Title
Visualization for Validation and Improvement of Three-dimensional Segmentation Algorithms

Authors
Weber, Gunther H.
Hendriks Luengo, Cristian Luis
Keränen, Soile V. E.
et al.

Publication Date
2005

Peer reviewed

EUROGRAPHICS - IEEE VGTC Symposium on Visualization (2005)
K. W. Brodlie, D. J. Duke, K. I. Joy (Editors)

Visualization for Validation and Improvement of Three-dimensional Segmentation Algorithms

G. H. Weber[1,2], C. L. Luengo Hendriks[2], S. V. E. Keränen[2], S. E. Dillard[1], D. Y. Ju[1], D. Sudar[2], and B. Hamann[1]

Berkeley *Drosophila* Transcription Network Project
[1] Visualization and Computer Graphics Research Group, Institute for Data Analysis and Visualization,
University of California, Davis, One Shields Avenue, Davis, CA 95616, U.S.A.,
{ghweber,sedillard,dyju,bhamann}@ucdavis.edu
[2] Genomics and Life Sciences Division, Lawrence Berkeley National Laboratory,
One Cyclotron Road, Berkeley, CA 94720, U.S.A.,
{GHWeber,CLLuengo,SVEKeranen,DSudar}@lbl.gov

Abstract

The Berkeley Drosophila Transcription Network Project (BDTNP) is developing a suite of methods that will allow a quantitative description and analysis of three dimensional (3D) gene expression patterns in an animal with cellular resolution. An important component of this approach are algorithms that segment 3D images of an organism into individual nuclei and cells and measure relative levels of gene expression. As part of the BDTNP, we are developing tools for interactive visualization, control, and verification of these algorithms. Here we present a volume visualization prototype system that, combined with user interaction tools, supports validation and quantitative determination of the accuracy of nuclear segmentation. Visualizations of nuclei are combined with information obtained from a nuclear segmentation mask, supporting the comparison of raw data and its segmentation. It is possible to select individual nuclei interactively in a volume rendered image and identify incorrectly segmented objects. Integration with segmentation algorithms, implemented in MATLAB, makes it possible to modify a segmentation based on visual examination and obtain additional information about incorrectly segmented objects. This work has already led to significant improvements in segmentation accuracy and opens the way to enhanced analysis of images of complex animal morphologies.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computing Methodologies/Computer Graphics]: Picture/Image Generation J.3 [Computer Applications/Life and Medical Sciences]: Biology and Genetics

1. Introduction

During the past few years, the genomes of an increasing number of species have been sequenced and published. While we are able to decipher the locations of the protein coding portions of many genes from genomic sequence information, the mechanisms that control when and where a gene is expressed are still poorly understood. The Berkeley *Drosophila* Transcription Network Project (BDTNP) is a multidisciplinary effort whose goal is to decipher how the regulatory information contained in DNA sequences directs the patterns of gene expression underlying animal development. Using the early embryo of the fruitfly *Drosophila melanogaster* as a model, the BDTNP is developing experimental and computational methods to systematically characterize and dissect the complex expression patterns and regulatory interactions that control the specialization of individual cells.

One of the major approaches the BDTNP is taking is to develop computational methods to describe and analyze 3D gene expression patterns. Typically, studies of animal gene expression patterns have employed visual inspection of photographic images of *in situ* hybridization experiments, which are two-dimensional by nature. Quantitative analysis of spatial gene expression patterns in *Drosophila* blastoderm embryos has only been employed for two-dimensional image data or one-dimensional profiles extracted from it [HWL02, KRS98, JSB*04]. These approaches do not capture 3D context and are difficult to use for quantitative mod-

eling. The BDTNP are developing ways to generate 3D matrices that describe the relative concentration of gene products (RNA or protein) in each cell. The aim is to produce an atlas of expression patterns of a large number of genes and to use this information in combination with other data sets to learn how to understand the extensive control mechanisms in genomic DNA and model the network. For example, by identifying clusters of genes with shared expression pattern features, it should be possible to determine if their promoter DNAs share similar sequence motifs.

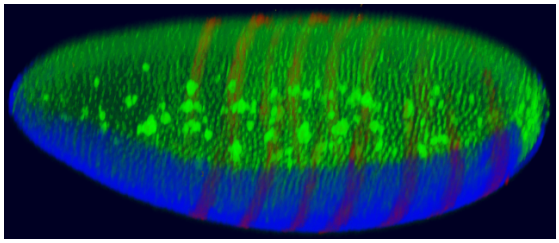

Figure 1: *Volume rendering of a whole Drosophila blastoderm embryo, stained for nuclei (green) and two genes (red and blue). Note the lower resolution along the optical axis and the attenuation with depth; artifacts inherent to the confocal microscopy approach. (Color versions of this figure and other figures are included on the color plate at the end of the proceedings.)*

