Eurographics Workshop on Visual Computing for Biology and Medicine (2014)
I. Viola, K. Bühler, and T. Ropinski (Editors)

Real-Time Dense Nucleus Selection from Confocal Data

Y. Wan[1] and H. Otsuna[2] and K. M. Kwan[3] and C. Hansen[1]

[1]Scientific Computing and Imaging Institute, University of Utah, USA
[2]Department of Neurobiology and Anatomy, University of Utah, USA
[3]Department of Human Genetics, University of Utah, USA

Abstract

Selecting structures from volume data using direct over-the-visualization interactions, such as a paint brush, is perhaps the most intuitive method in a variety of application scenarios. Unfortunately, it seems difficult to design a universal tool that is effective for all different structures in biology research. In [WOCH12b], an interactive technique was proposed for extracting neural structures from confocal microscopy data. It uses a dual-stroke paint brush to select desired structures directly from volume visualizations. However, the technique breaks down when it was applied to selecting densely packed structures with condensed shapes, such as nuclei from zebrafish eye development research. We collaborated with biologists studying zebrafish eye development and adapted the paint brush tool for real-time nucleus selection from volume data. The morphological diffusion algorithm used in the previous paint brush is restricted to gradient descending directions for improved nucleus boundary definition. Occluded seeds are removed using backward ray-casting. The adapted paint brush is then used in tracking cell movements in a time sequence dataset of a developing zebrafish eye.

Categories and Subject Descriptors (according to ACM CCS): I.3.8 [Computer Graphics]: Methodology and Techniques—Interaction techniques; J.3 [Life and Medical Sciences]: Biology and genetics—;

1. Introduction

In biology research, visualization and segmentation techniques have developed hand-in-hand for analyzing complex anatomical and developmental phenomena. For quick and accurate decision making in visual analysis, user interactions play a key role in bridging all the pieces of the analytic process, i.e., visualization, segmentation, and computation. In [WOCH12b], an interactive technique was proposed for extracting complex neural structures under user guidance. This technique allows users to paint brush strokes directly on rendered images and segment desired structures from the 3D volume. A brush stroke is usually composed of two substrokes, an inner stroke for seeding and an outer stroke for the definition of a fuzzy growing region. The integration of this technique to an existing visualization system, FluoRender [WOCH09] [WOCH12a], allows easy visual analysis of neural structures. Although some users commended its intuitiveness for selecting fibrous and branching structures from confocal scans, especially those with strong noise, the method was less effective when it was applied to selecting densely packed structures with condensed shapes, such as nuclei from scans of zebrafish eyes [KOK*12]. There are mainly two reasons why the technique broke down. First, the paint brush tool was designed for neural structure selection. The morphological diffusion calculation used in the region growing process tends to connect adjacent structures, which helps when a delicate fiber has dimly-connected or disconnected sections. However, it is undesirable that the growing region often selects neighboring nuclei when in fact only one is seeded. Second, the paint brush strokes are projected from the image plane to the volume. Both seeding and diffusion are calculated within the cone or cylinder of the projection, irrespective of the occlusion between structures. Therefore, undesired but obstructed nuclei can be easily selected, which have to be removed by an eraser brush painted from a different viewing angle. When multiple nuclei need to be selected from a single scan or multiple scans of a time sequence, manipulating among different view angles to remove incorrectly selected nuclei becomes laborious.

We collaborated with biologists studying zebrafish eye development and adapted the paint brush tool in FluoRender for real-time nucleus selection. The contributions of this paper are threefold. First, we adapted the morphological diffusion algorithm for nucleus selection, restricting the diffusion

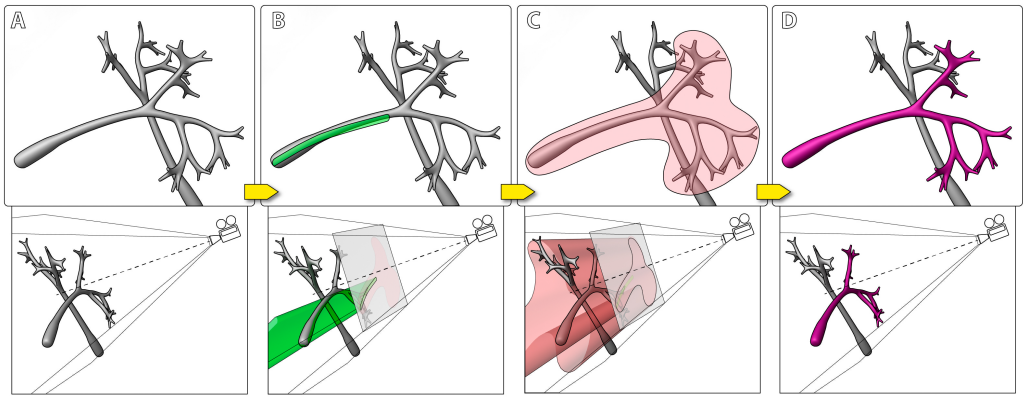

Figure 1: *Selecting a branching structure using the paint brush in FluoRender. Top row: visualizations from the user's point of view; Bottom row: a different viewing angle demonstrating the projections of the brush strokes. A: We would like to select one branching structure out of two, one overlapping another from the user's view; B: The seeding stroke is painted on the visualization and projected into the volume to define the seeds; C: The diffusion stroke is painted and projected similarly; D: After calculating the morphological diffusion of the seeds within the growing region, the front branching structure is selected.*

to only gradient descending directions. Second, we removed occluded structures in the seeding process of the paint brush using backward ray-casting. Finally, we presented detailed application cases to demonstrate the use of the paint brush tool in biology research of zebrafish eye development. The paper is structured as follows. We describe background and related work in Section 2. We present the technical details of the paint brush tool for nucleus selection in Section 3. Then, in Section 4, we discuss how the integration of the new paint brush tool helps cell tracking and development analysis in zebrafish eye studies. We conclude this paper in Section 5.

