CrossComp: Comparing Multiple Artists Performing Similar Modeling Tasks

Jonathan D. Denning
Dartmouth College

Fabio Pellacini
Sapienza University of Rome

June 2014, Dartmouth Computer Science Technical Report TR2014-760

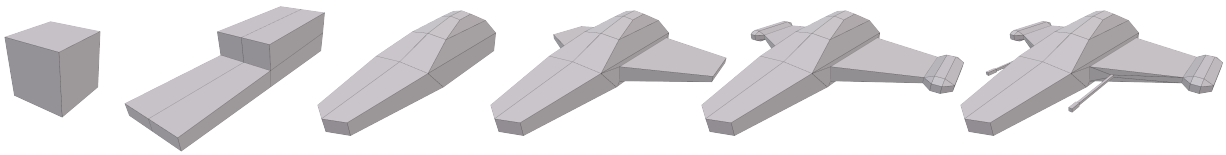

Figure 1: *A subset of snapshots from Scout sequence by Author.*

1 Abstract

In two previous papers, we have focused on summarizing and visualizing the edits of a single workflow and visualizing and merging the edits of two independent workflows. In this paper, we focus on visualizing the similarities and dissimilarities of many workflows where digital artists perform similar tasks. The tasks have been chosen so each artist starts and ends with a common state. We show how to leverage the previous work to produce a visualization tool that allows for easy scanning through the workflows.

2 Introduction

Let us consider the following scenario as a motivating example. Suppose that a digital arts instructor assigns to the students the task of creating a particular 3D model. The assignment could be used to assess the students' ability or technique or to teach the student a new technique. For the former use-case, the instructor might choose to give to the students a target model to recreate. For the latter, the instructor might present the instructions in the form of a video tutorial. The scenario illustrated above is a common practice especially for web-based mentoring, such as with CG Cookie.

When the assigned task involves many components, the instructor may ask the students to periodically save a snapshot of their model as they work and then submit their workflow as a work-in-progress sequence. When the task is a single piece, the students may only report the final state of their model.

In [Denning et al. 2011] and [Denning and Pellacini 2014], we demonstrated two systems that summarize and visualize a single artist working on a single task. Clearly, the instructor could use one of these tools to review the workflow of each student. These tools and techniques, however, do not help with determining how closely the student followed the tutorial, with identifying effective or efficient workflow patterns, or with finding poor techniques or common modeling problems.

In this paper, we present CrossComp, a system designed to help compare multiple artists performing similar mesh editing tasks. We focus on task-based polygonal modeling workflows, where the start and ending conditions are highly defined but the workflows from start to finish may differ. We demonstrate CrossComp by analyzing four subjects performing four tasks, where three tasks use a video tutorial and the fourth uses a target 3D model. We remark on some observations on the workflows that are clear in CrossComp but might have been missed with manual inspection or with inspecting only snapshots. Finally, we conclude with reporting on open-ended feedback from a professional digital artist and instructor and with discussing limitations and future research directions for this work.

3 Related Work

The works by Kong et al. [Kong et al. 2012] and Pavel et al. [Pavel et al. 2013] are closely related to the work presented in this paper. The goal of their work is to help users identify the trade-offs between many possible workflows that perform the same image-editing task, such as "Find Edges" or "Sketch Effect". They present and evaluate different workflow visualizations for displaying and comparing image-editing workflows. One visualization, called *union graph*, compares two sequences of commands by showing each sequence as a directed graph with a node for each operation and directed edges to indicate temporal order. The similarity and dissimilarity is indicated by overlapping nodes of the two graphs if the corresponding operations are sufficiently similar in terms of operation name or type and parameter settings. Another, called *alignment view*, compares two or more sequences of commands by arranging the workflows according to similarity in operation usage. The operations of each workflow is drawn as a list, and edges are drawn between neighboring operations that are similar.

While their data included short, highly-polished tasks scraped from photo-editing tutorials, our work focuses on much longer workflows that can contain errors and undone work. Furthermore, although they provided step-by-step visualizations of the workflow allowing for manual inspection and comparison, their automated methods rely solely on the operation type and parameter settings. Typically mesh editing software has far fewer number of operations that can be performed with many operations able to perform several different types of manipulations. In other words, when compared to image editing workflows, the differences in mesh editing workflows depend more on the effect of the operation or the combination of operations than the actual operation name, parameter setting, or order of operations.

