6-2014

3DFlow: Continuous Summarization of Mesh Editing Workflows

Jonathan D. Denning
Dartmouth College

Fabio Pellacini
Sapienza University of Rome

3DFlow: Continuous Summarization of Mesh Editing Workflows

Jonathan D. Denning
Dartmouth College

Fabio Pellacini
Sapienza University of Rome

June 2014, Dartmouth Computer Science Technical Report TR2014-757

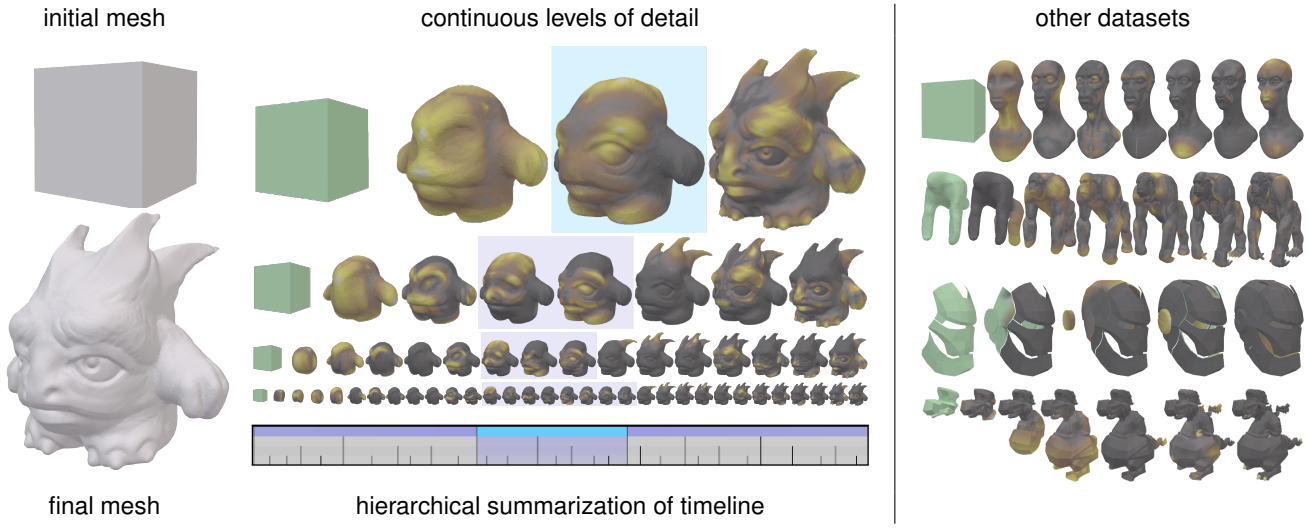


Figure 1: *Continuous levels of details automatically constructed from a 30 minute digital sculpting session of a professional artist. The artist sculpted the cube (top-left) into a monster (bottom-left) in 797 strokes using dynamic remeshing techniques. The center column shows the sequence summarized 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 continually improving allowing more detailed meshes to be create efficiently by skilled artists. Many of these are interested in sharing not only the final mesh, but also their whole workflows both for creating tutorials as well as for showcasing the artist's talent, style, and expertise. Unfortunately, while creating meshes is improving quickly, sharing editing workflows remains cumbersome since time-lapsed or sped-up videos remain the most common medium. 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 since it requires only mesh snapshots, using additional information only for optional overlays. We open source *3DFlow* for artists to showcase their work and release all our datasets so other researchers can improve upon our work.

1 Introduction

Various methods are used for learning how talented artists create polygonal meshes. Although document-based tutorials are an option, artists commonly showcase their workflows via a time-lapse or sped-up video recording of their editing session, since these videos are simple to create without interrupting their workflow. Even for relatively simple models, though, mesh editing workflows are long, ranging from tens of minutes to several hours of work, involving thousands of operations. Time-lapses are not very effective for these lengths since the artist must make a trade-off between presenting the details of their workflow and keeping the presentation as short as possible. 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. For example, *VisTrails* [VisTrails 2010] helps in navigating non-linear undo histories in 3D software, while Chen et al. [2011] present non-linear navigation of edits in images. *MeshFlow* [Denning et al. 2011] combines clustering of edits with annotations to get a summary of a polygonal modeling session. *Delta* [Kong et al. 2012] helps in comparing workflows in image editing. ZBrush [Pixologic 2013] has a workflow playback feature just for creating time lapses.