To construct this atlas, the BDTNP have developed a pipeline to determine the location and extent of nuclei in embryos and measure the relative levels of mRNA expression in the cytoplasmic regions surrounding each nucleus (M. D. Biggin, C. C. Fowlkes, B. Hamann, S. V. E. Keränen, D. W. Knowles, C. L. Luengo Hendriks, J. Malik, D. Sudar, G. H. Weber, unpublished data). First, embryos are fluorescently stained to detect cell nuclei and the expression patterns of selected genes and imaged by Zeiss LSM 510 confocal microscope at a resolution adequate to distinguish individual nuclei. Per embryo, the resulting volumetric data sets consist of a stack of 120-140 image slices with a resolution of 1024x1204 pixels. Fluorescence intensities for the nuclear stain and the fluorescently labeled gene products are captured as separate scalar values per pixel. Figure 1 shows such a raw data set as volume-rendered image that highlights nuclei and expression patterns of two genes. Next, a set of dedicated algorithms is applied to the nucleus channel to obtain a segmentation mask, an image of equal size in which each pixel is labeled with the identifier (ID) of the nucleus to which it belongs (a value of zero indicates background). These automated algorithms are based on a modified watershed algorithm to delineate individual nuclei.

The accuracy of the resulting segmentation mask is key to obtaining useful expression level measurements, but several features of the system make the segmentation procedure challenging. The embryos imaged contain a monolayer of tightly packed nuclei, previously estimated to number somewhere between under 5,000 to over 6,000 that surround the yolk [HCO85]. The anisotropic resolution of the microscope, which is about three times lower along the optical z-axis than in the optical xy-plane, combined with the tight nucleus packing makes it very difficult to correctly segment individual nuclei in certain parts of the embryo. Frequently, several nuclei are incorrectly fused together in the segmentation mask.

Evaluating the correctness of the segmentation for a given image stack is difficult without proper interactive 3D visualization tools, and it is not possible to develop and verify the underlying segmentation algorithms without efficient means of visualizing segmentation results and assessing their quality manually. Consequently, we have developed a volume rendering approach that aids this task. Due to the high resolution of the slices, we use *bricking* to partition data sets in blocks that fit in texture memory. We also support skipping of blocks that contain unsegmented portions of an embryo, such as yolk, to speed up rendering. Rendering the nucleus channel using a gray-scale map makes it possible to visualize a segmentation mask by rendering each region in a different color. The combination of nucleus channel brightness information and color obtained from the segmentation mask allows one to compare a segmentation mask with the acquired raw data and to assay segmentation quality manually. A selection scheme for individual nuclei in the rendered image enables users to query additional information such as the segmentation identifier (ID) of an incorrectly segmented region. Extracting a surface via marching cubes (MC) method around a region with a specified segmentation ID provides additional means of evaluating segmentation quality. Segmentation information can also be used to modify the volume rendered image [TSH98, HMBG00]. Selecting a small number of nuclei and modifying color, brightness and opacity of only this subset only allows one to focus on particular regions in the segmentation while still rendering the remainder to provide context information. Integration with MATLAB (The MathWorks Inc, Natick, MA) and image processing algorithms allows a user to interactively correct the displayed volume and segmentation data. It is possible to perform an analysis that, for example, selects nuclei based on unusual size and subsequently highlight corresponding regions in the rendered image. Furthermore, set of regions can be selected manually and image processing algorithms that split or merge these regions executed locally on those regions of the rendered image.

The study of gene expression and morphology in complex organisms will increasingly move to computationally based image analysis strategies. The high complexity and dense packing of features in tissues and cells require sophisticated visualization tools that can manipulate and explore these 3D spaces. The tool we present here will facilitate analysis of such complex biological data.

2. Visualization of Segmented Data and Segmentation Masks

2.1. Related Work on Interactive Segmentation Tools

Previous work on segmentation interaction tools focused mainly on medical data. Kang et al. [KEK04] presented a system for correction of pre-computed segmentations. A morphology-based hole closing tool, a morphology-based point-bridging tool, and a spline-based surface dragging tool make it possible for a user to manipulate a segmentation. While these tools work in 3D, visualizations are restricted to (arbitrarily oriented) 2D slices. Volume rendering has only been used recently in conjunction with segmentation tools. Lefohn et al. [LKHW04] integrated volume rendering with segmentation based on level-set methods. Performing level-set computations on the graphics processor (GPU), makes it possible to visualize evolving level-set solutions at interactive rates. Sherbondy et al. [SHN03] used a region-growing-based approach where a user interacts with segmentation algorithms via seed selection.

2.2. Volume Rendering of Single-Channel Confocal Microscopy Data

The image slices of a confocal microscopy stack form a 3D scalar field data set specifying brightness information at each location. Confocal microscopy stacks usually consist of multiple, independent channels corresponding to different emission wavelengths. Each channel represents a separate registered scalar data set. We use direct volume rendering [Sab88] to visualize an individual channel. Brightness information is mapped to color and opacity information using an absorption and emission light model [Max95]. We use a 3D texture-based slicing approach [CCF94, VK96] that renders view-perpendicular slices to simulate ray casting. This approach allows us to utilize commodity graphics hardware to achieve interactive rendering.