Lafrenier et al. [Lafreniere et al. 2013] describe a system, FollowUs, where a user can view a tutorial submitted by the original author or by other users performing their version of the tutorial. Matejka et al. [Matejka et al. 2009] describe a recommender system, CommunityCommands, that collects usage data from a user community and then displays to each user a set of commands the user may not be familiar with. These two systems enhance a user's understanding of the tutorial or software system by presenting how other users of the community perform the task or use the software. The focus of our work is to provide the user a tool to compare the workflows of the community, not just to review.

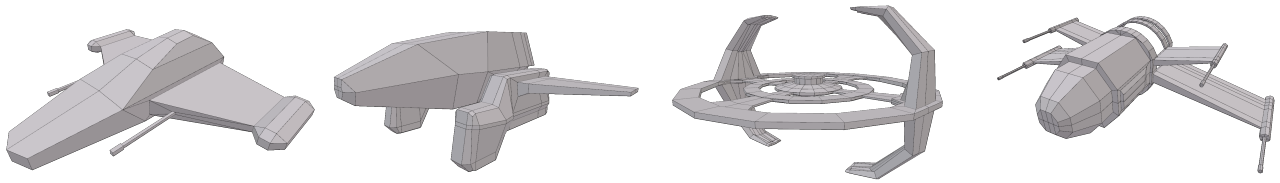

Figure 2: *Final meshes for each task. The scout, transporter, and station tasks were presented as video tutorials, and the subjects were asked to follow the tutorial. The interceptor task was presented as a final mesh, and the subjects were asked to recreate the mesh as closely as possible by using any modeling technique.*

4 Data Collection

Our experiments consisted of four relatively short tasks, involving roughly 20 to 60 minutes of modeling, of moderately increasing difficulty. The first three tasks we presented to the subject in video tutorial format, and the final task was given as a target model. We asked the subjects to follow as closely as possible the steps in the three video tutorials and to recreate as precisely as they could the target mesh of the fourth task. For the final task, the subject could use any modeling technique to replicate the model.

Although all of the subjects reported having some modeling experience, some did not have any experience using the chosen modeling software prior to starting the experiment. Therefore, we designed the video tutorials to provide software usage instruction in addition to high-level explanations of the mesh construction via an overlaid audio track. The video of each tutorial is a screen-capture time-lapse of the construction played back in an interactive video player at real time with a few pauses to point out features. The mesh of the final task is viewed within an interactive 3D viewer to allow the subject to inspect and interact with the mesh.

We chose for all four tasks the theme of spaceships. Although these goal-based tasks would limit the exploration and variability of the workflows, we felt that open-ended tasks or goals that were open to interpretation would inject a subjectivity and aesthetic component into the workflow that would make objective analysis significantly more difficult. Figure 2 shows the final mesh for each of the four tasks.

We used an instrumented version of Blender to record the workflows, both for the author and for the subjects. The starting condition for all tasks contains a single unit cube. Every action that modified the state of the modeling software was recorded, including the undoing of work. The entire recording system for the subject was a self-contained executable with a simple interface, which simplified the process for the subject, and allowed the subject to work at their own pace.

Four subjects participated in the study, but one subject did not submit one of the tasks. See Table 1 for statistics on the recorded workflows.

Model	Type	Author	Sbj. 1	Sbj. 2	Sbj. 3	Sbj. 4
scout	vid	100	125	298	144	217
transporter	vid	171	197	238	164	311
station	vid	244	—	160	306	377
interceptor	mesh	195	507	283	230	465

Table 1: *Statistics for workflow comparison data. The numerical values indicate number of mesh changing edits (no selections, view changes, etc.). The author created the video tutorials (scout, transporter, station) and mesh target goal (interceptor) that the other subjects followed and tried to reproduce. Note: Subject 1 did not finish the station task.*