2. Background and Related Work

2.1. Paint Brush for Neural Structure Selection

The paint brush tool in FluoRender was proposed to select and extract neural structures from confocal microscopy scans [WOCH12b], on which the work in this paper is based. It is helpful to revisit certain design choices made previously. Figure 1 illustrates the concept of the dual-stroke paint brush, one for seeding and one for diffusion region definition. For one branching structure to be selected, such as a neuron, it assumes that there exist both overlapping and non-overlapping sections from a single point of view. To successfully select the desired structure without changing the viewing angle, seeds have to be placed on the non-overlapping part and no connectivity exists between the desired structure and others within the diffusion region. The assumption holds well for most fibrous structures in confocal data. For neural structures with relatively simple shape, a brush combining both seeding and diffusion region definition is sufficient. For more complex shapes, such as neurons with a large number of branches, it usually requires multiple strokes of seeding and diffusion region definition. However, if structures are densely packed, it becomes difficult to obtain a non-overlapping section for seeding. The diffusion calculation also tends to connect different structures, resulting in excessively selected structures that have to be removed from a different viewing angle using the eraser brush. Unlike neural structures, certain cellular structures, such as nuclei, can be densely packed and become difficult to select with the previous brush tool. Figure 2 shows an example of the datasets from our collaborating biologists. The original dataset contains 210 continuous scans of a zebrafish eye in development. It also contains two channels of fluorescently stained nuclei and cell membranes. Here, we will be focusing on the nucleus channel only, as tracking the movement of individual nuclei reveals their development pattern. To facilitate analysis on organ or tissue development at cellular level, we needed to redesign the paint brush to make nucleus selection easier.

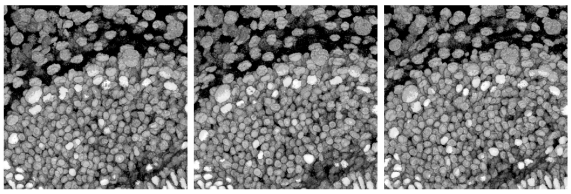

Figure 2: *Nuclei in the eye of a zebrafish embryo. The three panels are scans captured at different time points. In the eye region, cells are densely packed in 3D, making real-time selection or segmentation of individual nucleus difficult.*

2.2. Related Work

Selection by clicking (the magic wand tool or bucket tool), drawing (thin lines, such as the lasso tool), and painting (or brushing with thick strokes) are commonly adopted in 2D image creation and edit tools, such as Adobe Photoshop [Ado14]. Similar user interactions have been carried over and incorporated into tools for creating and editing 3D polygonal objects, such as Autodesk Maya [Aut13a] and Mudbox [Aut13b]. The prevalence of these artistic tools in the entertainment industry also increased their popularity among casual users from different areas, including biology researchers and visual analysis experts studying volumetric data. It is natural and intuitive to incorporate those similar user interactions into volume visualization and editing tools as well. Yuan et al. [YZNC05] presented a method for cutting out 3D volumetric structures based on simple strokes that are drawn directly on volume rendered images. They used a graph-cuts algorithm and could achieve near-interactive speed for CT and MRI data. Chen et al. [CSS08] enabled sketch-based seed planting for interactive region growing in their volume manipulation tool. Owada et al. [ONI*08] proposed several sketching user interface tools for region selection in volume data. Their tools are implemented as part of the Volume Catcher system [ONI05]. Bürger et al. [BKW08] proposed direct volume editing, a method for interactive volume editing on GPUs. They used 3D spherical brushes for intuitive coloring of particular structures in volumetric scalar fields. Abeysinghe and Ju [AJ09] used 2D sketches to constrain skeletonization of intensity volumes. They also tested the interactive tool on a range of biomedical data. However, different from the general image editing applications of the artistic tools, studies with volume data often require specialized functions for different structures. We feel that there has not been sufficient emphasis on real-world applications. The previously proposed volume editing tools are neither adequately generalized for all application scenarios nor sufficiently differentiated for each specific case. As we learned from our users' frustrations when trying to apply the tool for neural structure selection to nuclei, we decided to differentiate our existing paint brush tool and make adjustments so that it is suitable for selecting nuclei from fluorescence microscopy data in applications such as Kwan et al. [KOK*12].

3. Paint Brush for Nucleus Selection

We would like to solve the two problems of the previous paint brush when selecting an individual nucleus shown in Figure 2. First, we restrict the morphological diffusion to only gradient descending directions, so that neighboring nuclei to the seeded nucleus will not be incorrectly selected. Next, we check seed visibility using a backward ray-casting algorithm to prevent occluded nuclei from being selected.