To visualize segmentation results we render only the nuclei channel of a confocal image stack, which is used by our segmentation algorithms. (We are also developing a volume renderer for raw confocal microscopy images that supports visualization of up to three channels at the same time for quality control of acquired images.) We store volume data in a 3D texture (using trilinear texture interpolation) and the transfer function in a 1D texture (using linear texture interpolation). Transfer function look-up is implemented as a Cg [FK03] fragment program. Bricking allows us to handle data sets that are larger than texture memory. Volume data is split into bricks consisting of 32x32x16 samples that are rendered in back-to-front order. A slicing plane, implemented using an OpenGL clipping plane, allows users to "cut" into a volume-rendered image and examine the interior of a *Drosophila* blastoderm.

2.3. Volume Rendering of Segmentation Masks

For the display of segmentation masks we restrict transfer functions for the nuclei channel to gray-scale maps. Scalar values are mapped to brightness (usually using identity as transfer function) and opacity (usually using a linear ramp function, starting at a value around five for data quantized to eight bit). Segmentation masks are visualized by using the brightness information from the nuclei channel and choosing the color according to the nucleus ID in the segmentation mask, see Figure 2.

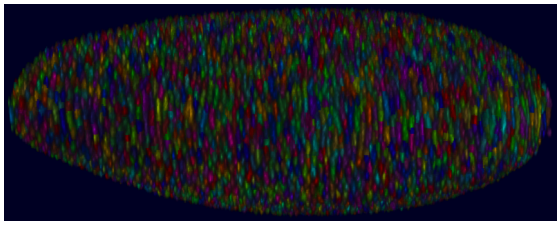

Figure 2: *Rendering of all nuclei of a Drosophila embryo with segmentation mask.*

In addition to an eight-bit texture used for the nuclei channel we load a segmentation mask into a second sixteen-bit 3D texture (using nearest-neighbor texture interpolation). Besides the transfer function stored as a 1D texture, we need a second texture that maps from segmentation IDs to colors. Sixteen-bit segmentation IDs require us to use a texture with $65536 = 2^{16}$ entries. This number exceeds the maximum resolution currently supported by commodity used PC graphics boards. (The used NVIDIA board, for example, supports 4096 entries. While the produced segmentation masks usually do not utilize all possible sixteen-bit IDs, they have IDs that exceed this resolution limit.) We solve this problem by storing segmentation colors in a 2D texture with a size of 256^2 yielding the required 65536 entries. A sixteen-bit ID is split into two eight-bit indices used as x- and y-coordinate for the 2D segmentation color map texture. Since graphics hardware and Cg do not support bit manipulation operations we simulate those with floating point operations:

```
segParam.x = modf(inSegment*256, segParam.y);
segParam.y /= 256;
```

During rendering, brightness information from the 1D transfer function texture (using the R component, assuming that a gray-scale transfer function assigns the same intensity to all color components) and color information from the 2D segmentation color texture are combined by multiplication. Alpha values of the transfer function and the segmentation color map are also multiplied, making it possible to hide individual nuclei from the volume-rendered image (by setting their opacity in the segmentation color map to zero) or to de-emphasize them (by setting their opacity to a value smaller than one).

Initially, a random color is used for each nucleus ID. A base color map provides a set of colors that the human eye can distinguish. Colors in the base color map are generated using the HLS color model. A fixed number of hues (16) and saturation levels (two) is chosen, and a corresponding number of colors (32) is generated by spacing hues equally on the color circle. For each hue equally spaced saturation levels are used, excluding a saturation of zero. (For example, when two saturation levels are used we choose saturation levels of 0.5 and one.) All base colors have an intensity of one, allowing for scaling with brightness information from the nuclei channel. Subsequently, all segmentation IDs are initialized by choosing a random color from this base map. Assigning segmentation colors randomly usually defines a good distribution of colors and allows one to distinguish neighboring nuclei by color. If this is not possible, a user can change the segmentation color for individual nuclei using the GUI.

2.4. Segmentation Mask-based Skipping of Empty Regions

When visualizing segmentation masks, only those portions of the volume that are segmented are of interest. These regions currently correspond to blastoderm nuclei and lie on an ellipsoidal shell. Its interior is mostly empty and contains only yolk and yolk nuclei that are currently of no interest and do not belong to the segmentation mask. By restricting volume rendering to regions that contain segmentation information, it is possible to save texture memory and accelerate rendering considerably, which is of particular importance for high-resolution data sets. We first create a "render mask," a bit set with one bit flag for each sample in the rectilinear grid indicating whether that sample belongs to a segmented region. Before uploading a brick to texture memory and rendering it, the segmentation mask is checked. If a flag corresponding to any sample is set, the brick is uploaded and rendered; otherwise it is skipped.

