be a tool for sharing summaries to their fullest extent, aiding the user to better understand how a model is built.

We refer the reader to the supplemental materials for a comparison video between *3DFlow* summaries and the fast-forwarded original sequence, a full-featured interactive viewer built for common platforms, a proof-of-concept WebGL interactive viewer, and a document containing images of all tested workflows and additional details. We release all workflow data and code for both *3DFlow* and our instrumentation as supplemental material, so that artists can take advantage of our algorithm in their daily work and so that other researchers have datasets readily available to test other approaches.

2 Related Work

Workflow Visualization and Reconstruction. Developers and users widely benefit from understanding common workflows on complicated UI. An example proposed by Terry et al. [2008] is to optimize the image editing interface for particular scenarios.

In a similar context, Kong et al. [2012] presented to users a corpus of workflows at three levels of granularity in order to understand how the users compared the workflows and which granularity was most preferred. Software users learn by studying the workflows of others through tutorials and teaching tools. For example, GamiCAD [Li et al. 2012] is an AutoCAD tutorial system for teaching first time users commonly used tools and workflow patterns. Matejka et al. [2009] proposes an algorithm and user interface that present command recommendations to the user based on history of command usage. Grossman et al. [2010] and VisTrails [2010] present systems with which users can explore the provenance of how images or 3D models were constructed. Nakamura and Igarashi [2008] present a system for visualizing user operation history with annotations. Nonlinear Revision Control for Images [Chen et al. 2011] visualizes the workflow of artists manipulating images with a focus on the non-linear relationships between operations induced by their spatial and semantic overlap. More recently, a few papers have shown complementary methods of visualizing workflows. MeshGit [Denning and Pellacini 2013] and 3D Timeline [Doboš et al. 2014] estimate and visualize mesh construction provenance as a sequence of mesh diffs with correspondence. Chen et al. [2014] presents a way to assist an artist in choosing viewpoints to showcase their 3D editing workflow. We leave the exploration of adapting the last three works into *3DFlow* for future work.

Video Summaries. Video Tapestries [Barnes et al. 2010] summarizes a video sequence into a multiscale tapestry with the ability to continuously zoom into the tapestry to expose fine temporal detail. This feature allows the summary visualization to adapt to the changes in the sequence as well as the user's preference, rather than forcing the summarized data to fit arbitrarily chosen intervals which may produce unintuitive results. We adopt a similar framework for summarizing workflows.

Polygonal Modeling Summaries. Most similar to our work, MeshFlow [Denning et al. 2011] provides summaries of mesh construction sequences by hierarchically clustering the steps in the sequence. Visual annotations are used to indicate the clustered operations performed by the artist: highlights for changed elements and overlayed visual annotations to indicate types of change.

While we take inspiration from MeshFlow, our work significantly differs in the approach to summarization and addresses key limitations. Specifically where MeshFlow uses a fixed set of rules based on editing patterns, our work provides continuous summarization of the workflows based on a cost function over edit strength and distance. We performed n-gram analyses on the digital sculpting workflows (see supplemental materials), but the results did not yield a clear set of summarization rules. We believe that MeshFlow-type summarization is not possible on digital sculpting workflows due to the vastly different editing patterns and the fact that a single sculpting tool can produce widely different effects. Moreover, because *3DFlow* uses a cost function, the input to the summarization algorithm does not require tightly-instrumented editing software—periodically saving is sufficient. As a final point of difference, MeshFlow summarizes the workflow linearly with respect to time, but *3DFlow* summarize over two dimensions (spatial and temporal) to allow for temporal reordering, producing more succinct summaries.

Stroke Summaries. When viewing a summary of the sculpting sequence, the artist's strokes are helpful for understanding how the artist worked. But for a heavily summarized sequence, the presence of all strokes obscures the object shape and remains too cluttered to provide a high level intuition. Recent work has presented ways to visualize large numbers of edges in a dense graph and to cluster artist strokes in order to provide a high-level overview of the underlying data. Holten and van Wijk [2009] show how a force-based system can organize edges in a graph visualization into bundles, which reduces the clutter and exposes underlying connections that might otherwise be obscured. When applied to our brush stroke data, we found that the artist's strokes are organized into patterns that suggest workflows not present in the original sequence. More recently, Orbay and Kara [2011] propose a method of beautifying design sketches by first clustering them and then fitting curves to the strokes. Their approach requires training of the clustering method and assumes that each stroke contributes directly to the final sketch. With our data, however, we found that the sculpting strokes affect the final result indirectly. For example, the smooth sculpting tool, used to smooth out abrupt features in the mesh, is typically used in a highly unstructured way, where the artist simply paints over a region they wish to smooth. Instead *3DFlow* de-clutters stroke display by providing continuous filtering of strokes based on the strength of the underlying edit.