3.1. Adapted Morphological Diffusion

Morphological diffusion is one discretization form of the heat equation using morphological gradient [RSB93] [Soi02]. It is an iterative filtering process of a scalar field, shown in Equation 1, where $u(x)$ is the scalar field, $g(x)$ the stopping function, and $\delta(x)$ the dilation of the scalar field.

$$u_{i+1}(x) = (1 - g(x))u_i(x) + g(x)\delta_i(x) \tag{1}$$

We chose the morphological diffusion for the paint brush, because it suited selecting neural structures in real-time. The morphological diffusion creates a stronger seed flow and converges faster in a separate mask volume than the standard anisotropic diffusion. The resulting mask volume after the application of each stroke is used to highlight, extract, or erase the selected structures. The gradient magnitude of the original data volume is used in the stopping function $g(x)$ to stop the diffusion at structure boundaries. However, when there exist many densely packed structures, thus many interfaces between structures, it becomes difficult to set proper stopping function parameters that can stop the diffusion at every interface each time when the brush stroke is applied. This is the case for nucleus selection from datasets such as in Figure 2. Notice the similarity between our method and the marker-controlled watershed algorithm [RBD91] [RBD92]: both grow regions that are marked by seeds. However, the watershed algorithm is usually for a full segmentation of an entire dataset, where the region growth is stopped when two regions meet, or barrier built. On the other hand, our paint brush tool is for generating partial selections of a dataset, where the region growth is unilateral. Therefore, we are in need of a different strategy to stop the morphological diffusion at boundaries than the watershed.

For real-time nucleus selection without preprocessing, we propose an adaptation of the morphological diffusion, which evaluates Equation 1 only at gradient descending locations of the original scalar field. Equation 1 is a diffusion process that generates a seed flow in the mask volume [WOCH12b]. Different from the binary mask volumes used in many other segmentation tools, the mask volume here contains scalar values, so that a seed flow can be computed. If we see the seed flow as energy transmission, the energy is always flowing from high to low, i.e., gradient descending directions. However, the gradient direction at the same location in the original data volume may differ from that of the seed flow in the mask volume. Figure 3 shows the gradients in both mask and data volumes for a given voxel. Notice that we are using the morphological gradient here, which is defined as the vector pointing from the maximum value within a neighborhood to the current location. In Figure 3, the circle indicates the voxel under discussion, which is surrounded by neighboring voxels within a 3×3 window, and M represents the maximum value within the neighborhood. In Case 1, we should allow the seed flow, as the gradient direction in the data is the same as in the mask; in Case 2, we should stop the seed flow,

as the gradient flow in the data is perpendicular to that of the mask. Specifically, we modify the stopping function $g(x)$ in Equation 1 and use the dot product of the two gradients to restrict the morphological diffusion to gradient descending directions (Equation 2).

$$
\begin{aligned}
p &= dot\left(\frac{\nabla_{mask}}{\|\nabla_{mask}\|}, \frac{\nabla_{data}}{\|\nabla_{data}\|}\right) \\
g'(x) &= \begin{cases} g(x)\cdot p, & p > 0 \\ 0, & p \leq 0 \end{cases} \\
u_{i+1}(x) &= (1 - g'(x))u_i(x) + g'(x)\delta_i(x)
\end{aligned}
\tag{2}
$$

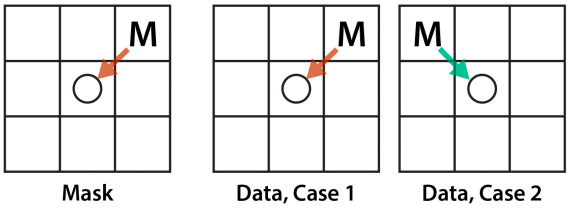

Figure 3: *Gradient directions of a given voxel can differ in mask and data volumes.*

The above adaptation of the morphological diffusion assumes that a nucleus has a spherical or elliptical shape with the grayscale intensity smoothly tapering down outward from the highest value at its center. However, confocal scans can contain noise and nuclei can have elongated shapes depending on their development stages. The limitation becomes most apparent when users were required to click on a nucleus for setting seeds. The seeding region is thus small and has a round or square shape. The seeds also have to include the peak intensity values within the nucleus. Otherwise the selection becomes incomplete, since the adapted morphological diffusion only proceeds in gradient descending directions. Therefore, for the adapted morphological diffusion to work on noisy and elongated nuclei, we provide the dual-stroke paint brush tool to ensure sufficient seeding. The dual-stroke paint brush has several advantages over simple clicking (magic wand tool) or a lasso tool for selecting nuclei. First, it has clearly defined regions for both seeding and diffusion, where calculations are localized within a small region. Second, the size of the brush can be adjusted according to nucleus size. We can ensure sufficient seeding area to cover peak intensity values within a nucleus. Third, brush strokes can follow the shape of an elongated nucleus more easily than both clicking and lasso tool. Figure 4 demonstrates selecting a nucleus from the zebrafish eye dataset as in Figure 2. It shows a magnified view of a region filled with densely packed nuclei. This region also contains noise. We use the dual-stroke brush tool to select the nucleus at the center of this view. The stroke sizes are adjusted to ensure the seeding region (red) is confined within the nucleus and the diffusion region (green) includes the entire nucleus. When the morphological diffusion is not restricted to the gradient descending directions, the neighboring nuclei, which are connected to the desired nucleus through low intensity values, are incorrectly selected (Figure 4C). The desired nucleus is selected easily and satisfactorily when we use the adapted morphological diffusion.

