SculptFlow: Visualizing Sculpting Sequences by Continuous Summarization

Jonathan D. Denning* Fabio Pellacini† Jiawei Ou*
*Dartmouth College †Sapienza University of Rome

June 2014, Dartmouth Computer Science Technical Report TR2014-759

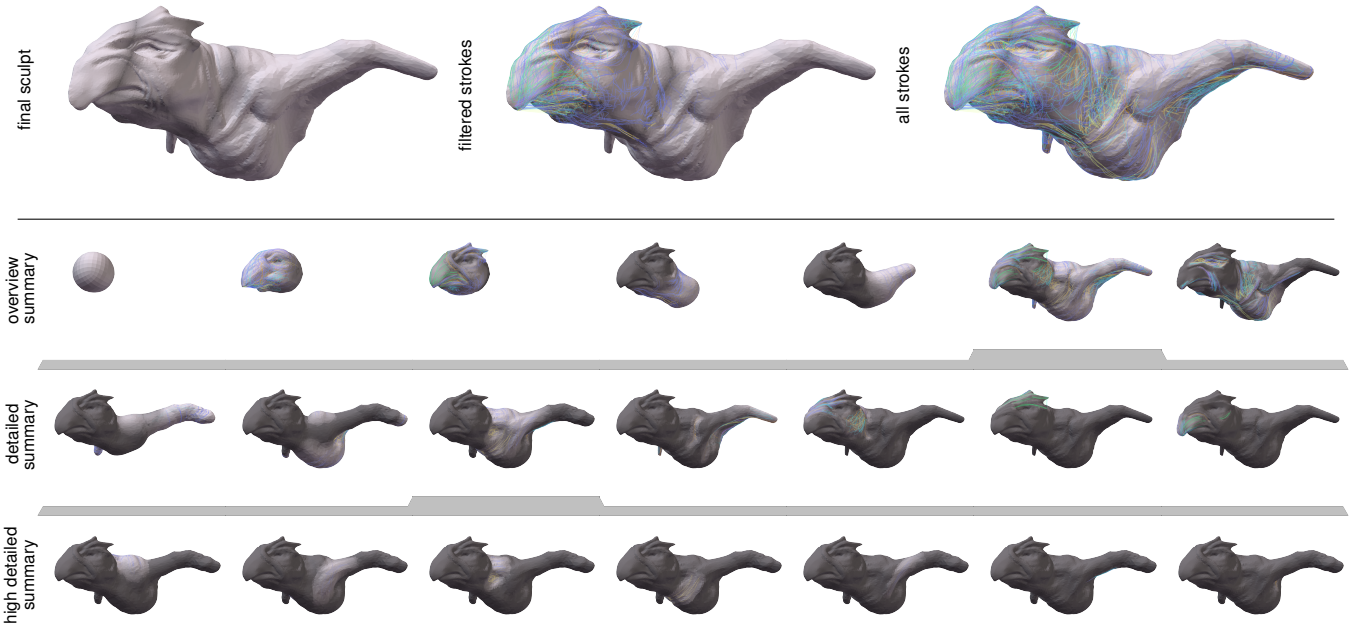

Figure 1: *Three different levels of summary automatically constructed from a digital sculpting session of a professional artist. The top row summarizes all 804 original editing steps with 7 meshes. For this session, the artist started with a primitive shape (top-left subfigure) and sculpted a winged creature (top-right) using digital sculpting software. The sequence of meshes are brightened to indicate the strength of the change, and the artist's brush strokes are overlayed. The lines between each row indicate the how the sequence is clustered.*

Abstract

Digital sculpting is becoming ubiquitous for modeling organic shapes like characters. Artists commonly show their sculpting sessions by producing timelapses or speedup videos. But the long length of these sessions make these visualizations either too long to remain interesting or too fast to be useful. In this paper, we present *SculptFlow*, an algorithm that summarizes sculpted mesh sequences by repeatedly merging pairs of subsequent edits taking into account the number of summarized strokes, the magnitude of the edits, and whether they overlap. Summaries of any length are generated by stopping the merging process when the desired length is reached. We enhance the summaries by highlighting edited regions and drawing filtered strokes to indicate artists' workflows. We tested *SculptFlow* by recording professional artists as they modeled a variety of meshes, from detailed heads to full bodies. When compared to speedup videos, we believe that *SculptFlow* produces more succinct and informative visualizations. We open source *SculptFlow* for artists to show their work and release all our datasets so that others can improve upon our work.