Using a brick size of 32^3, approximately 50% of all bricks can be skipped. If different dimensions are chosen that take the lower resolution along the optical axis into account, a larger number of bricks can be skipped. For example, when using a brick size of 64x64x16 approximately 60% can be skipped and when using a brick size of $32x32x16$, approximately 70% of bricks can be skipped. Choosing a smaller brick size increases the number of bricks that can be skipped, but increases overhead due to data duplication at brick boundaries. We found that a brick size of 32x32x16 usually yields good interactivity for data sets that we currently consider.

2.5. Selection in Volume-rendered Images

For a visualization tool for segmented data set to be useful, it is necessary to provide a user with means to obtain the ID of a segmented region in the rendered image. If, for example, a visualization of a segmentation mask shows that a particular nucleus is incorrectly segmented, it must be possible to determine its ID for later manipulation. We have implemented a scheme that supports selection of individual segmented nuclei in the volume-rendered image.

For selection we assume that the user is interested in the nucleus closest to the viewer. This nucleus corresponds to the "closest opaque region" (i.e., the closest region that exceeds a certain opacity threshold) in the volume-rendered image. Consequently, the region of interest is the last opaque region rendered. We apply a threshold to the opacity, rendering only samples with an opacity that exceeds 0.3. During volume rendering, a data set is rendered in back-to-front order. Using opacity thresholding, the sample that is rendered last for a particular pixel corresponds to the selected segmented region.

In order to determine the ID of the corresponding segmented region, we split the sixteen-bit ID into two eight-bit components (assuming an eight-bit framebuffer) and render those as two color components (red and green) into the frame buffer. Furthermore, we set the opacity of all samples that are rendered to one, ensuring that any previous ID is completely overwritten. Finally, we set the blue component of these samples to one, making it possible to determine whether a pixel is covered by any nucleus by testing this component. Using only floating-point operations these operations are expressed as:

```
r = (fmod(inSegmentID, 256) + 0.5) / 256;
g = (floor(inSegmentID / 256) + 0.5) / 256;
b = 1.0;
a = 1.0;
```

Once an image is rendered using this scheme, the segmentation ID for the first nucleus visible at a pixel can be computed as `r`256+`g`. We have found that this selection scheme works well in practice and allows a user to interact with a volume-rendered segmented data set effectively.

2.6. Segmentation Surfaces

To determine segmentation quality it is useful to draw a surface around a segmented nucleus. We use an MC method [MSS94] with minor modifications to extract such a segmentation surface. (In our implementation, a case table generated by implicit disambiguation [MSS94] avoids discontinuities in the extracted isosurface.) Since we are interested in a surface around a segmented region rather than an isosurface, we mark vertices as "inside" or "outside" the surface based on their segmentation ID rather than on the function values of the scalar field. All vertices with an ID equal to a specified ID are inside the segmentation surfaces, all others are outside.

Segmented nuclei are always connected regions in space. Thus, it is possible to avoid traversing all grid cells by starting the segmentation surface at a seed grid cell and grow-

ing it until it is complete. We store a list of so-called segmentation seeds in an array with 65536 entries. For each nucleus ID that exists in the segmentation, this array contains a reference to a grid cell with a vertex that is part of the segmented region with that index. Starting in this grid cell the segmentation surface is extracted based on the continuation method [WMW86]. Segmentation surface and volume-rendered image are displayed simultaneously. The segmentation surface is displayed first, with enabled depth-buffer writes. Second, depth buffer writes are disabled and view-perpendicular slices are rendered to display the volume data set.

2.7. Selective Rendering of Nuclei

In addition to rendering segmented nuclei in different colors we provide an alternate means of modifying the volume-rendered image based on the segmentation mask by maintaining a list of selected nuclei. Every time a user selects a segmented region (see Section 2.5) the ID of that nucleus is added to that list. Our tool supports the following rendering modes:

- Render the complete data set including regions that are not segmented. This mode disables skipping of bricks without segmented regions.
- Render only regions of the volume that have a valid ID in the segmentation mask. This rendering mode discards yolk from the rendered image and enables skipping unsegmented regions.
- Render only selected nuclei (i.e., nuclei in the selection list) using their assigned segmentation colors. All other nuclei are rendered in gray levels with their opacity and brightness scaled by user-specified factors. (By selecting a scaling factor of zero these nuclei can be completely hidden.) This is achieved by setting all color components in the segmentation mask to a gray level corresponding to the user-specified brightness and setting the segmentation mask opacity to the user-specified factor.
- Render all nuclei except those that are selected. This mode is complementary the previous mode and simply negates the selection of nuclei before rendering.

3. Interactive Control of Segmentation Algorithms

3.1. Integration with MATLAB

To facilitate control and verification of segmentation algorithms, we have integrated our segmentation renderer with MATLAB, which the BDTNP use to develop segmentation algorithms. Users can specify a set of MATLAB commands in a file, which are executed using MATLAB's external interface. These commands can access and modify three data blocks used by the volume renderer: (i) volume data corresponding to the nuclei channel that is segmented, (ii) segmentation mask, and (iii) the list of nuclei that are currently selected. For each command, it is possible to specify its name in the menu and the MATLAB command string. Boolean flags indicate whether volume data, the segmentation mask, or the nucleus list should be sent to MATLAB prior to the execution of the command, and whether this information should be replaced with modified data after the execution of the command.