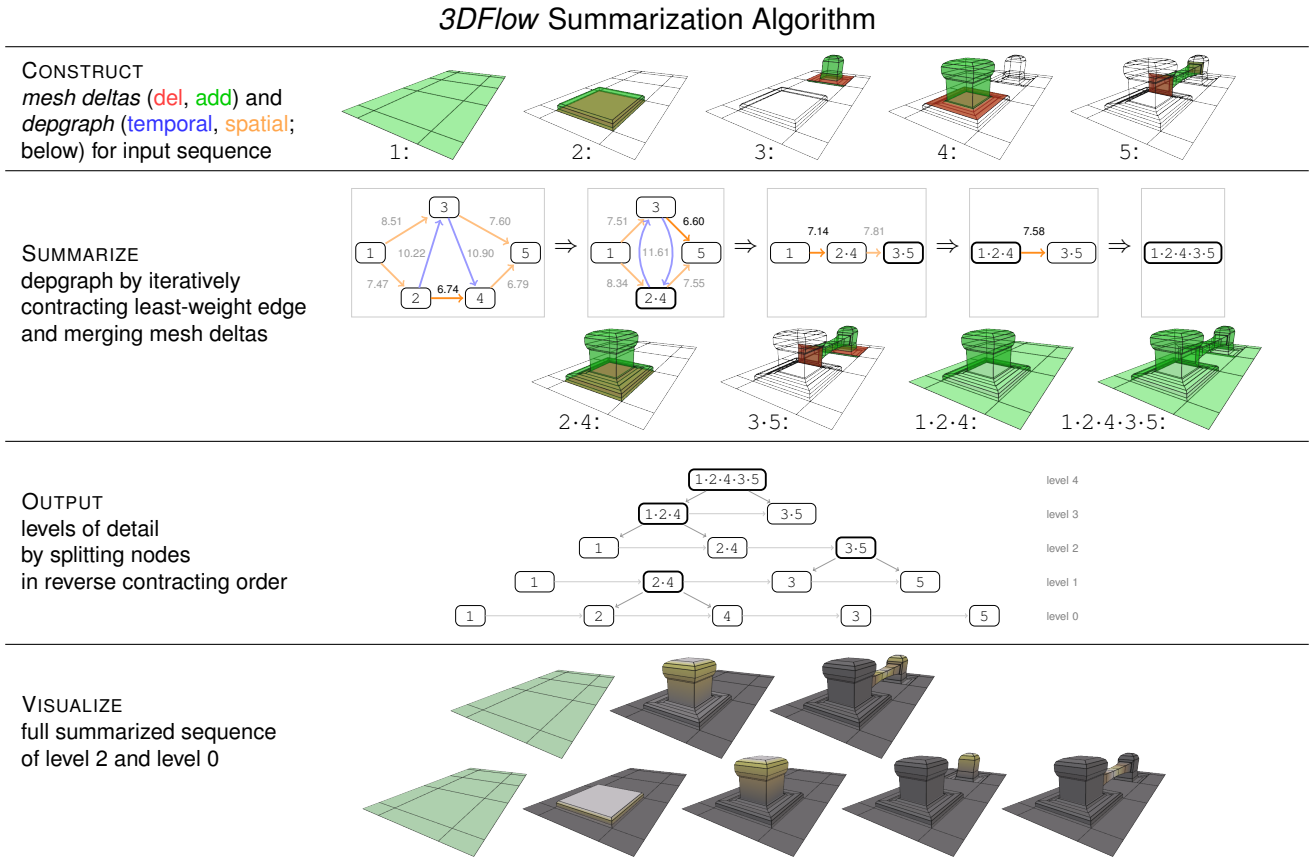


Figure 2: *The input is a sequence of meshes. In this example, each mesh is a single component and was created by performing a series of extrusions.* Mesh deltas *are constructed for each snapshot to find which faces are deleted (red) from and which are added (green) to the previous snapshot. A dependency graph (*depgraph*) is created to capture temporal (blue) and spatial (orange) dependencies with a node for each delta and a directed edge for each dependence. Every edge is weighted by the cost of merging the mesh deltas corresponding to the two nodes of the edge. We iteratively contract the least-weighted edge and merge the mesh deltas corresponding to the two nodes until no edges remain. The final remaining node corresponds to the mesh delta that is equivalent to adding the final mesh of the input sequence. Finally, we iteratively split the node(s) in reverse contracting order, creating continuous levels of details of the sequence as output.*

3 Sequence Summarization

The input to *3DFlow* is a sequence of mesh snapshots along with any associated software or edit information such as artist viewing orientation or sculpting stroke data. Note that the associated edit information is not required for summarization, as it is only used to overlay optional visual annotations to the sequence visualization. A sequence can be created in several ways by saving snapshots of the mesh after every change using instrumented software, periodically (e.g., every 5 minutes), or after every logical group of changes as is done with repository commits.

The following subsections describe the summarization algorithm in detail. Figure 2 presents an intuitive overview of this section using a simple example input sequence.

3.1 Constructing *Mesh Deltas*

We first convert the spatially normalized input into a sequence of mesh differences, which we call *mesh deltas*. A spatially normalized sequence has a union bounding box that fits in a unit cube, where at least one dimension has unit length. Each delta tracks the spatial changes and the temporal range the delta covers, which is initially a single snapshot of the sequence. More specifically we store in each delta three sets: a set of deleted faces, a set of added faces, and a set of the original snapshot indices that the delta covers. Note that we can reconstruct every original mesh by successively applying the sequence of deltas and then scaling by the inverse of the normalization factor.

We use a simple rule to build a mesh delta between two subsequent snapshots in a sequence: a face in the former snapshot that also exists in exactly the same position in the latter is considered unchanged; all other faces in former snapshot are *deleted*, and all other faces in latter are *added*. Under this rule, a transformed face is represented as a deletion of the face in the old position and an addition of the face in the new position. Despite its simplicity, this creation rule works surprisingly well. Faces do not need to be matched and tracked but only determined to be left unchanged, deleted, or added, which is inexpensive to compute and handles all types of mesh edits, including subdivision.

Two mesh deltas can be *merged* into a single mesh delta, thereby creating a summary of the original edits. The merged mesh delta is constructed using union and set difference operators. The merged set of added faces is the union of both sets of added faces minus the deleted faces, and the merged deleted set is the union of deleted faces minus the added faces. For example in Fig. 2, delta `4` deletes a face that is created in delta `2`, so the merged delta `2·4` does not include this face.

We summarize the sequence into continuous levels of details by iteratively merging mesh deltas. The following subsections describe the process of choosing which mesh deltas to merge.

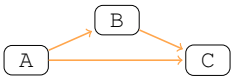
Figure 3: *Temporally reordering edits of shark sequence. A single spatially independent change on the dorsal fin (highlighted) interrupts the creation of work on the pectoral fins. Temporal reordering allows* 3DFlow *to create a more succinct summary with edits to dorsal fin and pectoral fins clustered in their respective summaries.*

3.2 Constructing a *depgraph*

A key observation is that two temporally subsequent mesh deltas may not spatially overlap, where the intersection of the set of the added faces in the former mesh delta and the set of deleted faces of the latter is empty. This implies that although one mesh delta may temporally follow another—having been performed by the artist subsequently—it is not always necessary that the deltas are merged in the same temporal order. For example, Fig. 3 shows two summaries of the shark fins construction. The artist first creates the dorsal fin and then begins working on the pectoral fins, but the pectoral fin workflow is interrupted by a single, spatially disconnected edit on the dorsal fin. The summary on the left maintains the original temporal order and therefore contains the single interrupting edit (highlighted). By temporally reordering the edits so the single, interruptive dorsal fin edit is summarized with the other dorsal fin edits, the right summary is much more intuitive and succinct.

While temporal reordering is useful, it is important to maintain spatial dependence of the mesh deltas. For example, if delta `B` deletes a face added by `A`, then temporally reordering `B` to be before `A` should not be allowed.

We build a dependency graph, *depgraph*, that captures and enforces the *temporal dependence* and *spatial dependence* of the mesh deltas. A node exists for each mesh delta, and a directed edge exists between a pair of nodes if one node depends on the other. We color the edges by the type of dependence. In order to simplify the depgraph and make summarization faster, we remove temporal edges between nodes that are also spatially dependent, and we remove any edge between two nodes that are also indirectly spatially dependent. For an example of the latter, the depgraph below shows that delta `C` depends both directly and indirectly on `A`.