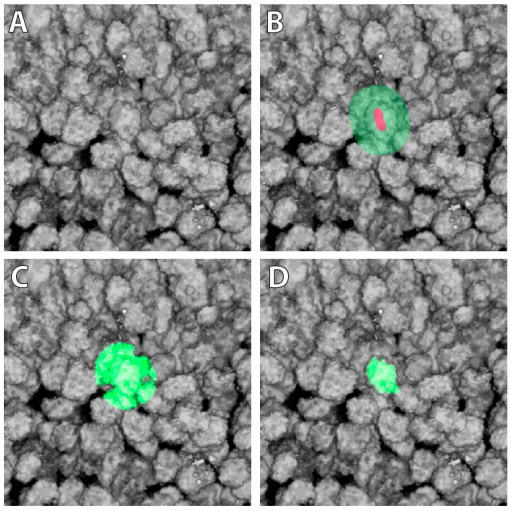

Figure 4: *Result of nucleus selection using the adapted morphological diffusion. A: A magnified view of a region of the zebrafish eye dataset; B: We would like to select the center nucleus using the paint brush tool, which has a center stroke (red) for seeding and an outer stroke (green) for diffusion; C: When morphological diffusion is not restricted to the gradient descending directions, the neighboring nuclei are incorrectly selected; D: The desired nucleus is easily and satisfactorily selected with the adapted morphological diffusion.*

3.2. Occlusion Check

Using brush stroke projections to select structures in volume data sometimes results incorrect selections of occluded voxels, which are actually disconnected from the desired structures. It is especially a problem for the nuclei in Figure 2, where structures are densely packed three dimensionally. To eliminate the need for refining selection from different viewing angles and thus allow intuitive nucleus selection, occluded structures must be excluded from selection in real-time. A most straightforward method would be performing a connected component analysis after the selections been made and then removing the occluded components using occlusion query. This method requires additional passes that may render the selection non-real-time. Considering the unique selection process of our dual-stroke paint brush tool (Figure 1), we propose a backward ray-casting calculation built in the seeding process to exclude occluded structures,

which introduces little computational overhead without additional passes.

Despite the fact that the selection depends on both the seeding and diffusion processes, connectivity has been taken into consideration in the diffusion process already. It suggests that it is adequate to perform occlusion check only in the seeding process, which also restricts computation within a much smaller region. Furthermore, occlusion check can be performed within the seed projection process without additional query passes, as stroke projection uses the same perspective as the viewport camera for rendering. Specifically, we propose the following backward ray-casting calculation, which is illustrated in Figure 5.

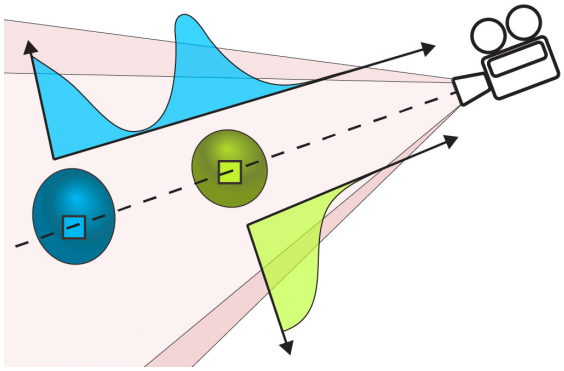

Figure 5: *Backward ray-casting for occlusion check of seeds.*

In Figure 5, there are two nuclei located on the same line of sight, and thus the second nucleus (blue) is occluded by the first one (green). When we paint a brush stroke on the viewport, the first nucleus is desired for selection, since it is the only one visible. During the seeding process, voxels with high intensity values within both nuclei are selected as seeds. If we proceed to the diffusion process without occlusion check, both nuclei will then be selected. To check the occlusion of the seeds, we first assume that the selected voxels are candidate seeds. For each candidate seed, we cast a ray from it backward to the camera. For easy volume traversal, we calculate the rays using the object space coordinates. When different view projection types are used, the ray directions are calculated differently. For orthographic projections, rays from all candidate seeds are parallel to each other, pointing against the camera's viewing direction. The vector $\vec{v}$ of the ray direction is calculated as:

$$\vec{v} = normalize(M_{mv}^{-1} \cdot \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}) \tag{3}$$

It transforms the reverse viewing direction $\begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$ back to the object space using the inverse of the model-view transformation matrix M_{mv}^{-1}. For perspective projections, rays from different candidate seeds can be different. The vector $\vec{v}$ of the ray direction is calculated as:

$$\vec{v} = normalize(M_{mv}^{-1} \cdot \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} - \vec{t}) \tag{4}$$

It transforms the camera position in the eye space $\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ back to the object space, and then calculates the vector from the seed position $\vec{t}$ to it. Next, we sample through the volume using the ray, similar to a standard ray-caster. Figure 5 illustrates the ray profiles (sample intensities along the ray) for each of the two seed candidates. To determine if a seed candidate is occluded, we search the ray profile to see if there is any other candidate seed in a separate connected component. For example, for the blue seed candidate (represented by a blue square), the sample value decreases first when we traverse the ray. It then increases when the ray enters the green nucleus. When the ray reaches the green seed candidate, which is disconnected from the blue seed candidate along the ray, we terminate the ray and discard the blue seed candidate because it is occluded. For the green seed candidate, the ray will reach the volume boundary without hitting a disconnected seed candidate. So we keep the green candidate as a seed for the diffusion process.