1 Introduction

As the capabilities of content creation software become more complex, it is increasingly important to understand how to view them effectively. Skilled artists, practiced at using the software available, can create intricate and diverse results, often in long editing sessions. Given the complexity of these edits, it is hard for other users to learn a particular workflow, and for researchers and engineers to understand how to improve upon those workflows. Motivated by these goals, recent research has explored ways to visualize and navigate complex and lengthy recordings of artists at work. For example, *VisTrails* [VisTrails 2010] helps in navigating non-linear undo histories in 3D software, while Chen et al. [Chen et al. 2011] allows 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.

Digital Sculpting In this paper, we focus on visualizing digital sculpting sequences. In sculpting, artists alter the shape of a mesh as they were sculpting a clay model 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 suite for modeling organic shapes like characters, with packages like ZBrush and MudBox becoming ubiquitous.

There are two major working phases with digital sculpting: *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. 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 in terms of their strength.

Even for relatively low detail models, sculpting sequences are long. For example, the meshes in this paper are a few thousand

strokes, on average. Artists illustrate their workflows by creating either timelapses or sped up videos. This is so common that ZBrush [Pixologic 2013] has a feature just for this. But the length of sculpting sessions make these visualization either too long to remain interesting or too fast to be useful.

SculptFlow. In this paper, we present *SculptFlow*, an algorithm to summarize sculpting sequences. Figure 1 show the summary of an example sequence at different levels of detail. *SculptFlow* 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 sculpting sequence. As in MeshFlow [Denning et al. 2011], we add visual annotations to highlight important changes and summarize the artist's brushing.

SculptFlow takes as input a mesh sequence with brush strokes data and outputs a summarized mesh sequence with visual annotations. To summarize the sequence, we repeatedly merge pairs of subsequent edits until only one is left. Each merge shortens the sequence by one, allowing any summary length to be generated. At each merge, we greedily choose the pair based on a cost function that accounts for the number of summarized strokes, the magnitude of the change, and whether the edits overlap. We highlight parts of the mesh that change to ensure that small edits are easily visible during playback. We summarize the artists' strokes by providing continuous filtering of the original strokes based on the edit magnitude.

We tested *SculptFlow* by recording professional artists with a lightweight software instrumentation. The artists modeled a variety of organic models, from detailed heads to full bodies, with different workflows based on their personal preference. Sequence length varied from several hundreds to a few thousand strokes. We tested sculpting on tessellated subdivision surfaces as well as meshes dynamically refined during editing. We found that *SculptFlow* worked well across all the datasets tested. We refer the reader to the supplemental video for a comparison between *SculptFlow* summaries and the fast-forwarded original sequence. We release all sculpting data as well as code for both *SculptFlow* 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 Summaries. As software packages for image and 3D scene creation become more complicated, both developers and users benefit from understanding common usage patterns. 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 a corpus of workflows to users at three levels of granularity to study how users compare workflows and which granularity is most preferred. Software users learn from other artists' experience by studying their workflows thought 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 will present command recommendations to the user based on the 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. 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.

Polygonal Modeling Summaries. 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 with color and drawing symbols like arrows 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. We take inspiration from MeshFlow by highlighting the individual edits that have been clustered. More specifically, we highlight sculpted regions where brightness of highlight indicate magnitude of change, and we overlay colored lines on the mesh to show the artist's brush strokes.

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 sculpting sequences.

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 [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 organize into patterns that suggest workflows not present in the original sequence. More recently, Orbay and Kara [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 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. In our system, we declutter stroke display by continous filtering of strokes based on the strength of the underlying edit.