We can remove the `A` $\rightarrow$ `C` edge and therefore simplify the depgraph without changing the spatial dependencies.

It is important to note that although we maintain spatial dependence, temporal dependence is still a critical data point to maintain. This note becomes obvious with workflows that create spatially disconnected meshes. Without temporal dependence, the depgraph would contain disconnected subgraphs, and while disconnected meshes are spatially independent, edits on one mesh may influence the changes of nearby meshes. For example, in order to get the shape and proportions correct when working on the eye socket area of a face mesh, the artist may insert a sphere representing the eye. Although this eye mesh is spatially independent from the rest of the mesh, its addition heavily influences the shaping of the face.

3.3 Summarizing a depgraph

We summarize a depgraph by contracting one of the edges in the graph and merging the mesh deltas corresponding to the nodes of the edge. The choice of which edge to contract (or which deltas to merge) affects the summary. For *3DFlow*, we motivate our choice with two intuitive and straightforward guidelines that apply to the temporal and spatial dimensions of the sequence:

- a merged delta should not contain too much change, and
- a merged delta should not contain edits that are too far apart.

In other words, merging deltas with strong changes may lose too many details in the summary, and merging distant deltas may divide the focus of the summary.

From these guidelines, we derive a cost function C for merging a pair of deltas `A` and `B` as a weighted sum of four terms, reflecting the two guidelines for each dimension of the data (spatial and temporal). We use the cost function to determine which edge to contract in the depgraphin order to create a summary. Note that in this notation, each delta may be the result of a previous merge of deltas. The merging cost function is defined as:

$$C(\mathtt{A},\mathtt{B}) = \underbrace{w_0 S_t + w_1 D_t}_{\text{temporal}} + \underbrace{w_2 S_x + w_3 D_x}_{\text{spatial}} \tag{1}$$

where S_t, D_t are temporal strength and distance costs and S_x, D_x are spatial strength and distance costs. Formally these individual costs are defined as:

$$S_t = \frac{|\Delta_t(\mathtt{A})| + |\Delta_t(\mathtt{B})|}{\text{avg}\,|\Delta_t|} \tag{2}$$

$$D_t = \min_{a,b \in \Delta_t(\mathtt{A}) \times \Delta_t(\mathtt{B})} \frac{|a-b|-1}{\text{avg}\,|\Delta_t|} \tag{3}$$

$$S_x = \frac{|\,\text{area}[\Delta_x^+(\mathtt{A}\cdot\mathtt{B})] - \text{area}[\Delta_x^-(\mathtt{A}\cdot\mathtt{B})]|}{\max(\text{area}[\Delta_x^+(\mathtt{A}\cdot\mathtt{B})], \text{area}[\Delta_x^-(\mathtt{A}\cdot\mathtt{B})])} \tag{4}$$

$$D_x = \min_{u,v \in \Delta_x(\mathtt{A}) \times \Delta_x(\mathtt{B})} \text{min-dist}(u,v) \tag{5}$$

where $\Delta_t(\mathtt{A})$ is the set of original delta indices covered by delta `A`, $\Delta_x^+(\mathtt{A})$ is the set of faces added by `A`, $\Delta_x^-(\mathtt{A})$ the set of faces deleted by `A`, $\Delta_x(\mathtt{A})$ the set of faces either added or deleted by `A`, the dot operator $(\cdot)$ indicates a merging of deltas, $\text{avg}\,|\Delta_t|$ computes the average size of snapshot indices sets for the deltas in the depgraph, area is a function that returns the total surface area for a given set of faces, and min-dist is a function that returns the minimum Euclidean distance between the given faces.

The temporal strength term, S_t, is the total number of original snapshots covered by merging deltas `A` and `B`. The temporal distance term, D_t, is defined as the minimum temporal distance between the `A` and `B`. This term is computed as the minimum absolute difference between all snapshot indices of `A` and of `B` minus one. For example, if delta `A` covers snapshot 1 and `B` covers snapshots 2 and 4, the temporal distance cost of merging `A` and `B` is 0. Both of the temporal terms are regularized by the average number of snapshots covered by the deltas to prevent the temporal terms from dominating the cost function.

The spatial strength term, S_x, is the absolute net change in surface area after merging both `A` and `B`, regularized by dividing by either the net added surface area or the net deleted surface area, whichever is larger. The denominator regularizes spatial changes to be relative to the size of region affected. In other words, spatial changes that are small in the absolute sense are relatively large if they affect a small region, and large spatial changes that affect large regions may be relatively small. The spatial distance term,

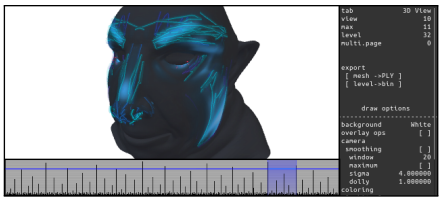


Figure 4: *User interface for* 3DFlow. *The top-left panel visualizes the mesh at the selected level of detail with visual annotations to indicate surface changes and brush strokes. The bottom-left panel shows the timeline. Visualization options are set in right panel.*

D_x, is the minimum Euclidean distance between the added and deleted faces of A and the added and deleted faces of B. Note that the spatial distance term is already regularized when the input was processed to fit in a unit cube.

The four terms of Equation 1 address the two guidelines mentioned earlier across both dimensions of the data. Each of the terms are linearly weighted to emphasize different types of clustering. For example, setting w_0 to 1 and the remaining weights to 0 will allow for hierarchical uniform clustering. We experimentally found the weights 2, 1, 4, and 14 (respectively) work well to give intuitive results across all shown datasets. All figures in this paper and the supplemental materials use these weights.

We consecutively summarize the depgraph, recording the order of edges we contract, until only one node remains. The delta corresponding to the remaining node covers all of the original deltas (possibly reordered) and adds all of the faces of the final mesh. As a note, *3DFlow* can optionally hold out summarizing the initial mesh (e.g., cube, bust, etc.) until the very last step. This holding out may produce more intuitive summaries, as the base mesh is visible in its original form for all summary levels except for the highest.

3.4 Outputting Levels of Detail

We create the highest summary level as a single delta, the delta corresponding to the single remaining node. This single node is then split into two nodes according to the last edge contraction performed during summarization. Note that the contracted edge encodes the dependence of the nodes, and we maintain this dependence by placing the dependent node temporally after the other node. The corresponding deltas of these two nodes define the second highest summary level. Now, we repeatedly split the nodes in reversed order of edge contraction to produce continuous levels of detail. Reconstructing the deltas in this manner produces linear, but also hierarchical, levels of detail.