The integration with MATLAB supports two main types of functionality:

- Improvement of Segmentation Algorithms. Iteratively perform a portion of the segmentation algorithm, displaying the result and allowing the user to modify that portion (using an external editor), without delays produced by transferring the large images between programs through files.
- Manual Modification of Segmentation Result. Our intention is to extend this method to generate a "perfectly" segmented image to be used as ground truth for further improvements of image segmentation and as a reference embryo in further analyses.

In addition, several other possibilities arise, due to the versatility of MATLAB and the integration with our tool. For example, it is possible to change the gray-value data being displayed to emphasize interesting parts. It is also possible to, e.g., automatically select nuclei based on location, size, shape, etc. and change the segmentation mask

3.2. Improvement of Segmentation Algorithms

Visual examination can provide valuable clues as to what changes in an algorithm are necessary to improve the results. Displaying horizontal 2D cross-sections is unsuitable to evaluate segmentation results. This is due to many factors, including the difficulty in visually detecting a change in gray-values as subsequent cross-sections are presented, and disalignment between the objects that are displayed and the orientation of the cross-sections. The latter, for our case, can easily be seen in Figure 3. The nuclei are elongated, but do not lie on the *xy*-plane. This causes neighboring nuclei in all directions to cut the same cross-section, which presents the user with an unnecessarily cluttered view. Ideally, the direction of the cross-section should change with the orientation of the object being studied, so that connectedness in the third dimension can easily be identified. 3D rendering mitigates this problem.

Being able to apply algorithms and display their results via the same user interface greatly streamlines the iterative process of identifying errors, changing the algorithm, and generating a new segmentation. Parameters in algorithms can be changed on the fly, or even custom routines can be implemented specifically for one special case, without recompiling the program or reloading data from external storage. Furthermore, MATLAB can produce plots displaying information and statistics about selected nuclei, fractions of nuclei, or clusters. Using MATLAB functions it is possible

to identify nuclei with particular properties, for example, all large nuclei in a segmentation mask. Manual counting of over-segmented nuclei and nuclei clusters marked as single nucleus supports interactive evaluation of segmentation results that is not possible without 3D visualization.

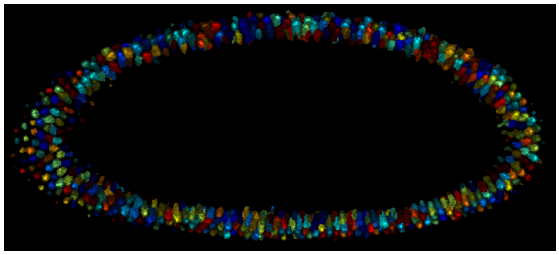

Figure 3: *Cross-section through a 3D segmented data set.*

3.3. Manual Modification of Segmentation Result

Improving a segmentation mask by hand produces a data set that can be extremely useful. The user can produce a segmentation close to a ground truth that is suitable for comparison with other results allowing, for example, to determine optimal values for certain algorithm parameters. The integration with MATLAB makes such an approach possible: a user can select two or more regions and merge them, split a single region into two or more, discard a region or create a new one.

4. Results

Using 3D visualization of segmentation results greatly enhances our ability to detect incorrectly segmented nuclei. Figures 4–6 show side-by-side comparisons of a 2D (generated using MATLAB) visualization and a 3D visualization (created with our new tool) of the same segmentation mask, where an incorrectly segmented nucleus is marked by a circle (the colors in the 2D and 3D display are not the same, since they are assigned randomly). In the 3D visualization in Figure 4 it can be seen that the marked yellow and green regions are actually one nucleus. In the 2D visualization, only one of these segmented regions is visible at the time. The other one lies on different z-planes. By moving back and forth through the stacks it is not possible to see that these two regions should be one.

Figure 5 shows the opposite effect: a group of nuclei seem fused because of the low z-resolution of the microscope, and therefore marked as a single nucleus. In the 3D view this is readily apparent, but in the 2D view one must take one region at a time and follow it through the various consecutive slices to note that it is elongated. Furthermore, the reason for it having that shape would elude the user.

Finally, in Figure 6 we show the use of the cutting plane to detect an erroneous region. In the 2D view it is inconspicuous because of the cluttered view presented. This view would be much clearer if it would cut the embryo perpendicular to its surface. Finding this correct orientation is trivial in the 3D renderer.