The proposed backward ray-casting can be calculated in parallel within the shader code that originally generates the seeds. The computational overhead equals performing the connected component analysis in 1D, which is much less complex than the straightforward method discussed in the beginning of this section. In addition, a ray can be terminated early when the first disconnected seed candidate is found. Furthermore, less sampling is required than rendering the same volume with a standard ray-caster. Also considering that the backward ray-casting is performed for seeds only, the paint brush tool equipped with occlusion check can select nuclei or general cellular structures in real-time. Figure 6 shows two examples of using the paint brush on synthetic data when occlusion check is enabled. It is easy to select just the occluding one from two disconnected objects in Example A. However, connected parts of one object are selected even when occluded (Example B). Selecting nuclei from real-world datasets is discussed in detail in the following case studies. Although the occlusion check with backward ray-casting can be incorporated into other user interaction schemes, such as magic wand and graph

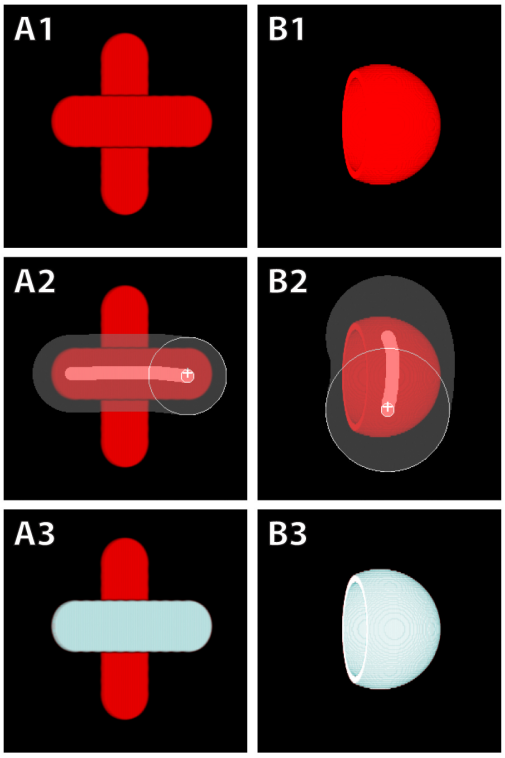

Figure 6: *Synthetic data for testing the occlusion check. Dataset A contains two disconnected objects. The front object is selected when we paint on it, without selecting the object behind. Dataset B contains a hollow ball cut open with a clipping plane. The entire ball can be selected even though the back wall is not seeded due to occlusion, because connected structures are selected in the diffusion process.*

cut [GPS89] [BJ01], it works most intuitively with the dual-stroke paint brush for selecting small-scale structures. This is because it has two separate stages for seeding and diffusion, each having clearly defined regions, the in-between of which delimits the search region for structural boundaries. In contrast, the magic wand tool does not specify the search region for boundaries. The selection can include more structures than expected. Other methods like graph cut requires multiple strokes to specify both background and foreground, which may still become time-consuming and ineffective when the desired structure is surrounded by similar objects.

4. Case Studies

The paint brush tool for nucleus selection is designed for the analysis of zebrafish eye development. The goal of this research is to track the movement and lineage of each cell within the eye region and establish its development pattern. Despite the fact that there exist tools and algorithms for both fully manual and automatic cell tracking, either it requires laborious work or the results are unsatisfactory, as the cells are densely packed in 3D space. In the following two case studies, we demonstrate how the paint brush tool for nucleus selection can help in both manual and automatic cell tracking.

4.1. Manual Cell Tracking

The most common practice for manually tracking cells in confocal data as in Figure 2 is selecting each nucleus from each image slice of each time point data, and then connecting the selected nuclei as one trajectory. Since a time sequence as in Figure 2 usually contains several hundred time points, each containing several thousand nuclei, manually tracking the complete dataset becomes extremely difficult. Therefore, biologists would usually start from a time point toward the end of the time sequence, choose landmark nuclei representing interesting features, and track backward in time. However, manually selecting nuclei from 2D image slices of volume data still would not guarantee an error-free result. This is because nuclei are migrating, and sometimes touching each other, three dimensionally. Viewing and selecting a nucleus only from 2D image slices may lose the context and lead to erroneous selections. With our paint brush tool, it allows viewing and selecting on the volume-rendered result, which keeps the surrounding context of a nucleus for selection. Furthermore, nuclei deep inside the volume can be revealed using clipping planes. For manual cell tracking, our work improves the accuracy and intuitiveness for single cell selection. Combined with other facilitating user interface elements, such as keyboard shortcuts, cell trajectories are generated with less time than tracking from image slices.

Figure 7 is a step-by-step example of generating one cell movement trajectory using the paint brush tool. The dataset is the same zebrafish eye development time sequence as in Figure 2. We track one cell forward in time starting from the first time point. We first paint on the nucleus of the cell to select it. A starting point of the track is then created by calculating the centroid location of the selection, which is labeled as Track 1 in Figure 7C. Then we can use either a keyboard shortcut or the time slider to advance one time point. The previously selected nucleus disappears, but we are able to visually track and identify the nucleus in the second time point. We then paint on the nucleus and select it. The centroid location of the selection is then calculated as the second point of the trajectory. We advance through time and repeat the steps for the subsequent time points. For this example dataset, since the playback of the time sequence is real-time, we can easily create one trajectory for all time points. Notice that depending on the requirement of tracking precision, we can skip time points for faster tracking, as long as the tracked nucleus can be visually identified. For cells undergoing mitosis process, we need to create individual trajectories for

each daughter cell after they separate, which are tracked as standalone cells afterwards. Although we track the cell forward in time in this example, tracking forward and backward use the identical procedure.