In this paper, we focus on summarizing mesh editing workflows, including digital sculpting and low-poly modeling. In sculpting, artists alter the shape of a mesh as though they were sculpting a block of clay using physical tools. The digital brushes can have different effects, such as creating new features, smoothing out uneven areas, or reposing parts of the mesh. Sculpting is particularly well suited for modeling organic shapes like characters. In low-poly modeling, artists directly manipulate the surface representation of the mesh by issuing commands such as extrude edge, split face, add new cube, etc. This workflow is particularly well suited for modeling hard-surface objects, meshes that will be animated or base meshes for subdivision surfaces.

There are two major working phases with modeling workflows: *blocking* and *refinement*. In blocking, the main shape of an object is roughed out. Blocking edits have strong magnitude and are applied over large regions usually relatively quickly. Finer details are carefully added during refinement. These details are more

precise and are repeated many times over smaller areas. In a sense, blocking and refinement edits work at different scales, both spatially and temporally.

In this paper, we present *3DFlow*, an algorithm for providing continuous summarizations of mesh editing workflows. Figure 1 shows at different levels of detail the summaries of several workflows, including low-poly modeling and sculpting sessions using dynamic or subdivision 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.

3DFlow takes as input a sequence of meshes with optional annotations, such as brush strokes, and outputs a continuously summarized mesh sequence with visual annotations. To do so, we first compute *mesh deltas*, one for each input mesh, that describe the changes performed in the current edit. A dependency graph, *dep-graph*, is constructed with nodes for each delta and edges representing the spatial and temporal dependencies of the deltas. We then repeatedly contract the edge of least weight, computed by a cost function over the strength and distance of changes in the spatial and temporal dimensions, and merge the corresponding deltas to produce continuously summarized dependency graphs. When only one delta remains, we split the merged deltas in reverse contracting order to produce continuous levels of detail. In the interactive viewer, we highlight changes to the mesh to emphasize the magnitude of the edit and, if supplied, overlay visual annotations to illustrate the artist's edits, such as summarized brush strokes for sculpting.

We tested *3DFlow* using digital sculpting sessions by professional artists obtained with a lightweight software instrumentation, the polygonal modeling sessions from *MeshFlow*, and committed snapshots from movie and tutorial production files [Goralczyk 2008; Vazquez 2009; Blender Foundation 2011]. The sculpting artists modeled a variety of organic models, from detailed heads to full bodies, with different workflows based on their personal preference, and using uniform subdivision remeshing or adaptive, dynamic remeshing. The length of sequences generated from instrumented software varied from several hundred to a few thousand individual edits, while those generated from production repositories varied from about ten to a couple hundred. We found that *3DFlow* worked well across all the datasets tested. We refer the reader to the supplemental video for a comparison between *3DFlow* summaries and the fast-forwarded original sequence. We release all workflow data as well as 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. As software packages for image and 3D scene creation become more complicated, both developers and users benefit from understanding common workflows. Developers can optimize the user interface for particular usage scenarios, as proposed by Terry et al. [2008] in the case of image editing. 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. Chen et al. [2014] present a way to assist an artist in choosing viewpoints to showcase their 3D editing workflow.

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. Two types of visual annotations are used to indicate the operations performed by the artist that were clustered: highlighting changed elements and overlaying visual annotations to indicate types of change. For example, when a face extrusion followed by vertex movements are clustered together, the individual operations are still visible to the user by highlighting the moved vertices, coloring the newly created face, and drawing an arrow to indicate direction of extrusion. While we take inspiration from MeshFlow, our work significantly differs in the approach to summarization and addresses key limitations of their work. Specifically, our work provides continuous summarization of the workflows based on a cost function over edit strength and distance, where MeshFlow uses a fixed set of rules based on editing patterns. We performed n-gram analyses on the digital sculpting workflows (available in 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 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. 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 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 get organized into patterns that suggest workflows not present in the original sequence. More recently, Orbay and Kara [2011] propose a method