3 Clustering Sculpting Sequences

Mesh Deltas The input to *SculptFlow* is a sequence of meshes with brush stroke data. We preprocess the input sequence by converting it into a sequence of mesh differences, which we call *mesh deltas*, in order to have a scalable in-memory representation of the entire edit sequence. A mesh delta is a pair of sets: one for faces that have been deleted or changed from the previous mesh, and one for faces that have been added or moved in the current mesh. We detect face changes by checking whether a face has moved beyond an epsilon from the previous mesh. We choose to use two face sets since this works well when adjacency changes over the sequence. This representation is not only scalable in terms of memory, but allows us to quickly perform operations by reducing computation to the delta itself, rather than the whole mesh.

All meshes in the sequence can be reconstructed by successively

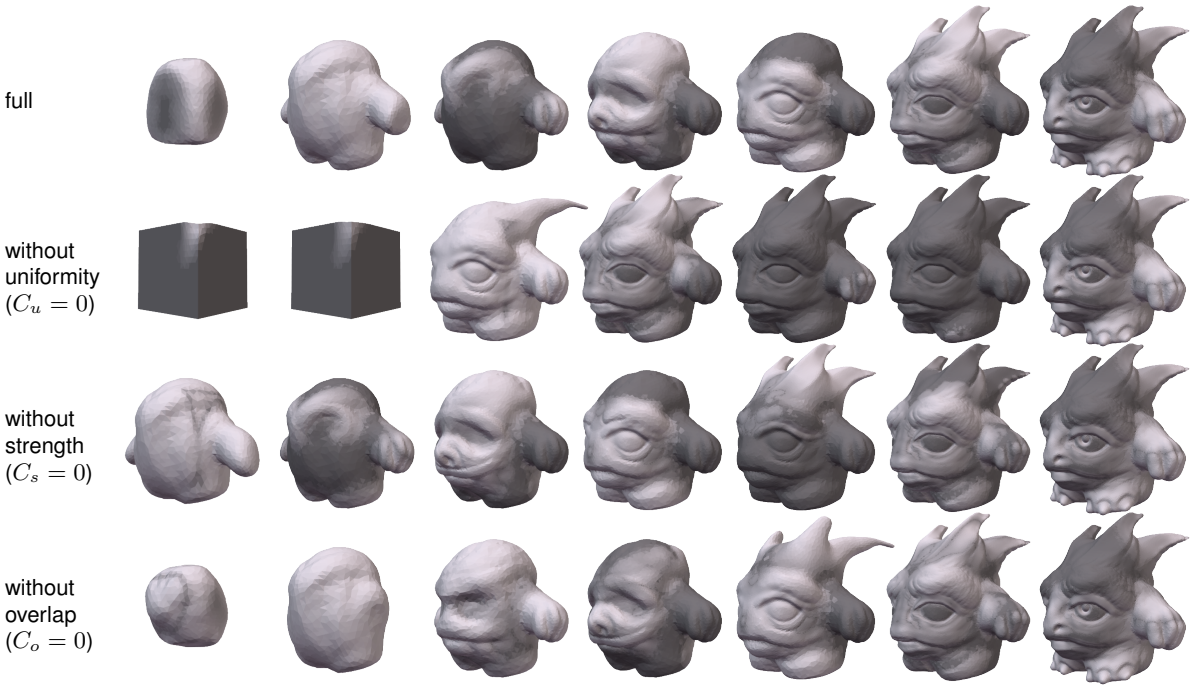

Figure 2: *Effect of removing terms from cost function. The top row shows the clustered edits using the full cost function (Equation 1). The bottom three rows show how clustering is affected by removing the uniformity, strength, and the overlap terms, respectively. In order to show better the strength of change in each cluster, the brush strokes are not visualized.*

applying the sequence of deltas. Two deltas overlap when the intersection of deleted faces in the current delta and the added faces of the previous delta is not empty. Two deltas can be merged by computing the union of deleted faces and the union of added faces, both with the faces in the overlap subtracted.