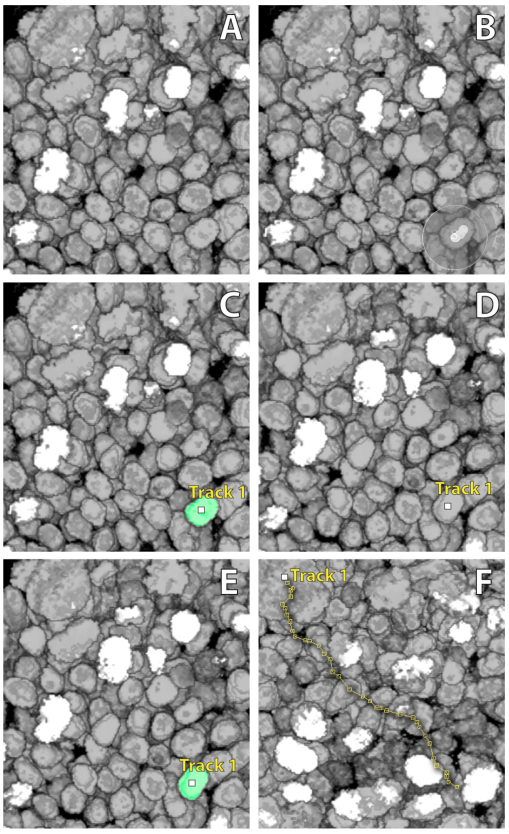

Figure 7: *Manual cell tracking with the paint brush tool. A: The first time point of the zebrafish eye development time sequence; B: We start tracking the nucleus at the lower right corner by painting a brush stroke. C: The nucleus is selected, its centroid location used as the starting point of the trajectory; D: We advance to the second time point of the sequence; E: Since the selected nucleus in the first time point can be visually identified in this time point, we can select it using the paint brush tool, its centroid location then calculated and added to the trajectory; F: We repeat the above steps and continue selecting nuclei for the subsequent time points. A complete trajectory of the cell is then generated.*

4.2. Trajectory Editing in Semi-Automatic Cell Tracking

Although tracking cells using the paint brush tool is more intuitive than selecting them from 2D image slices, the manual tracking method can only be used for precision when all other methods fail. It is still extremely time consuming if all cells were to be tracked in such manner. On the other hand, fully automatic cell tracking usually requires both automatic segmentation and matching algorithms. Even the minuscule error within one time point easily propagates and magnifies throughout subsequent time points. Without human interference, fully automatic tracking results usually deteriorate over time and cannot be used for any real analysis, especially for noisy data. In this case study, we present an example of using FluoRender's paint brush tool for trajectory editing in semi-automatic cell tracking.

The user-guided, semi-automatic cell tracking workflow is based on Synthetic Brainbows [WOH13] and maximum bipartite matching algorithms [Wes99]. For a time sequence from the developing zebrafish eye scan, we first process each time point using the Synthetic Brainbows process. This process takes advantage of GPU randomness within framebuffer feedback loops and generates clusters of nuclei, each with a uniquely assigned ID. For the nucleus channel from each time point of the scan, we first assign a unique ID to each nonempty voxel. Then we perform successive dilation operations on the entire ID volume. However, the dilation operations are computed using framebuffer feedback loops instead of framebuffer Ping-Pong. In addition, the dilation speed of each voxel of the ID volume is controlled by properties of the corresponding voxel of the original data volume. Properties, including grayscale intensity, gradient magnitude, grayscale variance, etc., are computed to modulate the dilation speed, so that IDs within one nucleus are merged faster than those on the boundaries. This process is repeated for all time points to generate the initial segmentation of the time sequence. Ideally, the IDs from two consecutive time points can be linked in pairs using the bipartite matching algorithm, which is then repeated for all subsequent time points, and thus solves the tracking problem. Different criteria can be used for establishing a bipartite graph between two consecutive time points. For example, an edge is created between two vertices of IDs if there is overlap between the two components with the same IDs, the overlapping volume being used as the weight of the edge. Similarly, proximity can be used for linking two vertices with an edge, which is weighted by the distance. Furthermore, shape similarity can be used to establish the bipartite graph as well. For the sake of simplicity and also because the time sequence used for the case study has ample temporal samples, we only consider using overlapping as the criterion for edges. The maximum bipartite matching algorithm finds the set of edges without common vertices and maximizes the overall weight at the same time. Detailed discussion of the algorithm can be found in [Wes99], with applications of maximum bipartite matching in cell tracking found in [CCGR10] and [CGCR13]. However, our experiments showed that the maximum bipartite matching can only be used for ideal cell tracking cases, where no segmentation error or mitosis should be present. This is because the maximum matching algorithm assumes the mapping of vertices between the two sets (time points)

is always one-to-one. It is obviously not the case if one cell is going through mitosis (one-to-two mapping) or death (no mapping). More importantly, over- and under-segmentations cause some cells to incorrectly split into different components, and some fuse together. Such imperfections in segmentation are unavoidable and cause the links between cells to shift from time point to time point. Therefore, we propose incorporating user interactions and guidance into the above tracking workflow, so that segmentation errors and incorrectly matched cell pairs can be edited directly.