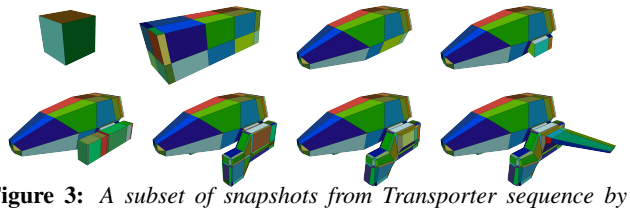

Figure 3: *A subset of snapshots from Transporter sequence by Author. Corresponding faces are colored similarly.*

5 Correspondence and Distance

CrossComp takes as input the recorded snapshots of the corresponding workflows. In order to compare, CrossComp must build an intra-correspondence of elements along each workflow and an inter-correspondence between the workflows. The intra-correspondences is constructed similarly to MeshFlow, where each face is uniquely labeled (locally) upon creation and tracked throughout the workflow.

While we cannot make any assumptions about the state of the mesh in the middle of the workflow, because the mesh can be arbitrarily changed, we can assume that the beginning and ending states of two workflows are similar to known states. As the beginning state for each task is a unit cube and therefore not very informative in terms of inter-correspondences, we use the final state of each workflow to build inter-correspondences. We use a slightly modified MeshGit* to build inter-correspondences between the ending state of the meshes and to uniquely label (globally) the faces. See Figure 4 for results of building inter-correspondences.

Snapshot Edit Distance. One way to compare two meshes to find how similar or dissimilar they are is to compute an edit distance between the pair. The edit distance between two meshes is defined as the minimal amount of change required to turn one mesh into the other. If the edit distance is small, then the two meshes are quite similar; if the distance is large, then the two meshes are quite dissimilar.

In [Denning and Pellacini 2013] we defined a mesh edit distance which we used to build a correspondence between meshes.

*The MeshGit modifications include: the the dot product of the elements' normals in geometric cost are made absolute, and the greedy step is performed one additional time at end without removing twisted faces or faces with mismatched adjacencies. The first modification accounts for flipped normals, and the second modification allows MeshGit to match as many faces as possible by ignoring mismatched adjacencies.