Uniform Summaries. Uniformly subsampling the sequence, similar to fast-forward playback, gives users a summary view of the sequence. Uniform subsampling is fast to compute and can easily reduce the length of the sequence by taking larger steps. At the same time, it does not adapt to the content of the sculpting sequence in three main manners. First, there may be abrupt discontinuities when transitioning between two levels of summary since the intervals do not align. Second, some intervals may contain large edits, while others may be minor. Lastly, the step sizes could unnaturally break a semantically continuous sequence of edits into separate intervals.

Continuous Summaries. We want to produce summaries that are efficient to compute, like uniform subsampling, but without the limitations. More specifically, we want to efficiently cluster the input sequence to provide continuous levels of summary, while ensuring that each cluster is neither too long, nor responsible for too large edits, and that it respects a semantically continuous sequence of edits. We chose to use continuous clustering since it adapts well to different sequences, does not require determining the number of clusters upfront, and since it has been shown to be effective for image sequences [Barnes et al. 2010]. To compute the clusters, we start with the original sequence and repeatedly merge one pair of subsequent deltas, until only one edit is left. We can think of each merge as creating a new summary, shorter than the previous one by one edit. The user can thus choose any summary length desired. To determine which pair of edits to cluster, we compute the cost of clustering each neighboring pairs of edits in the sequence and then greedily choose the pair with the lowest clustering cost.

Non-Uniform Summaries. In clustering edits, we follow three main guidelines. First, we want to ensure that a cluster does not contain too many edits since the length of the sequence represents the artist's effort and the summary should reflect that. Uniform subsampling captures this well, so we want to account for it similarly. Second, deltas representing semantically smaller edits should be clustered first, to ensure that larger changes are left in the summaries. Third, the clustering algorithm should try to cluster overlapping edits first. This will summarize well sequences of edits like repeated stroking.

From these clustering guidelines, we derive a cost C for clustering neighboring deltas A and B as a sum of three terms to reflect each of the three guidelines. Note that in this notation each delta may be the result of previous merges. Our cost is the weighted sum of three terms:

$$C(A,B) = \underbrace{\frac{l_A - l_B}{l_{avg}}}_{C_u} + \underbrace{\frac{(a_A - d_A) + (a_B - d_B)}{d_A + d_B - o_{AB}}}_{C_s} + \underbrace{\frac{1 - o_{AB}}{2d_B}}_{C_o} \quad (1)$$

where l_A, l_B are the number of original user strokes summarized by A and B respectively, l_{avg} is the average number of original user strokes summarized by the current sequence of mesh deltas, a_A, a_B are the surface areas added by A and B, d_A, d_B are the surface areas of the faces deleted by A and B, and o_{AB} is the surface area of overlapping faces. The term l_{avg} is computed as the original number of mesh deltas divided by the current number of mesh deltas. The overlapping surface area o_{AB} is the surface area of the faces added in A that are then deleted by the following mesh delta B.

The *uniformity cost*, C_u, penalizes the merge of deltas that contain too many user strokes. It is computed as the number of strokes summarized by both deltas divided by the average number of strokes of all deltas. The uniformity cost of clustering two relatively long deltas is greater than that of clustering two relatively short ones. We normalize this term by the average length of deltas

to ensure that it does not overwhelm the other terms in the cost function as the length of the deltas increase. The *strength cost*, C_s, penalizes the merge of deltas that change the mesh significantly. We compute this as the surface area that has changed after applying both deltas (i.e., added surface area minus deleted surface area) normalized by surface area that is deleted. The overlapping area is subtracted to prevent double counting. We normalize the change by the area of the region to capture both blocking and refinement edits that work at different scales. The *overlap cost*, C_o, penalizes the merge of deltas with little or no overlap. It is computed as the surface area of the overlap between the deltas normalized by the surface area deleted by second delta.

Figure 2 demonstrates how the clustering changes when one of the terms is removed from the cost function With the uniformity term removed, large or overlapping edits become too dominant, obscuring workflow. When the strength term is removed, large edits done with only a few strokes are lost in the summary, especially during blocking. When the overlap term is removed, semantically continuous edits can be unnaturally split into separate clusters. For example, the front horn is partly sculpted in the fifth subfigure of the bottom row but is fully sculpted in the other rows.