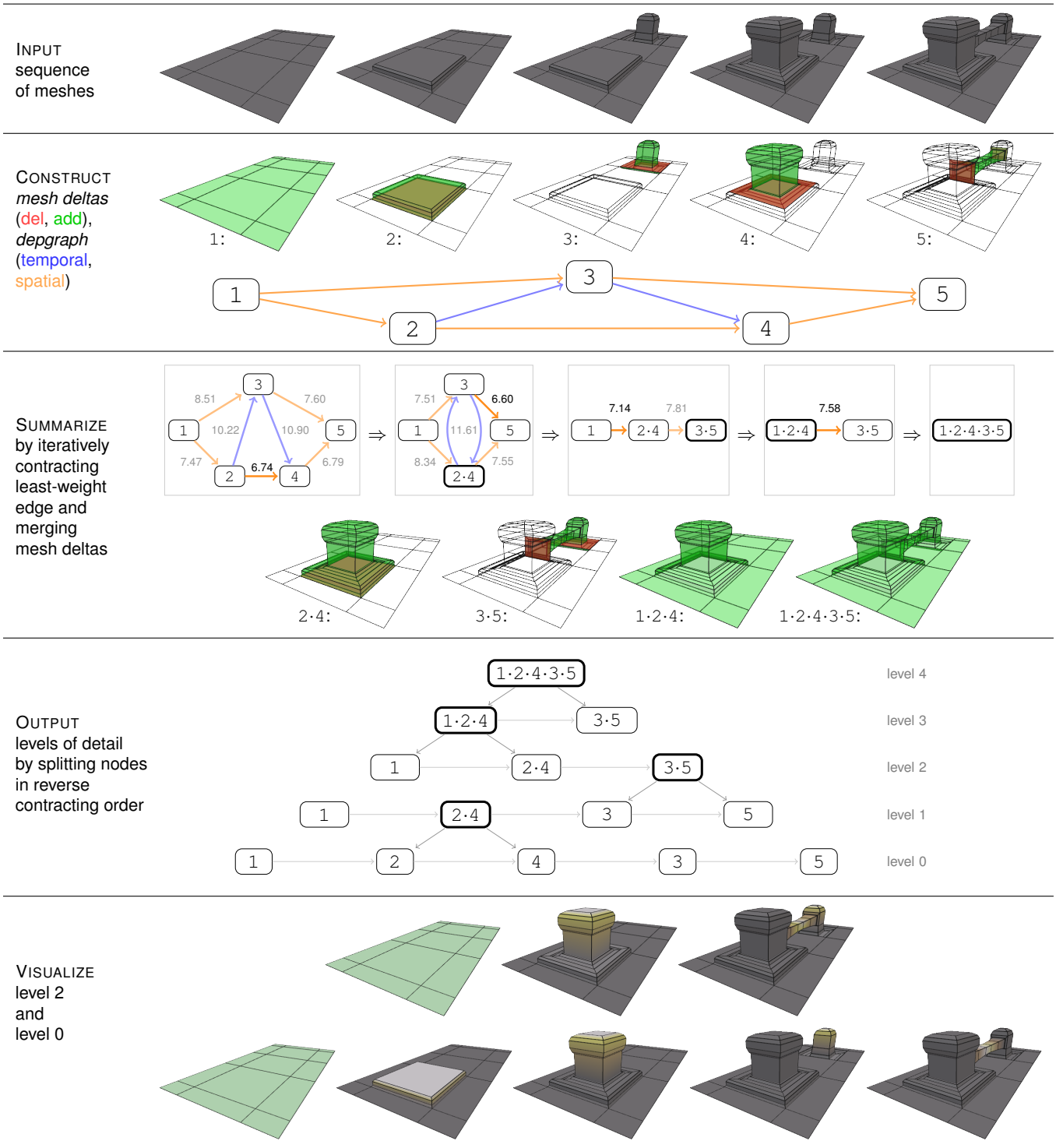


Figure 2: *The input is a sequence of meshes. In this example, each mesh is a single component and was created by performing a series of extrusions.* Mesh deltas *are constructed for each snapshot to find which faces are deleted (red) from and which are added (green) to the previous snapshot. These deltas capture any modification to the mesh, including translating vertices, creating new geometry, and subdividing the mesh. 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. For example, delta 4 deletes a face that is created in delta 2, so the node corresponding to delta 4 is spatially dependent on the node of delta 2. The node of delta 3 is temporally dependent on the node of delta 2, because delta 3 immediately follows delta 2 in the original sequence. 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.*

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 scultping 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. *3DFlow* de-clutters stroke display by providing continuous filtering of strokes based on the strength of the underlying edit.