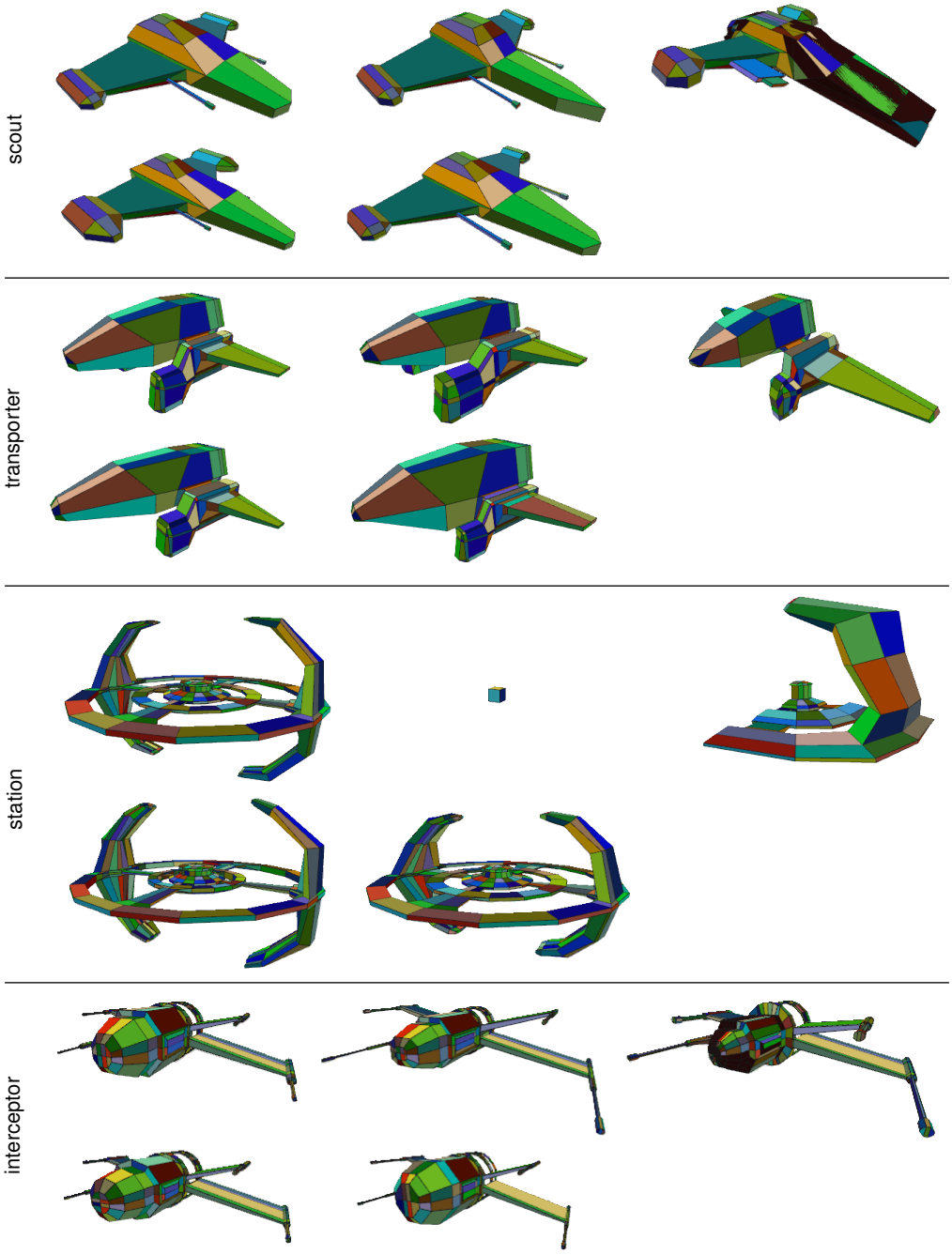

Figure 4: *Final meshes for each of the tasks with inter-correspondences illustrated by matching face colors. The top-left subfigure for each workflow was constructed by Author, and all other subfigures are for the modeling subjects. The faces of top-left subfigure are randomly colored, and the faces for other workflows are colored to indicate inter-correspondences. If the face does not have an inter-correspondence, it is colored dark red.*

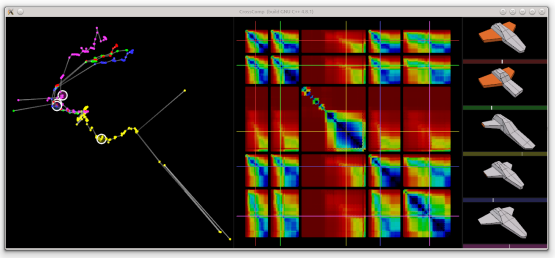

Figure 5: *Basic user interface. The left column visualizes the snapshots of workflows in low-dimensional space. The center column shows a pairwise edit distance heat map. The right column contains interactive views of each workflow.*

CrossComp uses a modified version of the mesh edit distance[†] along with the already established intra-correspondences and inter-correspondences to compute an edit distance between any two pairs of snapshots.

6 Visualization

The basic user interface for CrossComp is shown in Figure 5. The left column shows a 3D embedding of the snapshots after performing a nonlinear dimensionality reduction of the pairwise edit distances, the center column visualizes a heat map of the pairwise edit distances, and the right column consists of interactive views of the snapshots for each workflow. While each column visualizes different features of the workflows, they are synced over the time dimension for each workflow. This syncing means, for example, that adjusting the current time of a workflow in one column will automatically update the corresponding visualizations in the other columns. The first column indicates currently viewed time with a white circle; the second with horizontal and vertical lines; the third with white ticks on the colored bars below the model. Each workflow has an associated color (red, green, yellow, blue, purple, resp.). Changes to the mesh are indicated in the third column by coloring the modified faces orange.

In all of the figures, the original tutorial author workflow is the first workflow (red), and the subjects' workflows are compared to the author.