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 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 added computation.

Limitations While we believe that Equation 1 performs well in regards to our criteria, it does not capture the semantic of an edit. For example, it might make sense to cluster together edits that work on eye or those that add wrinkles across the face. The formulation above does not infer any semantical meaning from the brush stroke or from the region being changed. Additionally, we only consider clustering subsequent changes, treating the sculpting sequence linearly. Clustering spatially overlapping edits, regardless of whether they overlap temporally, could produce more intuitive summaries. We leave this for future work.

4 Annotations

Highlighting Changes. While clustering allows us to summarize the sculpting sequence, small edits are easily missed during playback. In *SculptFlow* we highlight regions of the mesh that have been altered by coloring each face lighter according to the magnitude of the change, while leaving unaltered faces as gray. We estimate the magnitude of the change by computing the minimum distance from each face center to the surface of the previous mesh. To reduce the search time, we only consider the faces in the mesh delta.

User Strokes. While highlighting indicates how much regions of the mesh have changed, it does not summarize how the artist performed such a change. In *SculptFlow* we visualized the artist's strokes directly on the mesh. Strokes are colored by brush type: pulling in blue, smoothing in cyan, creasing in orange, and grabbing or nudging in pink. To highlight repeated strokes, we draw densely packed strokes thin and opaque, to increase their visibility, while sparse strokes are drawn thick and mostly transparent. When strong clustering is performed, user strokes do not necessarily lay in the current mesh surface since they were applied on significantly different meshes that are now in the same cluster. We draw strokes projected to the nearest point of the visualized mesh, rather than using their original position, to avoid misperceived stroke placement.

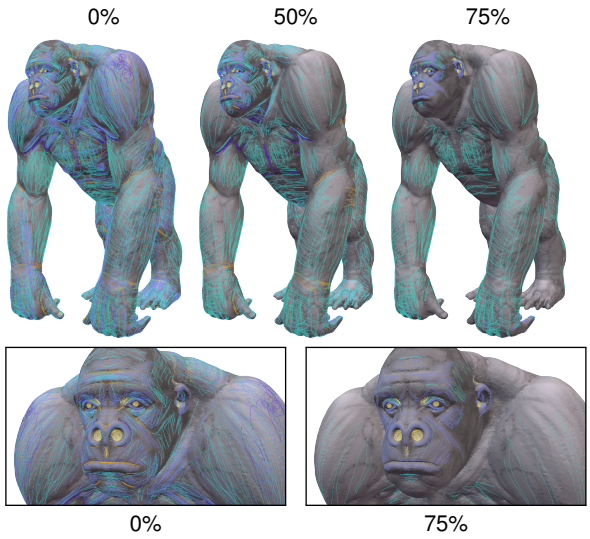

Figure 3: *Setting stroke filtering level at 0% (left), 50% (middle), and 75% (right). This clustered edit contains 621 strokes. When all of the strokes are overlayed (left), the view of the mesh is obscured by the strokes. By filtering the strokes that change the mesh the least, the summary shows the mesh more clearly without compromising the major stroke information.*

Filtering Strokes. As edits are clustered together, the number of strokes quickly increases, obscuring the view of the mesh. *SculptFlow* provides continuous levels of summary for strokes through filtering. Filtering removes the strokes that changed the mesh the least. The filtering can be continuously adjusted to show any number of strokes from all to only one. Continuous stroke filtering is conceptually similar to the continuous zooming implemented for mesh changes, and in our case implemented similarly. When clustering mesh changes, strokes corresponding to edits that have the greatest strength cost are preferred over the strokes of the other edit. For each mesh delta, this results in a sorted list of strokes prioritizing the strongest ones. See Fig. 3 for an example of different levels of stroke filtering.