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. 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 during normal creation workflows or with repository commits.

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.

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

3.1 Constructing *Mesh Deltas*

First we convert the spatially normalized sequence of mesh snapshots into a sequence of mesh differences, which we call *mesh deltas*. We normalize the spatial dimension of the sequence by scaling all the meshes so the union of all bounding boxes fits in a unit cube. 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 every mesh in the original sequence can be perfectly reconstructed by successively applying the sequence of deltas in the same temporal order and then inversely rescaling by 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 simple mesh delta creation rule works surprisingly well. Furthermore, 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.

The changes of two mesh deltas can be *merged* into a single mesh delta. The merged mesh delta is constructed by computing the unions of corresponding sets from the two mesh deltas. Because the former mesh delta can add a face that is deleted by the latter, we subtract from both the union of added faces and the union of deleted faces the faces that are in the intersection of the two deltas. For example in Fig. 2, delta `4` deletes a face that is created in delta `2`. When constructing the merged delta `2·4`, this face is removed from both unions of faces.

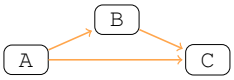
Merging two mesh deltas effectively summarizes in one delta the effects of the two individual mesh deltas. We summarize the sequence into continuous levels of details by iteratively merging mesh deltas.

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 necessary that the deltas are merged in the same temporal order. For example, Fig. 3 shows two summaries of the construction of shark fins. 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 in the top row maintains the original temporal order and therefore contains the single interrupting edit. By temporally reordering the edits so the single, interruptive dorsal fin edit is summarized with the other dorsal fin edits, the bottom summary is much more intuitive and succinct.

While temporally 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` → `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. Although two disconnected meshes are spatially independent, one of the meshes may have influence over the changes of the other. 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.

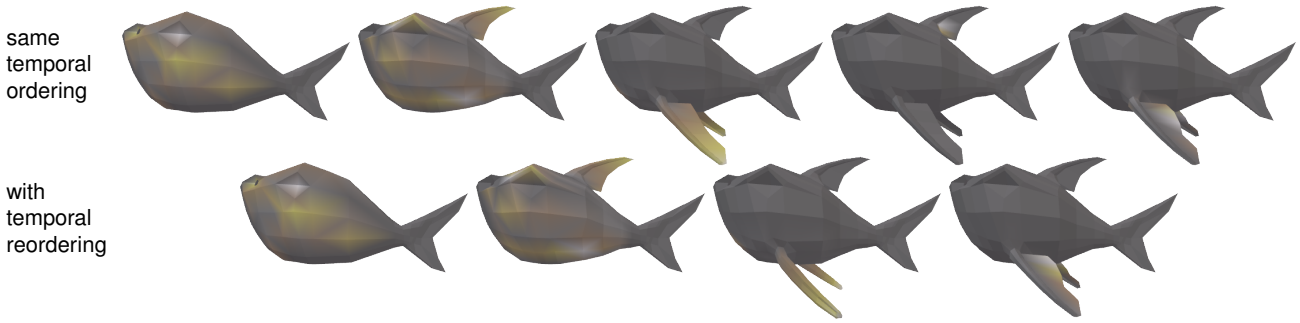


Figure 3: *Temporally reordering edits of shark sequence. The artist first created the dorsal fin and then worked on the pectoral fins. The latter work was interrupted by a single change to dorsal fin. The top row shows a summary of the edits using the same temporal ordering of the original input. The summary shown in bottom row is more succinct, because the sequence is allowed to be temporally reordered so the single edit can be summarized with the other edits to the dorsal fin and not interrupt the pectoral fin edits.*

- A merged delta should not contain edits that are too far apart.