Edit Distance Coordinates. The left side of Figure 6 shows a 3D embedding of the Scout workflows according to their pairwise snapshot edit distance. Each snapshot of the workflow is indicated by a dot, colored corresponding to the workflow. The edges between dots indicate temporal order of edits. We performed a nonlinear dimensionality reduction on the pairwise edit distances by using Isomap [Tenenbaum et al. 2000] with a k-nn search to find the local neighborhood. We used a value of $k = 10$, but forced at least one mesh from each workflow to be included (the mesh with smallest edit distance) so the embedding would take all workflows into account.

The dots corresponding to two similar snapshots will appear close in this space, while the dots of two quite different snapshots will be far apart. Referring back to Figure 6, note the inset figure which zooms into the large cluster of dots near the center of the column. These dots correspond to the early snapshots of the workflows, where the meshes were very similar in shape (the initial

[†]The snapshot edit distance considers only the face elements of MeshGit's mesh edit distance.

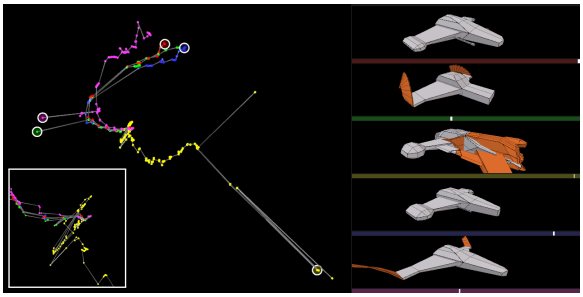

Figure 6: *Outliers in Scout task. Two of the workflows (2,5) used the wrong operation or and one the wrong parameter setting (3), causing a very large change that differed greatly from the other workflows. Inset zooms into the initial state of the workflows.*

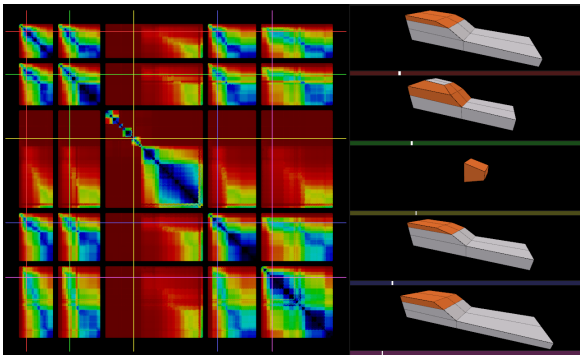

Figure 7: *Heat map of Scout task. The color in the pairwise heat map indicates the edit distance between a pair of mesh snapshots from all workflows of a task. Cool colors (blue, green) indicate similar meshes, i.e., small edit distances. Warm colors (red, yellow) indicate dissimilar meshes, i.e., large edit distances. Note: the heat map is symmetric because we used a symmetric edit distance.*

cube mesh). From these dots, all the workflows except the third (yellow) follow very closely to each other with just a few outliers. The outliers, selected in the figure, were caused by the artists performing an incorrect operation (here, the *spin* operation instead of *subdivide*). The artist quickly corrected the error by undoing the work and then continued following closely the tutorial. The third workflow, however, diverged from the other workflows after performing a large number of incorrect operations, seen as the numerous back and forth edges near the center of the inset. Close to the end of the third workflow, we see some additional outliers where the artist attempts to choose the correct parameter settings for the *mirror modifier*. We discuss this more below.

Edit Distance Heat Map. Figure 7 shows a heat map visualization of the pairwise edit distances of the Scout task workflows. The topmost row and leftmost column of the heatmap correspond to the first workflow, followed by the second workflow moving down and right, etc. The color in the intersection of a specific row and column indicates the edit distance between the mesh snapshots corresponding to the specific row and column. The color is determined by linearly mapping the regularized distance to a color gradient that runs from black to blue, green, red, and dark red, where a black color indicates no edit distance (exactly the same mesh), and a dark red color indicates a large distance (very different meshes). Extra space is added between rows and columns to distinguish the workflows. The horizontal and vertical lines running across the heat map indicate the currently viewed time for the corresponding

workflow. We regularize the edit distances by dividing by the total number of faces. We found that edit distance regularization helps filter accumulated change and generates more intuitive heat maps.