Discussion. We considered two alternatives to stroke filtering: determine a representative through spatial clustering or performing edge-bundling [Holten and Van Wijk 2009] to reduce clutter. We found though that these alternatives were of little help for uncorrelated strokes, or suggested stroke patterns that were not representative of the artist's workflow in the case of close-by sets of correlated strokes.

5 Results

We tested *SculptFlow* on a variety of sculpting sequences, shown throughout the paper and in supplemental videos. Source code and dataset are available in supplemental material.*

Implementation We obtain sculpting sequences by a lightweight instrumentation of Blender. In our instrumentation, we save a copy of the mesh after each change, which includes brush strokes, undos, and global operations such as transformations. We compute

*Due to space limitations during submission, only a subset of the meshes is currently included.

Model	Fig.	Edits	Resolution Type	Cluster Time	Annotate Time
ogre	4	1459	subd.	4s	127s
merman	4	2245	subd.	8s	240s
gargoyle	1	804	dyn. tess.	2s	62s
sage	4	1642	subd.	6s	1428s
monster	2	797	dyn. tess.	2s	100s
engineer	4	844	subd.	5s	440s
elder		2709	subd.	11s	196s
alien		2143	subd.	17s	1228s
elf		4125	dyn. tess.	17s	263s
gorilla	4	2481	dyn. tess.	8s	313s
man		1413	subd.	6s	451s
explorer	4	1697	dyn. tess.	5s	205s
fighter		1522	subd.	4s	300s

Table 1: *Statistics of input sculpting sessions. Eight of the sculpting sessions used subdivision surface rules (subd.) to generate higher resolution meshes, while the other five sessions used dynamic remeshing (dyn. tess.) techniques. Time to cluster the sequence and compute annotations are given in the last two columns. All meshes are shown in supplemental video.*

the mesh deltas as as offline post-process in order to keep the interface fluid for the artists.

For fast visualization and to keep the user interface simple, we cache snapshots of the meshes with visual annotations into discrete levels of zooming, where each level summarizes the level below and has half as many edits. When the user changes the level of summary to display, we interpolate the position of the meshes being clustered to the summarized edit position. This provides feedback to the user on how the edits are clustered.

Mesh sequences. Table 1 summarizes statistics for the tested sequences. We consider sequences that use either subdivision or dynamic tessellation to control resolution. Sequence length in terms of brush strokes varies from several hundreds to a few thousands. We tested both detailed heads and more complete full-body sculpting. We include more sequences in supplemental material. *SculptFlow* worked well in all these cases.

Our data was obtained by two professional artists with different working style. One has a stronger tendency to explore while editing, making strong changes often throughout the sequence. The other prefers a more structure blocking followed by refinement approach. *SculptFlow* was able to summarize both sequences with ease, essentially adapting to different workflow styles.

Performance. We collect performance statistics by running our implementation on a quad-core 2.93GHz Intel Core i7 with 16GB RAM and an ATI Radeon HD 5750 graphics card. Clustering the data is performed very quickly, taking at most 17 seconds for the sequence with the most number of edits. In our implementation, the dominant computation is the magnitude of change for the visualizations.

6 Conclusion and Future Work

We presented *SculptFlow*, a system for visualizing sculpting sequences. *SculptFlow* allows viewers to continously zoom in the detail of sculpting sequences. The summarized sequence is computed by greedily merging changes with a metric that considers the uniformity, strength, and overlap of the changes. For each change, we highlight edited regions by computing the magnitude of the change. For summarizing artist strokes, we filter the strokes of changes with the smallest impact.

In the future, we are interested in extending *SculptFlow* to other data types, for example painting textures and materials onto surfaces. Eventually, we would like to develop a scalable method for whole-scene edit visualization capable of handling sequences of edits for full scenes and all types of data.

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.

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.

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.

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.

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.

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.

VISTRAILS, 2010. VisTrails Provenance Explorer for Maya. www.vistrails.com/maya.html.