3.5 Discussion

We chose to define our cost function using surface area of deltas to measure shape differences since, compared to other metrics (see [Pottmann et al. 2009; Silva et al. 2009] for a review), it is efficient to compute, it is well-defined even on non-manifold meshes or meshes with holes, and it does not require a registration between two meshes beyond finding which faces have been altered. Despite the simplicity of the terms introduced above, we found that the cost function worked well over a range of sculpting and polygonal modeling datasets. Furthermore, we tested more expensive cost functions (e.g., mean curvature, volume delta, hausdorff distance, distance between corresponding points), and found that they did not improve upon the results enough to warrant the additional computation. We leave further investigations to future work.

In *3DFlow*, we do not consider the category or name of the edit operation or even editing patterns when clustering. We did perform n-gram analysis on the digital sculpting workflows (see supplemental materials), but it is unclear how to construct

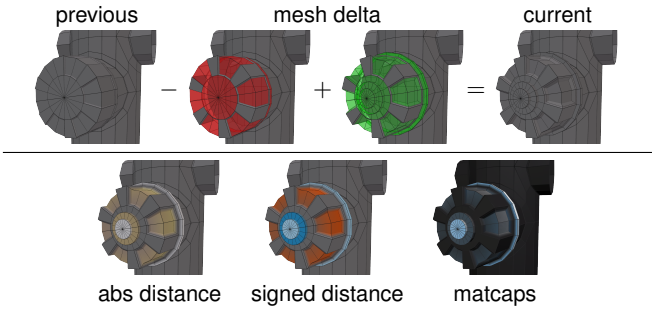


Figure 5: *Emphasizing surface changes in mesh delta. Applying the mesh delta (top-middle, 31 edits) to the previous mesh (top-left) results in the current mesh (top-right). The bottom row shows three different ways to highlight and emphasize the magnitude and direction of changes to the surface. See Sec. 4 for more details.*

clustering patterns similar to those by Denning et al. [2011] that would produce intuitive results. Furthermore, by only considering the edited region and not the name or category of edit operation, *3DFlow* can summarize more general workflows such as those in which instrumentation was not used. For an example see the supplemental material where we used as input to *3DFlow* every version of the character Sintel from the Subversion repository of the open movie Sintel [Blender Foundation 2011].

Limitations. While we believe that Equation 1 performs well in regards to our guidelines, it does not capture the semantic of an edit. For example, it might make sense to cluster together edits that work on the eyes or those that add wrinkles across the face, but the formulation above does not infer any semantical meaning from the edit itself or from the region being changed.

Finally, although the spatial distance computations are highly parallelizable and many other computations can be cached, the nature of greedily choosing a single edge to collapse in the depgraph imposes sequential constraint on the algorithm. We focused on computing accurate values or highly accurate approximations when possible, and we leave further optimization for future work.

4 Visualizations

In this section, we describe some of the ways we visualize different features of the data. We also discuss a few ways for a viewer to interact with the data.

Basic User Interface. Figure 4 shows the user interface. To maintain simplicity, we use a basic layout that is similar to a simple video player. At the top-left is the main 3D view, where the mesh is seen at the selected time and level of detail. Regions of the mesh that are altered by the selected delta are highlighted in blue. The timeline at the bottom-right acts much like a scrub bar in a video player. The vertical axis of the timeline is the level of detail, with highest summary at the top and greatest details (deltas of original sequence) at the bottom. Black vertical lines indicate where each delta begins and ends. The blue vertical bar indicates the coverage of the selected delta, and the blue horizontal bar indicates the selected level of detail. The visualization options on the right allow the viewer to control how the mesh is rendered.

While *3DFlow* generates continuous levels of detail from every delta down to a single delta, by default we simplify the user interface to show only a subset of the levels. We choose the levels that are at a log-scale of the original deltas (all, half, quarter, etc.), and then we add the levels with 2–20 deltas and the levels with odd number of deltas in the 20–50 range. This simplification can be turned off.

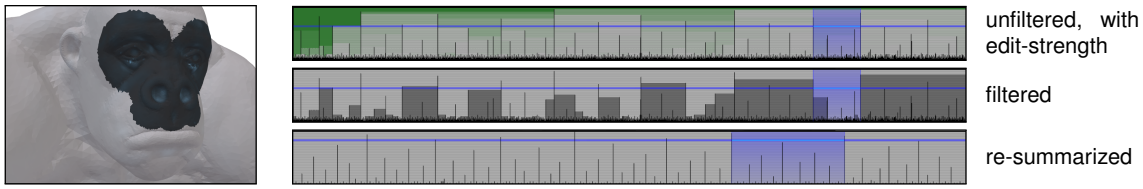


Figure 6: *Edit-strength annotations and spatial filtering on gorilla sequence. The mesh on the left is partially deemphasized to indicate the selected regions. The top timeline shows the original unfiltered workflow with the viewed step highlighted in blue and the edit-strength annotation enabled, the middle shows the workflow with spatial filtering enabled, and the bottom shows the workflow filtered and re-summarized. Edit-strength annotations shade deltas in green to indicate the magnitude of change in surface area. Spatially-filtered deltas which do not modify the selected region are darkened and are not viewable.*

Highlighting Changes. The changes in a mesh delta are emphasized by highlighting the added faces, where the *magnitude* of the change modulates the visual strength of the highlight. For each delta, we approximate a magnitude of change for each vertex of an added face as the minimum distance between the vertex to the surface defined by the deleted faces. If in a delta no faces were deleted, then all of the vertices of the added faces are marked as *added*. This can happen, for example, whenever the artist creates disconnected geometry. We visualize added geometry in green and modified geometry by using it as a mixing value. To adapt highlighting for edits that are globally large (e.g., creating a large appendage) and for edits that are globally small but locally large (e.g., adding wrinkles), *3DFlow* can individually rescale the magnitudes by the local or global maximum. Rescaling highlights locally helps keep small refinements visible even when summarized with large changes.