One observation to note about the figure is the wide band of dark red rows and columns in the early parts (top-left corner) of the third workflow, where the artist made and corrected several mistakes. Finally, after nearly a third of workflow, the artist was able to follow along with the tutorial, although with some errors which is seen with green color (moderate distance) in bottom-right corner of each block of third row or column.

Cross-Workflow Scrubbing. While the user scrubs through the timeline of one workflow, CrossComp can automatically snap the other workflows to the closest snapshot in terms of the edit distance. This cross-workflow scrubbing allows the user to inspect how all of the workflows progressed, even though the artists may have worked at a different pace. We define the closest snapshot in a specific workflow to a given snapshot as the snapshot with the lowest regularized edit distance from the given.

Spatial Filtering. Similarly to MeshFlow and 3DFlow, the user can perform spatial filtering on the workflows to find when the artists modified a region of interest. When the user selects a face in one workflow, the corresponding faces in the other workflows are selected, too. The timeline (colorbar below the model) is darkened to indicate the edits that do not modify the selected faces. See Figure 8 for an example.

7 Results

Figure 9 displays the results of the Transporter, Station, and Interceptor workflows. Below we will discuss briefly some observations for these workflows.

Transporter. Generally, all four subjects followed the Transporter tutorial relatively closely. The fifth workflow contained a few corrected errors (visualized as the purple outlier runs in the first column.) The first and fourth workflows were the closest pair of workflows. While all the final meshes were similar in shape, the differences of proportions and fine details of the engines caused a divergence of the workflows in the 3D embedded view.

Station. In the Station task, one of the subjects did not submit the completed task, so the second workflow remains as a cube. Also, the third workflow only loosely followed the tutorial and involved fewer edits than the video tutorial, and the subject did not have the mesh positioned correctly for the mirror modifier to duplicate the other three quadrants properly, resulting in an outlier in the first column. The first, fourth, and fifth workflows followed each other closely.

Interceptor. Where the previous tasks were presented as a video tutorial, the interceptor task was presented to the subjects as a final target mesh. The subjects were free to construct the mesh using any techniques and in any order. One important observation to note is that while the artists can construct the mesh in any order, the majority of divergence was due to differences in adjacencies. For example, the first and fourth workflows are relatively close in the first column, because their meshes are topologically quite similar. However, the second, third, and fifth workflows contained many changes in adjacency (missing features, extra faces, incorrectly connected faces, etc.) and therefore appear to diverge from the first and fourth workflows. The extremely large distances seen in the third workflow are due to setting incorrectly the mirror modifier parameters.

7.1 Feedback

We presented our findings to Jonathan Williamson, a professional digital modeling artist and instructor for CG Cookie, in order to gather some open-ended feedback. Williamson stated that the embedded view made it clear when the artists made and then corrected a mistake and that the curves hinted at the similarities of the workflows. When shown the Interceptor dataset, he remarked about how the subjects took a similar approach to constructing the spaceship despite not having step-by-step instructions, which was an unexpected observation.

Williamson said that he is quite excited about the results and interested in finding ways to use CrossComp to instruct. One usage scenario he proposed centers on an assignment he has given before, which follows closely the Interceptor workflow, where he asks the students to create a challenging mesh. CG Cookie has created four exercises of this type in the past, and Williamson states that while they receive many more requests to do more, they have not been able to due to the time involved in reviewing the workflows. After looking over the submitted final versions, he would create a video tutorial on constructing the model while pointing out common mistakes and pitfalls seen in the students' results. He believed that CrossComp would help him in finding, analyzing, and pointing out these situations.

7.2 Limitations

There are a few limitations to our input data and approach to analyzing. We discuss some of these limitations in this subsection.

Input Data. We designed our experiments to include instructions for using Blender and to be relatively short and simple. This decision was motivated by some of our subjects may have no experience using Blender and possibly only little experience modeling. Furthermore, despite walking the subjects step-by-step through first three tasks, one workflow was submitted incomplete, and two submitted with gross errors. Although these issues limit the scope of our experiments to novices and amateurs, we found that CrossComp was able to produce intuitive results that helped with making key observations about individual workflows and with comparing the workflows with one another. We leave for future work the study of more experienced subjects performing longer and more advanced tasks.