Our semi-automatic cell tracking workflow still uses the preprocessed results from the Synthetic Brainbows algorithm, which generates components with unique IDs for each time point. Then, a complete link graph comprising all time points is generated by executing the maximum bipartite matching algorithm for all the consecutive time points. As mentioned, we only use the overlapping criterion for edges and associated weights when the graph is established as the initialization step for maximum matching. Also notice that links between IDs are one-to-one on the link graph generated from maximum matching. However, two linked IDs are not necessarily the same, their values and linkage also editable. We designed a user interface built into FluoRender, which allows loading, visualizing, and editing cell tracking results generated from Synthetic Brainbows and maximum matching. Figure 8 shows the dialog window for ID and trajectory editing. In this dialog, we can choose and load a link map generated from the ID matching step, or generate a new link map from scratch. After editing, the link map can be saved on disk. There are two types of errors that need editing. First, there are tracking errors, which are nuclei segmented correctly but linked incorrectly, therefore need relinking. Also, one ID can be linked to two in case of mitosis. Second, there are segmentation errors, which are components with incorrectly assigned IDs. For over-segmentation, where one nucleus is assigned multiple IDs for its components, we need to combine them with a single ID. For under-segmentation, where multiple nuclei are fused and assigned with only one ID, we need to select each nucleus and assign a new unique ID. We discuss these two types of editing with examples next. Figure 9 demonstrates two examples of editing the cell tracking results with the help of the paint brush tool. In the first example, there is a nucleus tracked to the incorrect one, although their segmentations are correct. We paint and select the nucleus in the first time point (Figure 9A1). Its ID is then loaded to the list for the current time point. Then we advance to the second time point by pressing the Forward button, the ID of the previously selected nucleus is loaded to the list for previous time point. The selection is also updated to the automatically tracked nucleus (Figure 9A2). Its ID is loaded to the list for the current time point, which is incorrect. We use the eraser paint brush to remove the incorrect selection and then select the correct one (Figure 9A3). Then we use the ID link tool to link the correct pair and thus fix the tracking error (Figure 9A4). In the second example, we fix an error in

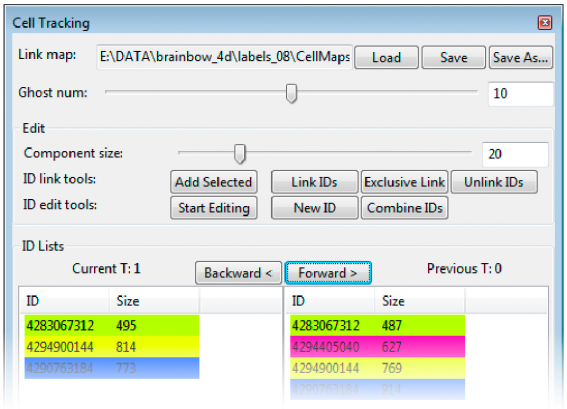

Figure 8: *Dialog window for editing cell tracking results. User can load, generate, and save a link map that contains all tracking information. Two sets of tools are provided to edit both tracking and segmentation errors. For tracking errors, IDs can be linked or unlinked; for segmentation errors, a new ID can be generated or several IDs combine into one. IDs of currently selected components are listed in the list. A list of IDs from previous time point is also provided for linking and unlinking. User can also use the Backward and Forward buttons to perform automatic tracking.*

the automatic segmentation. A nucleus in the first time point is segmented into four different components, each assigned an ID (Figure 9B1). Only the largest (magenta) component of this nucleus is correctly tracked into the next time point. We select the entire nucleus with the paint brush tool (Figure 9B2) and then press the Combine IDs button. The selection is assigned with the ID from the largest component (Figure 9B3). Then the nucleus can be tracked correctly in the second time point (Figure 9B4).

5. Conclusions and Future Work

In this paper, we present and discuss a paint brush tool for real-time nucleus selection from volume data acquired with fluorescence microscopy. We also demonstrate the application of our paint brush tool in biology research, i.e., cell tracking in the study of zebrafish eye development. We have shown that our paint brush is an intuitive tool in both manual tracking and trajectory editing for semi-automatic tracking. Our work also demonstrates that general tool and algorithm needs adaptation for different application scenarios. At current stage, both biologists and visual analysis experts agree that semi-automatic tracking with intuitive user interactions is the most practical solution to ensure accuracy, although both automatic segmentation and tracking methods used in this paper can be improved. We will continue our research on visual analytics methods for cell development studies. For

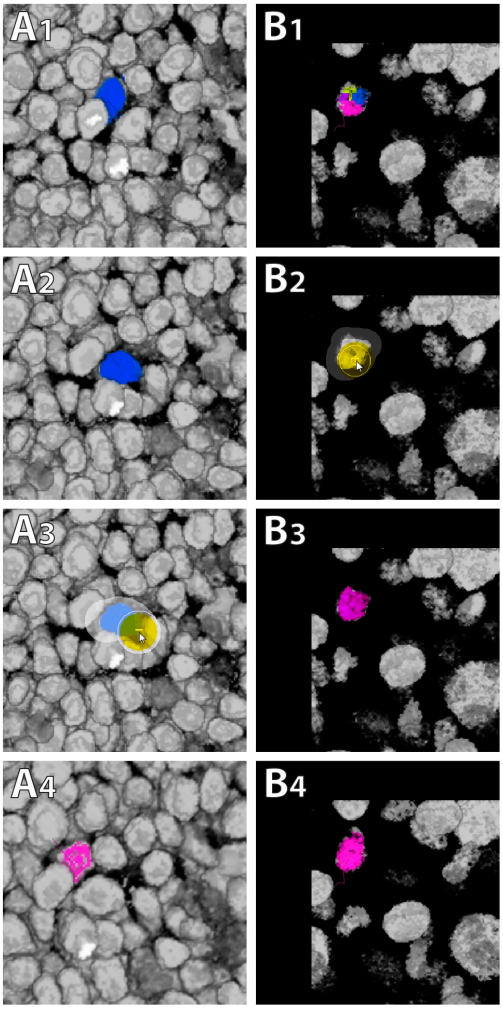

Figure 9: *Examples of fixing a tracking result. The top row shows a tracking error. A nucleus (in blue) of A1 is incorrectly tracked to a nucleus of A2. The trajectory is unlinked by unselecting the incorrectly tracked nucleus (A3) and relinking to the correct one (magenta in A4). The bottom row shows a segmentation error. A nucleus in B1 is over-segmented into four components. The entire nucleus is selected (B2), IDs from its components combined (B3). The combined ID is then tracked to the correct nucleus in the next time point (B4).*

example, a limitation of the semi-automatic tracking workflow in this paper is that detection of errors greatly relies on user experience, especially for errors generated from mitosis events. For future work, we would like to explore visualization methods that help the detection of problems in cell tracking, which can then be easily fixed using the paint brush tool.