3DFlow offers several highlighting options for the vertices. Figure 5 demonstrates a few different possible visualizations which are briefly explained below. One option is to linearly map the magnitude to a color gradient, where unchanged vertices are colored a neutral gray, moderately changed vertices are yellow, and vertices with strong magnitude of change are white. A multi-color gradient provides better resolution to help resolve strong changes from minor changes. Another option is choosing different color gradients based on the *sign* of change. Specifically, the vertex has a *positive* change if it was moved "outside" the deleted surface and *negative* if moved "inside", where sidedness is determined by the surface normal. Positive changes are colored blue, while negative changes are colored orange. This option of highlighting visualizes the approximate magnitude and direction the vertex was moved, giving a sense of the change in volume. Lastly, rather than mapping the magnitude to a color gradient, the magnitude can influence a mixing value between two *matcaps*. Matcaps simulate complex material and lighting setups and are often used to help sculpting artists focus on certain characteristics of the surface. For example, increasing the material's specularity can emphasize high-frequency details and creases.

Visualizing Sculpting Annotations. While highlighting indicates how much the mesh has changed, it is not very descriptive of which sculpting tool the artist used or how the tool was used. When tool usage metadata is provided, *3DFlow* can visualize the artist's tool usage by overlaying visual annotations. In *3DFlow*, we visualize the artist's sculpting strokes as lines drawn over the mesh. Because the sculpting strokes may fall inside or behind the mesh, we render the strokes in two passes: once with a thick, transparent line without performing depth tests, and then another with a thin, opaque line with depth testing. The first pass allows the viewer to see strokes that are obscured by the mesh without adding too much clutter, and the second shows details. Strokes are colored by brush type: pulling in blue, smoothing in cyan, creasing in orange, and grabbing or nudging in pink. Although we visualize only the sculpting strokes, visualizing other types of edits, such as extrude edge and merge vertices, can be easily added to *3DFlow*.

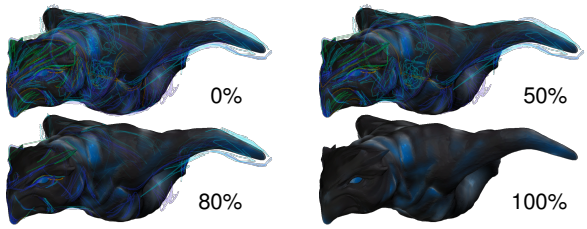


Figure 7: *Filtering annotations. The mesh is heavily obscured when visualizing the sculpting stroke annotations of all 343 merged deltas (top-left).* 3DFlow *sorts the annotations by magnitude of change and continuously filters them from showing all to none.*

Filtering Annotations. As the number of covered deltas increases, visualizing all of the tool annotations can obscure the view of the mesh and may overwhelm the viewer. Similar to providing levels of detail and summary of mesh deltas, *3DFlow* provides continuous levels of detail and summary for tool annotations through filtering. Filtering removes the annotations that change the mesh the least. The filtering can be continuously adjusted to show any number of annotations from all down to none. Each edit annotation is assigned a weight equal to Equation 4 of the corresponding delta. The annotations are sorted by their weight, and *3DFlow* visualizes only the annotations with an order that is above a user-specified threshold. Figure 7 shows the effect of filtering tool annotations at varying levels, where 0% filtering shows all tool annotations, 50% shows only half of the annotations, and 100% shows none.

We considered two clutter-reducing alternatives to sculpting stroke annotation filtering: determine a representative through spatial clustering or perform edge-bundling [Holten and Van Wijk 2009]. Unfortunately we found that these alternatives were of little help for uncorrelated tool usage or suggested tool usage patterns that were not representative of the artist's workflow, as in the case of spatially-close sets of correlated edits.

Spatial Filtering. In order to help the viewer find deltas that modify particular spatial regions, *3DFlow* provides spatial filtering. When the viewer clicks on the mesh, every face in the entire sequence that is within a given radius of the point on the mesh is selected. Unselected regions of the mesh are deemphasized in the main 3D view by desaturation and brightening. All deltas that do not affect a selected face are darkened in the timeline and made unviewable, indicating to the viewer when the selected region was modified. This filtered workflow can then be re-summarized as a new, customized view of the workflow. Figure 6 shows the timeline filtered to the deltas that modify the face of the gorilla.

Edit-Strength Annotations. Users may wish to have a summary of the strength of edits along a workflow. Determining this feature allows the user to find quickly when the artist made large changes or small refining changes to the mesh. *3DFlow* can optionally shade the timeline to indicate the edit strength. In Figure 1, the first three steps summarize large surface changes to block out the form of the gorilla, while the last four steps summarize the addition of

details and small features. In the second timeline of Figure 6, the strength of green linearly corresponds to the magnitude of surface area change.

Other Visualization Options. We refer the reader to the supplemental material for a demonstration of other visualization options. These include: rendering the summarized workflow using external software; rendering with a mirror effect to see edits on front and back sides of mesh at the same time; smoothly interpolating or warping the surface to simulate the artist's summarized work; and centering on and zooming into the region of the mesh that is edited. We leave a more exhaustive investigation of visualization options for future research.

5 Results

In this section we report about the input workflows and briefly discuss the results. In the following section we report on results from two user studies.

Input Workflows. We tested *3DFlow* on a variety of mesh editing workflows, shown throughout the paper and in supplemental material. Source code and all datasets are available in supplemental material. Table 1 summarizes statistics for all of the input workflows.

Our sculpting data was obtained by two professional artists using an instrumented version of Blender. One artist has a stronger tendency to explore while editing, making strong changes often throughout the sequence, while the other artist prefers to make strong edits first then refine. Both artists sculpted using both uniform and dynamic remeshing to control mesh resolution. Workflow lengths in terms of the number of sculpting edits varies from several hundreds to a few thousand. The initial meshes consisted of a cube, a generic human bust, and a full-body human basemesh.

The helmet, hydrant, robot, shark, and biped polygonal modeling workflows were imported from the MeshFlow dataset, which is publicly available online. The durano and creature workflows are from two Blender Open Movie Workshop DVDs, *Venom's Lab!* [Vazquez 2009] and *Creature Factory* [Goralczyk 2008], respectively. The sintel [Blender Foundation 2011] workflow is from the Subversion repository of the open movie *Sintel* [Roosendaal 2011] available online. All workflows were used directly without processing or manual filtering.