Correspondences. MeshGit builds a one-to-one correspondence between two meshes. A discrete correspondence works well when the two meshes are very similar in terms of face adjacency. However, when only a fuzzy correspondence is necessary or computable, such as when the models use the mesh to provide a relatively loose representation the surface, other surface correspondence methods might be more appropriate. We chose to use MeshGit's correspondence building method and designed our experiments to fit in these limitations, because MeshGit computes a mesh edit distance which we use directly. We leave the exploration of other correspondence building and distance computing methods for future work.

Edit Distances. Computing a full pairwise edit distance can become quite expensive, growing polynomially in the lengths of workflows and number of subjects. It should be noted that the pairwise distances needs to be computed only once and then

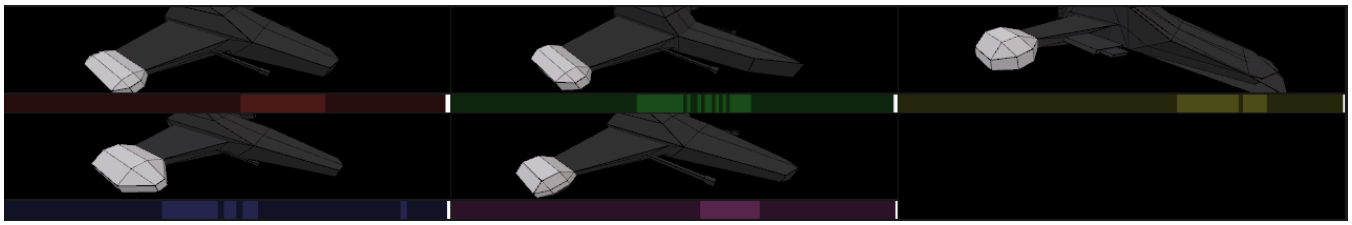

Figure 8: *Filtering to spatial selection. The user selected the faces of the engine in one task, and the corresponding faces are highlighted for the other tasks. The edits that modify the selected region are highlighted in the timeline.*

cached, is a highly parallel operation, is symmetric, and can be only sparsely computed.

8 Conclusion

In this paper, we presented CrossComp, a method for comparing multiple artists performing similar tasks. Motivated by real-world digital modeling exercises, we demonstrated how to use intra- and inter-correspondences within a set of workflows to compute a pairwise snapshot edit distance. CrossComp can visualize these edit distances as a heat map, where similar and dissimilar snapshots are identified using cool and hot colors, respectively. CrossComp can also perform nonlinear dimensionality reduction on the distances to embed the workflows in a 3D space, where curves and distances indicate similar editing patterns or mistakes and errors. Open-ended feedback from a professional artist and instructor indicate that a system like CrossComp could strongly benefit the instruction community.

References

DENNING, J. D., AND PELLACINI, F. 2013. MeshGit: Diffing and merging meshes for polygonal modeling. *ACM Transactions on Graphics 32*, 4, 35:1–35:10.

DENNING, J. D., AND PELLACINI, F. 2014. 3DFlow: Continuous summarization of mesh editing workflows. Tech. rep., Department of Computer Science, Dartmouth College, June.

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.

KONG, N., GROSSMAN, T., HARTMANN, B., AGRAWALA, M., AND FITZMAURICE, G. 2012. Delta: a tool for representing and comparing workflows. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, 1027–1036.

LAFRENIERE, B., GROSSMAN, T., AND FITZMAURICE, G. 2013. Community enhanced tutorials: improving tutorials with multiple demonstrations. In *Proceedings of the 2013 ACM annual conference on Human factors in computing systems*, ACM, New York, NY, USA, CHI '13, 1779–1788.

MATEJKA, J., LI, W., GROSSMAN, T., AND FITZMAURICE, G. 2009. CommunityCommands: Command recommendations for software applications. In *UIST*.

PAVEL, A., BERTHOUZOZ, F., HARTMANN, B., AND AGRAWALA, M. 2013. Browsing and analyzing the command-level structure of large collections of image manipulation tutorials. Tech. rep., Electrical Engineering and Computer Sciences, University of California at Berkeley, October.