Choosing to merge deltas with strong changes might lose too many details in the summary. Choosing to merge distant deltas may divide the focus of the summary.

From these guidelines, we derive a cost function C for merging a pair of deltas $\texttt{A}$ and $\texttt{B}$ 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:

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

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:

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

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

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

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

where $\Delta_t(\texttt{A})$ is the set of original delta indices covered by delta $\texttt{A}$, $\Delta_x^+(\texttt{A})$ is the set of faces added by $\texttt{A}$, $\Delta_x^-(\texttt{A})$ the set of faces deleted by $\texttt{A}$, $\Delta_x(\texttt{A})$ the set of faces either added or deleted by $\texttt{A}$, 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 $\texttt{A}$ and $\texttt{B}$. The temporal distance term, D_t, is defined as the minimum temporal distance between the $\texttt{A}$ and $\texttt{B}$. This term is computed as the minimum absolute difference between all snapshot indices of $\texttt{A}$ and of $\texttt{B}$ minus one. For example, if delta $\texttt{A}$ covers snapshot 1 and $\texttt{B}$ covers snapshots 2 and 4, the temporal distance cost of merging $\texttt{A}$ and $\texttt{B}$ 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 $\texttt{A}$ and $\texttt{B}$ 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, D_x, is the minimum Euclidean distance between the added and deleted faces of $\texttt{A}$ and the added and deleted faces of $\texttt{B}$. 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, including the polygonal modeling workflows of MeshFlow and MeshGit. 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, to help in presenting the most intuitive summaries to the viewer at every level of detail, the initial mesh (e.g., cube, bust, etc.) of the sculpting workflows is held out from being merged until only two nodes remain.

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 encoded 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, similar to the levels produced by MeshFlow.

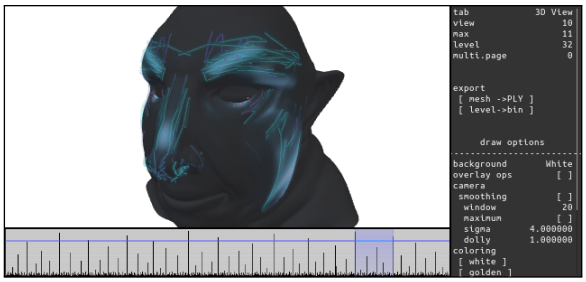


Figure 4: *User interface for* 3DFlow. *The mesh is shown at the top-left for the selected delta and level of detail with surface changes highlighted and sculpting stroke annotations visualized. The timeline at the bottom-left visualizes the deltas at different levels of detail, from every original delta (bottom) to the highest summary (top). The blue highlight indicates the selected level of detail, selected delta, and the deltas of lower levels of detail that are covered by the selected delta. Visualization settings are shown on the right.*

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.

Unlike MeshFlow, 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 clustering patterns 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 where 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. 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 dep-graph 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.

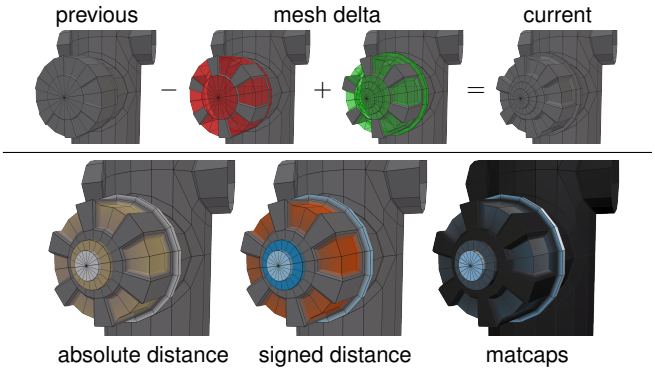


Figure 5: *Emphasizing surface changes in mesh delta. Applying the mesh delta (top-middle) to the previous mesh (top-left) results in the current mesh (top-right). The mesh delta covers 31 deltas in the original sequence. The bottom row shows three different ways to highlight and emphasize the magnitude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

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. 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.

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 new disconnected geometry to the mesh. 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.