Discussion. We compare results of summarizing the biped workflow using *3DFlow*, uniform intervals (similar to a timelapse), and MeshFlow in Figure 8. Due to having continuous summarization, *3DFlow* and uniform intervals can summarize the workflow anywhere down to a single step, while MeshFlow can only summarize to discrete steps because of using fixed clustering rules. In this example, we summarized the workflow to ten steps for *3DFlow* and uniform intervals and twenty steps for MeshFlow (the minimum possible number of steps for this data). The timelines below the rows of meshes report the coverage of deltas for each workflow summary. Notice that *3DFlow* summarizes changes into small, localized groups, such as the main figure, head, feet, etc. On the other hand, uniform intervals and MeshFlow summaries contain merged edits that are spatially distant (e.g., mixing edits to feet and hands) or contain many strong edits (e.g.,, the first step of uniform summary and the tenth step of MeshFlow). Another important note is that in the original sequence, the hands were created before the feet, but the arms shortened last. With temporal reordering, *3DFlow* summarized together all of the edits to the forearm and hands.

Figure 10 shows five sculpting workflows that started with a base mesh and used subdivision remeshing. One artist created the merman, engineer, and sage workflows, and the other artist created the alien (also from a cube with subdivision; see Fig. 1), fighter, and man workflows.

	model	fig.	deltas	faces	time
subdivision sculpting	*ogre*	4	1459	1,660,475	1:26
	merman	10	2218	2,171,310	3:40
	sage	10	1686	1,961,133	2:19
	engineer	10	863	2,919,865	2:54
	elder		2958	1,500,632	3:01
	alien	1	2118	6,094,173	8:49
	man	10	1459	1,953,859	3:03
	fighter	10	1532	1,156,686	2:06
dynamic sculpting	*gargoyle*	7	819	1,090,882	0:33
	monster	1	797	1,389,906	0:47
	elf		4125	4,791,845	2:46
	gorilla	1	2482	4,241,528	3:32
	explorer		1699	3,416,354	2:21
polygonal modeling	*helmet*	1	1321	17,579	0:05
	hydrant	5	691	49,892	0:04
	robot		1810	139,527	0:15
	shark	3	1457	19,177	0:06
	biped	8	1267	18,162	0:05
	*durano**	1	11	7,165	0:01
	*creature**		123	280,338	0:14
	*sintel**		210	2,948,611	2:11

Table 1: *Statistics of input workflows. The workflows are grouped by editing types: digital sculpting with uniform tessellation (top 8), digital sculpting with dynamic tessellation (middle 5), and low-poly modeling (bottom 8). Workflows were either recorded with instrumented software or generated from committing versions (starred). The* deltas *column reports length of workflow, and* faces *reports total count of added faces. The final column indicates how much processing time (mm:ss) was needed to summarize the workflow. All meshes are shown and cited in supplemental materials.*

Assumption Validation. The aim of our work is to find a way to better explain to a user how a 3D model is built. To validate the interactive viewer with summarizing workflows, we ran two user studies and collected spontaneous feedback from various professional modelers. All the responses we received confirm our assertion. We report the obtained results in Section 6.

Future Work. We tested *3DFlow* with a large set of workflows across a variety of techniques. There are several other common and interesting mesh editing workflows that we did not try, including retopologizing and sculpting using Boolean operations. We plan to extend the techniques developed with *3DFlow* to summarize these types of workflows as well as workflows that change the properties of the mesh, such as texturing or rigging, or workflows that modify full-scene data. When summarizing workflows, *3DFlow* does not consider the type nor the technical complexity of the edit operations performed. Further *3DFlow* does not consider the context of edits, e.g., adding wrinkles to forehead versus shaping the eye socket. We plan to investigate these areas in the future.

6 User Study

To validate the effectiveness of *3DFlow*, we ran two user studies that differ in terms of subjects and questions. In the first study, we quantitatively measure whether *3DFlow* can better explain how the model was built than using existing methods. For this experiment, we use novice users since we want to measure the impact of *3DFlow* on understanding 3D modeling. In a second experiment, we qualitatively investigate how *3DFlow* would impact expert's workflow, both as a documentary tool, and as a summarization and teaching tool for others.

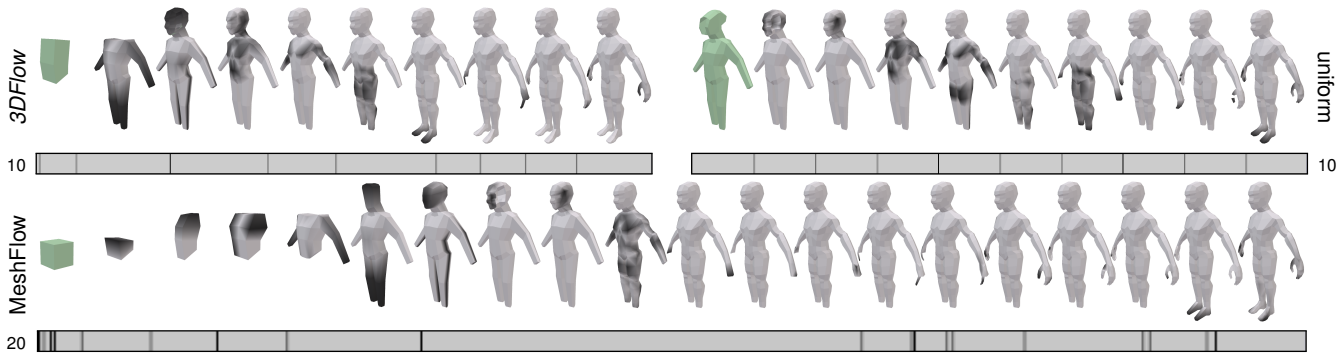


Figure 8: *Comparing summaries produced by* 3DFlow *(top-left), uniform intervals (top-right), and MeshFlow (bottom). Changes are highlighted in black, and the timelines show the coverage of deltas for each summary. The shortest summary MeshFlow can produce of the biped sequence is 20 steps, while* 3DFlow *and uniform intervals can have any length. See Sec. 5 for detailed analysis of this figure.*

model	prob.
global	0.730
alien	0.786
biped	0.629
engineer	0.887
explorer	0.717
gargoyle	0.698
gorilla	0.630
merman	0.736
monster	0.792

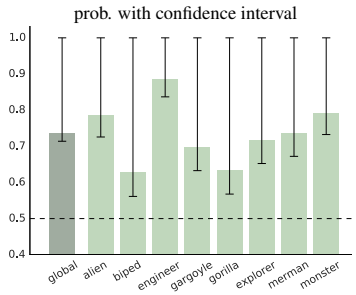


Figure 9: *User study results. We report the probability for the model to be chosen. The first row and the first bar indicate measures on aggregate values. For each model the computed p-value is* 2.2^{-16} *and* $\mu > 0.6$*. The bars indicate the confidence intervals.*

6.1 Quantitative Comparison

To test user comprehension, we chose to have viewers compare the summarization algorithm against a standard timelapse. We compare videos rather than using the full interface since we selected users novice enough that they would likely not benefit from the use of the full interface. We asked subjects to watch videos that show side-by-side a timelapse recording of the full editing session and a clustered version generated by our method. We generated the timelapse with fixed camera since Chen et al. [2014] shows how camera movement significantly hinders comprehension. We show the cluster of operations with highlights only. After the video, we ask the user for a preference between the two sides with the question, "Which video explains better how the model was built?". We purposely did not over-characterize the question to avoid bias.

