Visualization and Analysis of Digital Light-Sheet Microscopy

Luke Peeler

7 August 2014

Abstract

The movement of cells during embryonic development is an important part of understanding how tissues and organs function in complex organisms. The nerves within nervous system are examples of structures that depend on cell migrations in early development. Early neurological development of zebrafish, mice, and humans includes the migration of facial branchiomotor neurons (FBMNs), an identifiable group of motor neurons that migrate tangentially in mammals and fishes, from their origin in the rhombomere (r4) of the zebrafish hindbrain to their final destination. There are many unanswered questions about the movement of the first FBMN, the pioneer neuron, that may be answered with computational analysis of light-sheet microscopy imaging. Light-sheet microscopy is the only modality that images the entire organism at cellular resolution in three-dimensions, for the time-span over which migration occurs. Visualization and analysis of the light-sheet data will be facilitated by the use of Diderot, a new parallel domain-specific language.

Contents

Chapter 1

Introduction

The study and understanding of cell behavior in organisms at various stages of development is a central goal of developmental biology [1]. In particular, the dynamic behavior and motion of cells during embryonic development hold important keys to a greater understanding of how more fully developed systems, organs, and tissues function. Microscopy of embryonic development and resulting image analysis and visualization are thus indispensable to arriving at a more complete model of cellular dynamics. Of special interest is the tracking of cells through time to discern their provenance and, if possible, to construct a model of their lineage.

Digital light-sheet microscopy (DLSM) is a new, effective technology that has improved performance and image analysis results over other methods, such as confocal and point-scanning two-photon microscopy [1]. Light-sheet microscopy greatly reduces the exposure of a specimen to laser light by only illuminating a single slice at any given time. This process allows for a complete 3D image of a specimen to be quickly acquired with a high signal-to-noise ratio, while reducing damaging effects, such as photo-bleaching. Imaging of organisms using this technology, especially during embryonic development, affords researchers with high-quality 3D+t data sets with which to perform useful analyses, both qualitative and quantitative in nature.

A key aspect of cellular dynamics is concerned with understanding the origins and development of entire lineages of cells. Developing such models allows researchers to understand how complex systems grow and develop from a single cell or group of cells and lends critical insight into the processes by which certain genes are expressed and how, for example, a

given organism might develop predispositions to contract various diseases. Cell tracking is a useful tool in pursuit of such models. Various methods exist which allow researchers to trace the development of individual cells through time in order to ascertain their origins and migratory paths.

The quantitative information afforded by models derived from the results of cell tracking algorithms are complemented by a qualitative view of organismal development offered by 3D volume renderings. Using such visualizations, researchers are able to more effectively contextualize the development of particular systems against the development of the organism as a whole. Moreover, given a sufficiently high temporal sampling rate, 3D volume renderings present a useful high-level view of organismal development through which important developmental events, such as gastrulation, may be studied.

Computational analysis will be facilitated by the use of Diderot, a parallel domain-specific language (PDSL) designed for image analysis and visualization. Diderot offers support for quickly developing and prototyping code suited to image analysis by providing a high-level model of computation and concise notation for expressing image analysis and visualization algorithms [3].

In this paper, I aim to present useful visualization and analysis results using the Teem software and Diderot programming language as a proof-of-concept that can be refined in future work. I will focus on the tangential migration of "pioneer" facial branchiomotor neurons (FBMNs) in zebrafish, a cell type only recently described [10]. During development of vertebrates, all neurons perform radial migration outwards from their birthplace close to the central lumen of the neural tube, but some, such as the FBMNs, also undergo tangential migration, which is important for the proper development of the brain [7, 2, 10]. Visualizations and analysis of the pioneer FBMN and its path during the early stages of embryonic development will be complemented by 3D volume renderings that will offer a high-level view of organismal motion and individual cellular dynamics at varying time points.

Chapter 2

Related Work

The promise of achieving a more complete understanding of cellular dynamics and developmental processes in early development of organisms through light-sheet microscopy and various computational methods, such as cell tracking, has contributed to a growing body of work in which model organisms, such as the zebrafish, fruit fly, and mouse are imaged and studied.

In order to learn more about the movement and cellular dynamics of epiblast cells in mouse embryos during gastrulation, imaging using light-sheet microscopy and computational analysis, including cell tracking, have been performed, with methodology and software made freely available [4].

Moreover, due to its effectiveness and flexibility, light-sheet microscopy has been used for imaging of specimens at varying spatial and temporal scales. This has facilitated the study of many developmental processes, such as gene expression and cellular dynamics, particularly migration. The adaptability of light-sheet microscopy to varying spatial scales may be seen from observations of individual molecules in the salivary gland tissue of *C. tentans* larvae to the imaging of wild-type and mutant zebrafish for 24 hours, at the embryo level [9].