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

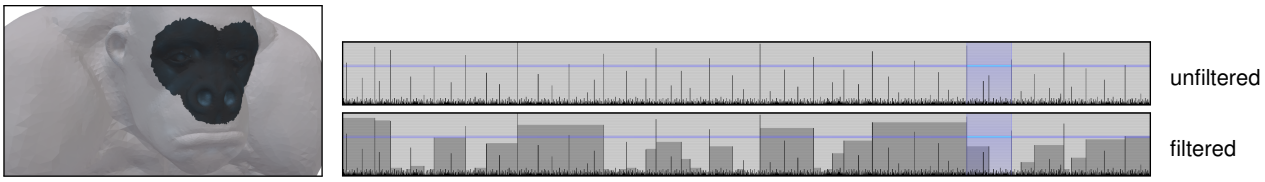


Figure 6: *Spatial filtering on gorilla sequence. The mesh on the left is partially deemphasized to indicate the selected regions. The timelines on the right show without (top) and with (bottom) filtering. The deltas that do not modify the selected region are darkened and are not viewable.*

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 mesh, such as the contour and overall shape or the high-frequency details and creases.

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 is made unviewable and is darkened in the timeline, indicating to the viewer when the selected region was modified. Figure 6 show the timeline filtered to the deltas that modify the face of the gorilla.

Visualizing Sculpting Annotations. While highlighting indicates how much regions of the mesh have 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 stroke 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 but without adding too much clutter. 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 trivially added in *3DFlow*.

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. Similarly 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 performing 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.

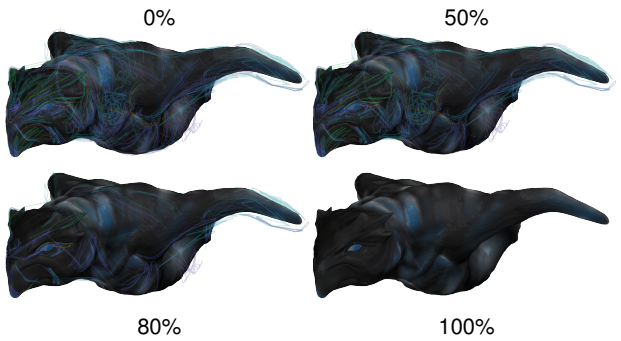


Figure 7: *Filtering annotations at 0%, 50%, 80%, and 100%. The mesh is heavily obscured when visualizing the sculpting stroke annotations of all 343 merged deltas (top-left). With the annotations sorted by a computed weight of change,* 3DFlow *provides continuous filtering to show anywhere from all annotations (0%) to none (100%).*

Other Visualization Options. We refer the reader to the supplemental material for a demonstration of other visualization options. These include: render the summarized workflow using external software; render with a mirror effect to see edits on front- and back-side of mesh at the same time; smoothly interpolate or warp the surface to simulate the artist's summarized work; and center-on and zoom-into the region of the mesh that are edited.

5 Results

In this section we report about the input workflows and briefly discuss the results.

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 with different working styles. One artist has a stronger tendency to explore while editing, making strong changes often throughout the sequence. The other artist prefers a more structured blocking followed by refinement approach. Both artists sculpted using both subdivision 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. *3DFlow* was able to summarize well all sculpting scenarios for both artists, essentially adapting to different workflow styles.