We ran the experiment on Amazon Mechanical Turk. For ensuring experiment qualities, we pay each HIT $0.30 USD, tuning price as Kaufmann et al. [2011] propose, and we allow just high-rated users to attend the experiment. We tested on a sample of 160 Amazon workers. Each task consisted of eight videos, with their order and left-to-right positions randomized, and each user had only one opportunity to complete the experiment. We disabled highlights in the timelapses since they appear as distracting "flickering" flashes due to the number of operations per second and the lack of clustering. For avoiding Good-Subject-Effect bias, we picked novice subjects. In this way, we assume that most of them have never seen a sped up timelapse nor our work: without knowing which is the novelty, they do not know which video is the one we propose so they cannot be affected by the bias.

We ran a Binomial Exact Test on the results for test significance, where the null hypothesis is that choosing one side of the video has the same probability of the other. In Fig. 9 we show the results of the test with their confidence interval: for all the models, we reject the null hypothesis with a global $p = 2.20e^{-16}$. The mean among the models is $\mu = 0.786$. This implies with high statistical confidence that users prefer the clustered video for understanding how the model was built, concluding that *3DFlow* can efficiently convey information about this task.

6.2 Qualitative Expert Feedback

We interviewed five expert artists that are also instructors of digital sculpting. We showed them our prototype in action, then we provided a questionnaire. We recorded their assessment of how *3DFlow* could impact their day-to-day work. We validated in the questionnaire that our subjects have confidence with digital sculpting tools and have previous experience with summarization tools both for sharing and learning. Our subjects confirmed that timelapse videos are the most used method for sharing and watching workflows.

In the questionnaires, the most mentioned problem with timelapses was selecting the ideal speed, which is not uniform across editing sessions varying from details to general changes on meshes. All experts confirmed that they would prefer using *3DFlow*, particularly for the interactivity of the summarization. All of them requested to be contacted when the work of *3DFlow* is published.

All interviewees expressed that *3DFlow* would be useful for sharing their workflow with others using the interactive workflow along with a traditional video or document tutorial, with all rates greater than 4 in a scale from 1 to 7 ("very unhelpful" to "helpful", resp.), confirming that there is interest in how *3DFlow* can benefit their work.

Here is a sample of experts' comments when asked to give a personal opinion about *3DFlow* (we include all questionnaires and responses in supplemental). "This program has amazing application potential when it comes to 3D education. I would consider this for use in my classroom." "This has a huge amount of potential for teachers and students alike." "I'd be very interested in trying this out as an instructor." "As a tutorial creator I've often thought that something like this would be helpful in addition to the regular tutorial media options." "This looks awesome!"

Informal Feedback. We asked for feedback from the two professional artists who authored the sculpting workflows. They found the summarizations captured the workflows quite well, and both agreed that *3DFlow*'s interactive viewer with summarized workflow is a significant improvement. One artist commented, "I've recently finished working on the materials for a sculpting course I'm teaching. Having 3DFlow available would have allowed students to better visualize changes to the mesh." The other artist

commented that it is astonishing to see how *3DFlow* breaks down the workflow process. We also presented our work to the CEO of the leading online platform for publishing and sharing 3D content. His response was, "We are always looking to share and showcase tutorials on the working process and workflows of our best users, and how they manage to get to beautiful results [...] So far, we have three ways to illustrate it: photos, 3D models with [our site], and videos. A way to let them record and share the modeling process in 3D in real-time would be a killer feature."

7 Conclusion

We presented *3DFlow*, an algorithm for providing continuous summarizations of mesh editing workflows. *3DFlow* summarizes the input sequence of meshes by constructing a corresponding dependency graph where nodes represent changes to the mesh and edges the spatial and temporal dependence of the edits, iteratively contracting the least-weighted edge according to a cost function until only one node remains, and then splitting the nodes in reverse order into levels of detail. The visualization of the workflow is enhanced by highlighting the changed regions and (optionally) overlaying visual annotations describing the artist's edits. We tested *3DFlow* with a large set of mesh editing workflows from a variety of sources and found *3DFlow* performed well with all. We validate the idea that *3DFlow* can be a tool for sharing and fruition of summaries, making users better understand how a model is built. All source code and data is released as open source.

8 Acknowledgments

We would like to thank the authors of the workflows used, the participants of the user studies and expert interviews, the editors at the Taylor University Writing Center, and the reviewers for their constructive and helpful feedback. This work has been partially supported by the NSF (CCF-0746117), the Sloan Foundation, the European Commission 7th Framework Programme (project TROPIC), and the Intel Corporation.

References

BARNES, C., GOLDMAN, D. B., SHECHTMAN, E., AND FINKELSTEIN, A. 2010. Video tapestries with continuous temporal zoom. *ACM Trans. Graph. 29* (July), 89:1–89:9.

BLENDER FOUNDATION, 2011. Sintel. www.sintel.org.

CHEN, H.-T., WEI, L.-Y., AND CHANG, C.-F. 2011. Nonlinear revision control for images. *ACM Transaction on Graphics 30*, 4, 105:1–105:10.

CHEN, H.-T., GROSSMAN, T., WEI, L.-Y., SCHMIDT, R., HARTMANN, B., FITZMAURICE, G., AND AGRAWALA, M. 2014. History assisted view authoring for 3D models. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '14.

DENNING, J. D., AND PELLACINI, F. 2013. MeshGit: Diffing and merging meshes for polygonal modeling. *ACM Transaction on Graphics 32*, 4.

DENNING, J. D., KERR, W. B., AND PELLACINI, F. 2011. MeshFlow: interactive visualization of mesh construction sequences. *ACM Transaction on Graphics 30*, 4, 66:1–66:8.

DOBOŠ, J., MITRA, N. J., AND STEED, A. 2014. 3D Timeline: Reverse engineering of a part-based provenance from consecutive 3d models. *Eurographics Symposium on Rendering 33*, 2.

GORALCZYK, A., 2008. Creature. Creature Factory Blender Open Movie Workshop, vol. 2.