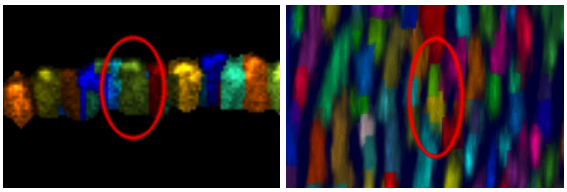

Figure 4: *Side-by-side comparison of 2D (left side) and 3D (right side) visualization of segmentation results, showing one nucleus marked as two separate regions*

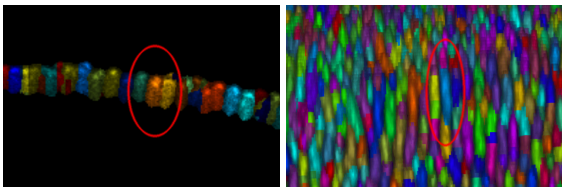

Figure 5: *Side-by-side comparison of 2D (left side) and 3D (right side) visualization of segmentation results, showing a couple of nuclei fused by the low z-resolution of the microscope.*

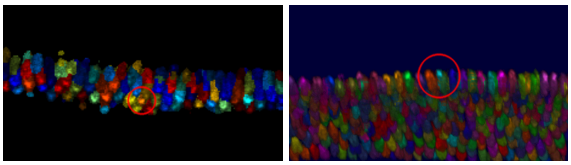

Figure 6: *Side-by-side comparison of 2D (left side) and 3D (right side) visualization of segmentation results, showing the use of the cutting plane to find an incorrect region.*

The integration with MATLAB allows modification of the data being displayed. This includes the segmentation mask as well as the input gray-value volumetric image, and to either enhance certain regions to draw the operator's attention to them, or to improve visualization to make evaluation less difficult. For example, when visualizing certain intermediate result of the segmentation process, it is very helpful to increase the size of the regions in the segmentation mask, as in Figure 7. At this stage of the process, the segmentation mask consists of one small marker for each nucleus, which is very hard to distinguish. In Figure 8, we show an example of changing the input volume data set. The user selected a feature caused by dust or something similar on the embryo, here marked with a red circle, and removed that feature from

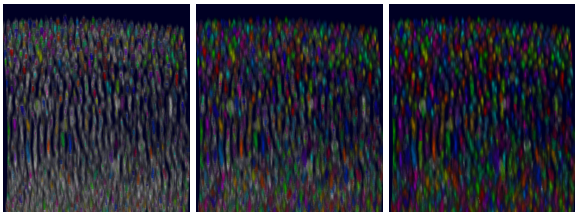

Figure 7: *Growing regions in segmentation mask to improve evaluation.*

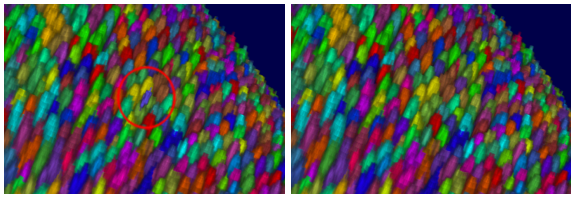

Figure 8: *Removing a feature in the input image.*

the input image. This might be used to, for example, clean up the microscope images.

As discussed earlier, the integration with MATLAB also allows editing the list of selected nuclei. This can be used to have MATLAB automatically select a set of nuclei that the user should pay closer attention to. In the example of Figure 9, MATLAB returned a list of large nuclei, which are rendered in color. The remaining nuclei are rendered with decreased brightness and opacity. While it is possible to hide de-emphasized nuclei completely, it is usually helpful to use them to provide users with a context of the embryo.

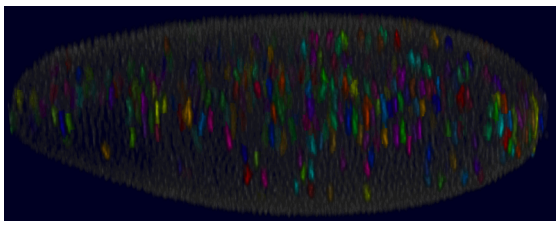

Figure 9: *Segmentation of Drosophila nuclei highlighting large regions.*

The second main reason to create the integration with MATLAB is to allow the user to manually correct errors in a segmentation. Figure 10 illustrates this. After an incorrectly segmented nucleus (marked with a red circle in the left image) has been identified by the user, a call to MATLAB can split it along its longest axis, changing the value of the segmentation mask in one of the halves to a new ID. Similarly it is possible to merge various selected regions by setting all pixels in those regions to the same ID.

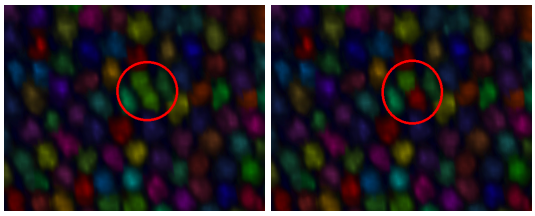

Figure 10: *Splitting an incorrectly segmented nucleus.*

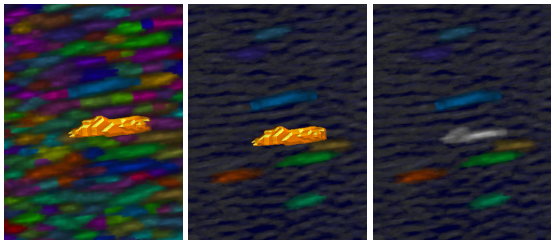

Figure 11: *Highlighting modes for a selected region.*

Figure 11 shows different methods of highlighting selected nuclei. In the left image, a segmentation surface (orange) is generated that surrounds the selected nucleus. In the middle image, only a set of nuclei is highlighted (drawn in color). All other nuclei are de-emphasized by rendering them in gray-scale and increasing their transparency. The currently selected nucleus is still surrounded by a segmentation surface. In the right image no segmentation surface is drawn. The currently selected nucleus is also drawn in gray scale. Since its transparency remains unchanged, it is still distinguishable from the de-emphasized nuclei.