The sculpting artists used an instrumented version of Blender that

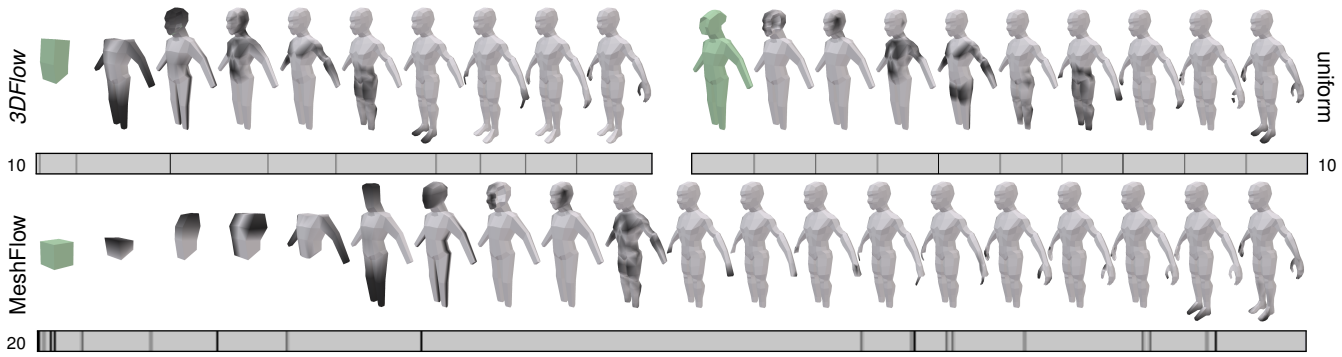


Figure 8: *Comparing summaries produced by* 3DFlow *(top-left), uniform intervals (top-right), and MeshFlow (bottom). Changes are highlighted in black, and the timelines show the coverage of deltas for each summary. While MeshFlow can only summarize the biped sequence down to 20 steps,* 3DFlow *and uniform intervals can provide continuous summarization (10). See Sec. 5 for detailed analysis of this figure.*

	model	fig.	mesh deltas	added faces	record type	process time
subdivision sculpting	*ogre*	4	1459	1,660,475	i	1:26
	merman	9	2218	2,171,310	i	3:40
	sage	9	1686	1,961,133	i	2:19
	engineer	9	863	2,919,865	i	2:54
	elder		2958	1,500,632	i	3:01
	alien	1	2118	6,094,173	i	8:49
	man	9	1459	1,953,859	i	3:03
	fighter	9	1532	1,156,686	i	2:06
dynamic sculpting	*gargoyle*	7	819	1,090,882	i	0:33
	monster	1	797	1,389,906	i	0:47
	elf		4125	4,791,845	i	2:46
	gorilla	1	2482	4,241,528	i	3:32
	explorer		1699	3,416,354	i	2:21
polygonal modeling	*helmet*	1	1321	17,579	i	0:05
	hydrant	5	691	49,892	i	0:04
	robot		1810	139,527	i	0:15
	shark	3	1457	19,177	i	0:06
	biped	8	1267	18,162	i	0:05
	durano	1	11	7,165	c	0:01
	creature		123	280,338	c	0:14
	sintel		210	2,948,611	c	2:11

Table 1: *Statistics of input workflows. The first eight workflows are digital sculpting sessions that used subdivision surface rules to generate higher resolution meshes. The middle five workflows are sculpting sessions that used dynamic remeshing techniques. The last eight workflows were constructed using polygonal modeling techniques. The added faces column reports the number of unique faces added by the original deltas. The record type column reports whether the workflow was created using instrumented software (i) or by committing versions (c). The final column indicates how much processing time (mm:ss) was needed to summarize the workflow. All meshes are shown in supplemental materials.*

saves a copy of the mesh along with any associated tool usage information after each change. The summarization process is performed off-line in order to keep the mesh editing interface fluid for the artists.

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. The *3DFlow* workflows for durano, creature, and sintel were created directly from the committed files without processing or manual filtering.

Discussion. We compare results of summarizing the biped workflow using *3DFlow*, uniform intervals (similar to a time-lapse), 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 blocked-out figure, face, upper body, lower body, feet, and hands. 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 9 show 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 cube with subdivision; see Fig. 1), fighter, and man workflows.

We asked the professional artists who authored the sculpting workflows to provide feedback on the results of *3DFlow*. They found the summarizations captured their workflows and the workflows of the other authors quite well, and both agreed that *3DFlow*'s interactive viewer with summarized workflow is a significant improvement over time-lapsed videos. One artist commented, "I've recently finished working on the materials for a sculpting

course I'm teaching. Having 3DFlow available would have made it unnecessary to share both the final sculpture and the videos of the process, allowing 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.

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 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. All source code and data is released as open source.

7 Acknowledgements

This section is left blank for the review process.

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.

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.

PIXOLOGIC, 2013. ZBrush. http://www.pixologic.com/zbrush.

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.