Acknowledgments

This research was sponsored by NIH: FluoRender: An Imaging Tool for Visualization and Analysis of Confocal Data as Applied to Zebrafish Research, R01-GM098151-01, the DOE SciDAC Institute of Scalable Data Management Analysis and Visualization DOE DE-SC0007446, NSF ACI-1339881, NSF IIS-1162013, and grants from the Knights Templar Eye Foundation and the March of Dimes Foundation.

References

[Ado14] ADOBE SYSTEMS INCORPORATED: *Photoshop Help*, 2014. https://helpx.adobe.com/photoshop.html. 3

[AJ09] ABEYSINGHE S., JU T.: Interactive skeletonization of intensity volumes. *The Visual Computer 25*, 5 (2009), 627–635. 3

[Aut13a] AUTODESK, INC.: *Autodesk Maya 2014 Documentation*, 2013. http://download.autodesk.com/global/docs/maya2014/en_us/. 3

[Aut13b] AUTODESK, INC.: *Autodesk Mudbox 2014 Documentation*, 2013. http://download.autodesk.com/global/docs/mudbox2014/en_us/. 3

[BJ01] BOYKOV Y. Y., JOLLY M.-P.: Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images. In *Proceedings of the 8th IEEE International Conference on Computer Vision (ICCV)* (2001), pp. 105–112. 6

[BKW08] BÜRGER K., KRÜGER J., WESTERMANN R.: Direct volume editing. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1388–1395. 3

[CCGR10] CHOWDHURY A. S., CHATTERJEE R., GHOSH M., RAY N.: Cell tracking in video microscopy using bipartite graph matching. In *Proceedings of the 20th International Conference on Pattern Recognition (ICPR)* (2010), pp. 2456–2459. 7

[CGCR13] CHATTERJEE R., GHOSH M., CHOWDHURY A. S., RAY N.: Cell tracking in microscopic video using matching and linking of bipartite graphs. *Computer Methods and Programs in Biomedicine 112*, 3 (2013), 422–431. 7

[CSS08] CHEN H.-L. J., SAMAVATI F. F., SOUSA M. C.: Gpu-based point radiation for interactive volume sculpting and segmentation. *The Visual Computer 24*, 7 (2008), 689–698. 3

[GPS89] GREIG D., PORTEOUS B., SEHEULT A.: Exact map estimation for binary images. *Journal of the Royal Statistical Society: Series B (Statistical Methodology) 51* (1989), 251–276. 6

[KOK*12] KWAN K. M., OTSUNA H., KIDOKORO H., CARNEY K. R., SAIJOH Y., CHIEN C.-B.: A complex choreography of cell movements shapes the vertebrate eye. *Development 139* (2012), 359–372. 1, 3

[ONI05] OWADA S., NIELSEN F., IGARASHI T.: Volume catcher. In *Proceedings of the Symposium on Interactive 3D Graphics and Games* (2005), pp. 111–116. 3

[ONI*08] OWADA S., NIELSEN F., IGARASHI T., HARAGUCHI R., NAKAZAWA K.: Projection plane processing for sketch-based volume segmentation. In *Proceedings of the 5th IEEE International Symposium on Biomedical Imaging: from Nano to Macro* (2008), pp. 117–120. 3

[RBD91] RIVEST J.-F., BEUCHER S., DELHOMME J.: Marker-controlled picture segmentation applied to electrical logging images. In *Proceedings of SPIE Nonlinear Image Processing II* (1991), pp. 179–190. 3

[RBD92] RIVEST J.-F., BEUCHER S., DELHOMME J.: Marker-controlled segmentation: an application to electrical borehole imaging. *Journal of Electronic Imaging 1*, 2 (1992), 136–142. 3

[RSB93] RIVEST J.-F., SOILLE P., BEUCHER S.: Morphological gradients. *Journal of Electronic Imaging 2*, 4 (1993), 326–341. 3

[Soi02] SOILLE P.: *Morphological Image Analysis: Principles and Applications*, second ed. Springr-Verlag, 2002. 3

[Wes99] WEST D. B.: *Introduction to Graph Theory*, second ed. Prentice Hall, 1999. 7

[WOCH09] WAN Y., OTSUNA H., CHIEN C.-B., HANSEN C.: An interactive visualization tool for multi-channel confocal microscopy data in neurobiology research. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1489–1496. 1

[WOCH12a] WAN Y., OTSUNA H., CHIEN C.-B., HANSEN C.: Fluorender: an application of 2d image space methods for 3d and 4d confocal microscopy data visualization in neurobiology research. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis)* (2012), pp. 201–208. 1

[WOCH12b] WAN Y., OTSUNA H., CHIEN C.-B., HANSEN C.: Interactive extraction of neural structures with user-guided morphological diffusion. In *Proceedings of the IEEE Symposium on Biological Data Visualization (BioVis)* (2012), pp. 1–8. 1, 2, 3

[WOH13] WAN Y., OTSUNA H., HANSEN C.: Synthetic brainbows. *Computer Graphics Forum 32*, 3-4 (2013), 471–480. 7

[YZNC05] YUAN X., ZHANG N., NGUYEN M. X., CHEN B.: Volume cutout. *The Visual Computer 21*, 8 (2005), 745–754. 3