5. Conclusions and Future Work

Volume visualizations of segmentation results convey information about segmentation quality that 2D visualizations do not. The examples in the Results section show clearly that two-dimensional cross-sections are unsuitable to show certain types of segmentation errors and evaluate segmentation quality. Volume visualizations of segmentation masks make it possible to detect these errors easily and correct them by integration with segmentation algorithms. As the BDTNP's goal is to image, segment, and analyze *Drosophila* embryos in a high-throughput pipeline, it is not feasible to correct each segmented embryo manually. Our segmentation mask visualization tool is mainly intended to aid in development of substantially more reliable segmentation algorithms. However, our tool can also be valuable in manual curation of acquired images and segmentation results. For example, we plan to use this tool to generate a set of reference embryos by manually correcting all errors in an automated segmentation.

Further research in visualization will be aimed mainly at developing tools for understanding derived 3D expression

data. The segmentation-based volume renderer will serve as the basis of such tools. By combining original raw data, segmentation masks, and derived data, it should become possible to produce a qualitative view of the raw data and, using our volumetric selection scheme, perform quantitative gene expression analysis using the derived data.

Studying even the relatively simple morphology of the *Drosophila* blastoderm requires advanced 3D visualization tools, due to the high information content of the 3D images. We plan to extend our tool to handle the morphology of more complex organisms such as later stages of *Drosophila* embryos. In the rapidly developing field of quantitative biology, extremely complex datasets such as these are becoming commonplace and the integration of biology with computational data analysis requires sophisticated visualization and interaction techniques to gain a true understanding of the underlying biological mechanisms.

6. Acknowledgments

This work was supported by the National Science Foundation under contracts ACI 9624034 (CAREER Award), through the Large Scientific and Software Data Set Visualization (LSSDSV) program under contract ACI 9982251, through the National Partnership for Advanced Computational Infrastructure (NPACI) and a large Information Technology Research (ITR) grant; the National Institutes of Health under contract P20 MH60975-06A2, funded by the National Institute of Mental Health and the National Science Foundation; the National Institutes of Health under contract 1R01 GM70444-01A1, funded by the National Institute of General Medical Science; by the Director, Office of Science, U.S. Department of Energy under contract DE-AC03-76SF00098; and the Lawrence Berkeley National Laboratory (LBNL) through a Laboratory Directed Research Development (LDRD) project. We thank Oliver Kreylos for his slice generation code used in the slicing-based volume renderer. We also thank the members of the Visualization and Graphics Research Group at the Institute for Data Analysis and Visualization (IDAV) at the University of California, Davis, and the members of the BDTNP at LBNL for their support.

References

[CCF94] CABRAL B., CAM N., FORAN J.: Accelerated volume rendering and tomographic reconstruction using texture mapping hardware. In *Proceedings of the 1994 Symposium on Volume Visualization* (New York, New York, 1994), ACM Press, pp. 91–98. 3

[FK03] FERNANDO R., KILGARD M. J.: *The Cg Tutorial*. Addison-Wesley, 2003. 3

[HCO85] HARTENSTEIN V., CAMPOS-ORTEGA J. A.: Fate-mapping in wild-type Drosophila melanogaster i. the spatio-temporal pattern of embryonic cell divisions. *Roux's Archives of Developmental Biology*, 194 (1985), 181–195. 2

[HMBG00] HAUSER H., MROZ L., BISCHI G.-I., GRÖLLER M. E.: Two-level volume rendering - fusing mip and dvr. In *IEEE Visualization 2000* (Los Alamitos, California, 2000), IEEE, IEEE Computer Society Press, pp. 211–218. 2

[HWL02] HOUCHMANDZADEH B., WIESCHAUS E., LEIBLER S.: Establishment of developmental precision and proportions in the early drosophila embryo. *Nature 415*, 6873 (February 14 2002), 798–802. 1

[JSB*04] JAEGER J., SUKOVA S., BLAGOV M., JANSSENS H., KOSMAN D., KOZLOV K. N., MANU, MYASNIKOVA E., VANARIO-ALONSO C. E., SAMSONOVA M., SHARP D. H., REINITZ J.: Dynamic control of positional information in the early drosophila embryo. *Nature 430*, 6997 (July 15 2004), 368–371. 1