TENENBAUM, J. B., DE SILVA, V., AND LANGFORD, J. C. 2000. A global geometric framework for nonlinear dimensionality reduction. *Science 290*.

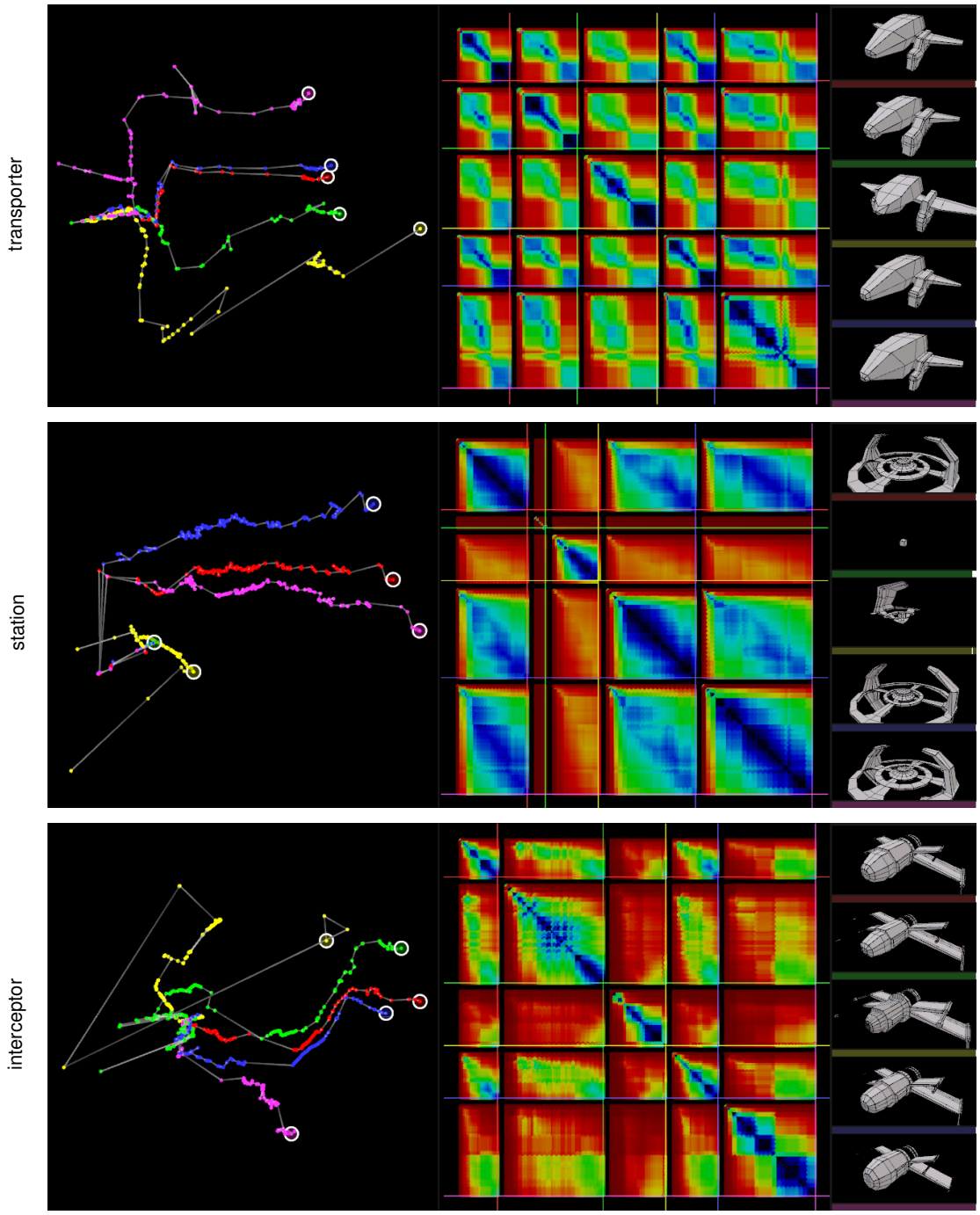

Figure 9: *Results of Transporter, Station, and Interceptor.*