GROSSMAN, T., MATEJKA, J., AND FITZMAURICE, G. 2010. Chronicle: capture, exploration, and playback of document workflow histories. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '10, 143–152.

HOLTEN, D., AND VAN WIJK, J. J. 2009. Force-directed edge bundling for graph visualization. *Computer Graphics Forum 28*, 3, 983–990.

KAUFMANN, N., SCHULZE, T., AND VEIT, D. 2011. More than fun and money. worker motivation in crowdsourcing–a study on mechanical turk.

KONG, N., GROSSMAN, T., HARTMANN, B., AGRAWALA, M., AND FITZMAURICE, G. 2012. Delta: a tool for representing and comparing workflows. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '12, 1027–1036.

LI, W., GROSSMAN, T., AND FITZMAURICE, G. 2012. GamiCAD: a gamified tutorial system for first time autocad users. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '12, 103–112.

MATEJKA, J., LI, W., GROSSMAN, T., AND FITZMAURICE, G. 2009. CommunityCommands: command recommendations for software applications. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '09, 193–202.

NAKAMURA, T., AND IGARASHI, T. 2008. An application-independent system for visualizing user operation history. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, ACM, New York, NY, USA, UIST '08, 23–32.

ORBAY, G., AND KARA, L. B. 2011. Beautification of design sketches using trainable stroke clustering and curve fitting. *IEEE Transactions on Visualization and Computer Graphics 17*, 5 (May), 694–708.

POTTMANN, H., WALLNER, J., HUANG, Q.-X., AND YANG, Y.-L. 2009. Integral invariants for robust geometry processing. *Comput. Aided Geom. Des. 26*, 1 (Jan.), 37–60.

ROOSENDAAL, T., 2011. Durian open movie project : Sintel full studio SVN online. www.sintel.org/news/sintel-full-studio-svn-online.

SILVA, S., MADEIRA, J., AND SANTOS, B. S. 2009. PolyMeCo—an integrated environment for polygonal mesh analysis and comparison. *Computers & Graphics 33*, 2, 181 – 191.

TERRY, M., KAY, M., VAN VUGT, B., SLACK, B., AND PARK, T. 2008. Ingimp: introducing instrumentation to an end-user open source application. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '08, 607–616.

VAZQUEZ, P., 2009. Durano model. Venom's Lab Blender Open Movie Workshop, vol. 4.

VISTRAILS, 2010. VisTrails provenance explorer for Maya. www.vistrails.com/maya.html.

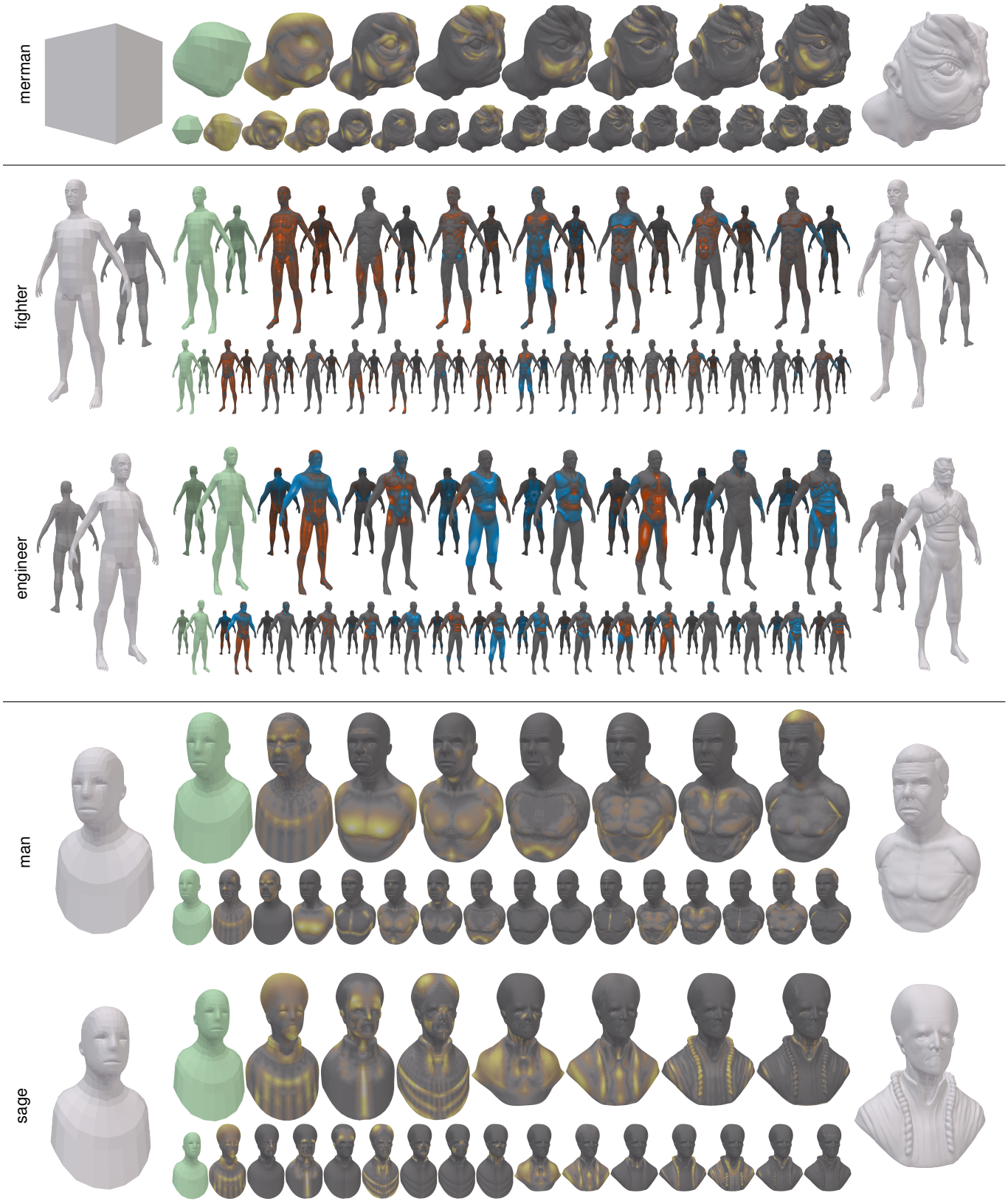


Figure 10: *Five sculpting workflows summarized in 8 and 16 steps. These workflows started with a base mesh (left column) and used subdivision remeshing. The initial and final meshes (right column) are shown without highlighting. The fighter and engineer workflows are visualized with a mirror effect to show both sides of the mesh.*