[KEK04] KANG Y., ENGELKE K., KALENDER W. A.: Interactive 3d editing tools for image segmentation. *Medical Image Anaylsis 8*, 1 (2004), 35–46. 3

[KRS98] KOSMAN D., REINITZ J., SHARP D. H.: Automated assay of gene expression at cellular resolution. In *Pacific Symposium on Biocomputing* (1998), pp. 6–17. 1

[LKHW04] LEFOHN A., KNISS J. M., HANSEN C. D., WHITAKER R. T.: A streaming narrow-band algorithm: Interactive deformation and visualization of level sets. *IEEE Transactions on Visualization and Computer Graphics 10*, 40 (7 2004), 422–433. 3

[Max95] MAX N. L.: Optical models for volume rendering. *IEEE Transactions on Computer Graphics 1*, 2 (1995), 99–108. 3

[MSS94] MONTANI C., SCATENI R., SCOPIGNO R.: A modified look-up table for implicit disambiguation of marching cubes. *The Visual Computer 10*, 6 (1994), 353–355. 4

[Sab88] SABELLA P.: A rendering algorithm for visualizing 3D scalar fields. *Computer Graphics (Proceedings of ACM SIGGRAPH 88) 22*, 4 (1988), 51–58. 3

[SHN03] SHERBONDY A., HOUSTON M., NAPEL S.: Fast volume segmentation with simulataneous visualization using programmable graphics hardware. In *IEEE Visualization 2003* (Los Alamitos, California, 2003), IEEE, IEEE Computer Society Press, pp. 171–176. 3

[TSH98] TIEDE U., SCHIEMANN T., HÖHNE K. H.: High quality rendering of attributed volume data. In *IEEE Visualization 1998* (Los Alamitos, California, 1998), IEEE, IEEE Computer Society Press, pp. 255–262. 2

[VK96] VAN GELDER A., KIM K.: Direct volume rendering with shading via three-dimensional textures. In *1996 Volume Visualization Symposium* (New York, New York, Oct. 1996), Crawfis R., Hansen C., (Eds.), ACM Press, pp. 23–30. 3

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structure for soft objects. *The Visual Computer 2* (1986), 227–234. 5

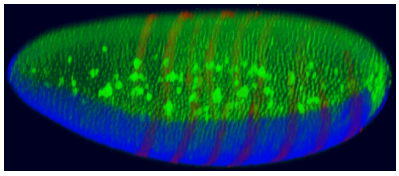

Figure 1

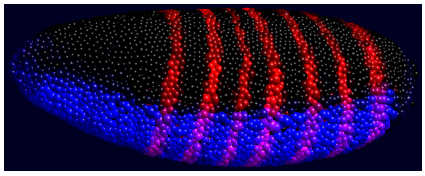

Figure 2

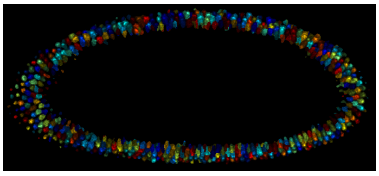

Figure 3

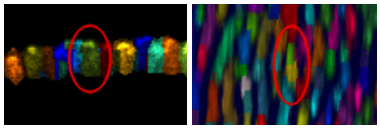

Figure 4

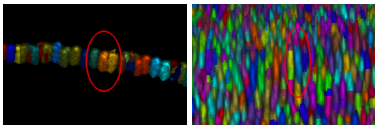

Figure 5

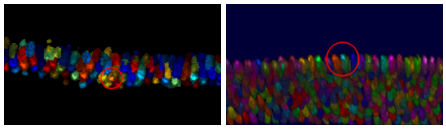

Figure 6

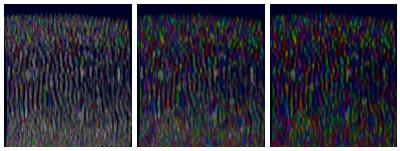

Figure 7

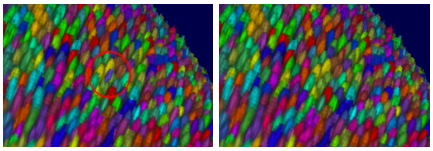

Figure 8

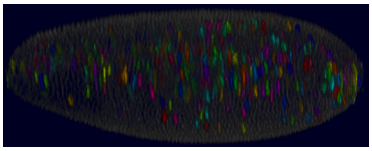

Figure 9

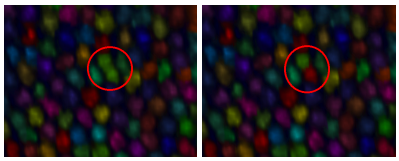

Figure 10

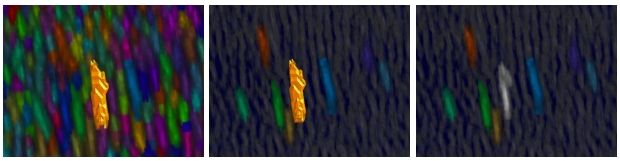

Figure 11