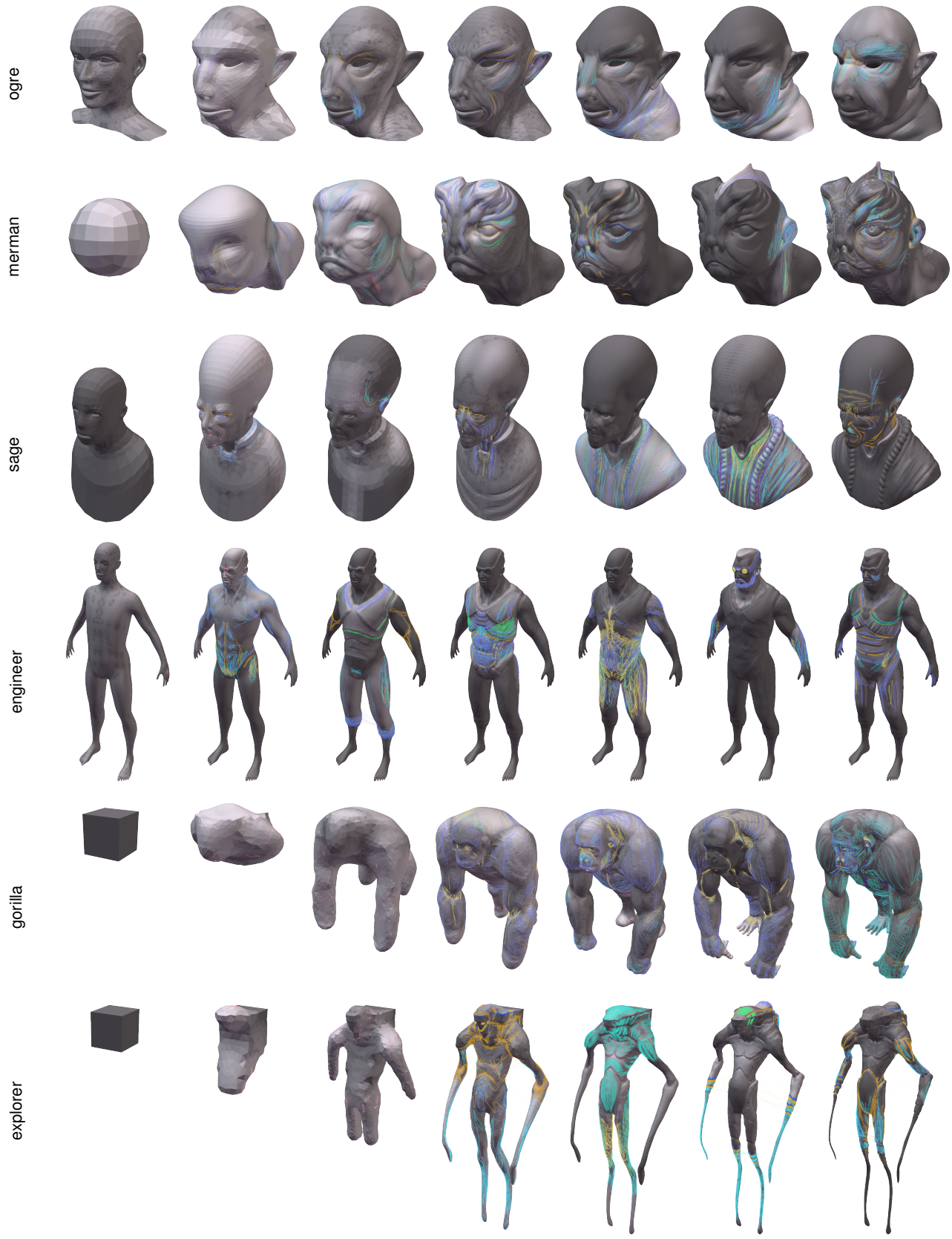

Figure 4: *Several sculpting sessions summarized in seven meshes. The leftmost subfigure is the initial mesh; the rightmost is the final mesh. The middle subfigures are automatically generated from* SculptFlow *and drawn with visual annotations indicating strength of change and the artist's brush strokes. The strokes are filtered at 50%. We refer the reader to supplemental videos for high-resolution playback of all sculpting sequences.*