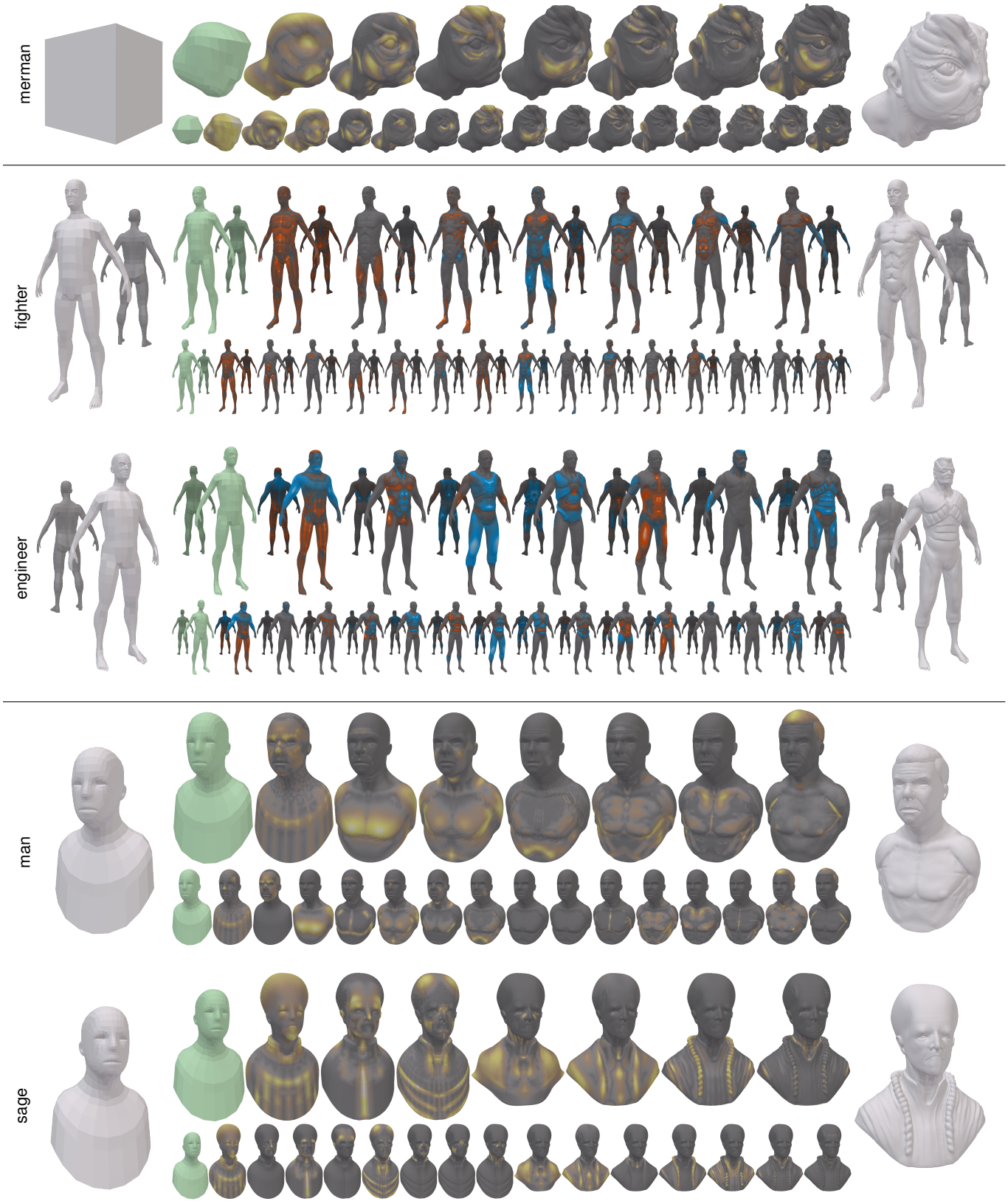


Figure 9: *Five sculpting workflows summarized in 8 and 16 steps. These workflows started with a base mesh (left column) and used subdivision remeshing. The initial and final meshes (right column) are shown without highlighting. The fighter and engineer workflows are visualized with a mirror effect to show both sides of the mesh.*