In the latter case, researchers leveraged the high spatial and temporal resolution provided by light-sheet microscopy to generate 'digital embryos', which include quantitative information related to cell position, migratory paths, and division patterns. Digital embryos were computed using automated image segmentation, and the information gleaned from this analysis is to be used in further models of the mechanical forces contributing to

morphogenesis, which constitutes an important step toward a more complete model of early organismal development [5].

Chapter 3

Methods

Analysis and visualization will be performed via two methods, each serving distinct research needs. The important insight provided by understanding the provenance and migratory pathways of cells may be addressed from both a qualitative and quantitative perspective using cell tracking methods. Visualizations which clearly delineate cellular dynamics through time and space offer researchers a high-level view from which to formulate conceptual frameworks governing early organismal development. Additionally, the quantitative data provided by cell tracking methods facilitates more precise modeling though further quantitative analysis. 3D volume renderings are complementary to the results of cell tracking methods, as such visualizations assist researchers in contextualizing the information provided they provide.

Cell tracking methods' aim, in essence is to recover the motion of cells between frames of a 2D+t or 3D+t dataset. There are a number of useful algorithms employed for this purpose, such as cross-correlation methods, deterministic methods, and Bayesian methods [8]. For our purposes and image data, cross-correlation between individual frames seems well-suited. Cross-correlation is also known as a sliding dot product or sliding inner-product. In essence, the image at time $t + 1$ is analyzed against the image at time t using cross-correlation to produce values corresponding to the degree of overlap, and predicting the most likely location of the cell at the later time step is performed using least-squares fitting against such values–specifically through minimization of the χ^2 function, described below. This method suits our data particularly well due to the high signal-to-noise ratio provided by light-sheet microscopy and the fluorescence of the neurons.

Assuming our image is of the following form:

$$I_c(\vec{x}) = \sum_{n=1}^{N} I_p(\vec{x} - \vec{x}_n(t); D, \ldots),$$

where N is the number of particles and

$$I_p(\vec{x}; D, \ldots)$$

is a function describing the shape of an idealized particle, with its center at the origin. The center of such an idealized particle is the point at which its intensity is highest.

The most likely location of a particle can be found using least-squares fitting. We define:

$$\chi^2(\vec{x_0}; D, w) = \int W(\vec{x} - \vec{x_0})[I(\vec{x}) - I(\vec{x} - \vec{x_0}; D, w)]^2 \, d\vec{x},$$

where $W(\vec{x} - \vec{x_0})$ is some weight function. We minimize χ^2 when $\vec{x_0}$ is at the position of some particle, allowing us to locate its position. We extend this process to all particles and find several minima of χ^2 using convolution.

We expand from above:

$$\chi^2(\vec{x_0}; D, w) = \int W(\vec{x} - \vec{x_0})[I(\vec{x})^2 - 2I(\vec{x})I(\vec{x} - \vec{x_0}; D, w) + I(\vec{x} - \vec{x_0}; D, w)^2] \, d\vec{x}$$

giving us

$$\chi^2(\vec{x_0}; D, w) = I^2 \otimes W - 2I \otimes (WI_p) + \langle WI_p^2 \rangle$$

where $\otimes$ and $\langle \cdot \rangle$ are defined as follows:

$$f \otimes g = [f \otimes g](\vec{x_0}) = \int f(\vec{x}) g(\vec{x} - \vec{x_0}) \, d\vec{x}$$

$$\langle f \rangle = 1 \otimes f$$

For simplification, when $W = 1$,

$$\chi^2(\vec{x_0}; D, w) = \int I^2 \, d\vec{x} - 2I \otimes I_p + \langle I_p^2 \rangle$$

We can see that $I \otimes I_p$ will achieve a maximum at the positions of the particles, reflecting the greatest degree of overlap. When $I \otimes I_p$ is at a maximum, we see that χ^2 achieves a minimum, reflecting the minimal error achievable for least squares fitting.

The following code performs cross-correlation on a frame-by-frame basis for 3D image data.

```
// TODO: add code here
```

Volume renderings offer researchers an informative, high-level view from which to observe organismal development through time and study important developmental milestones, such as gastrulation. Moreover, image data that is sufficiently well sampled spatially provides researchers the opportunity to view embryonic development on the scale of individual cells, offering potential clues into the mechanisms and forces which influence important processes, like cell differentiation and gene expression.

Maximum intensity projection (MIP) is one effective means of generating volume renderings for biological image data. As opposed to direct volume rendering techniques, which require each sample of the image data to be assigned an opacity and color by the transfer function, MIP projects out the voxels of the highest intensity that are intersected by rays traced from the viewpoint to the projection plane. The assignment of opacity and color to these voxels aims to give the appearance of a 3D image, despite being a 2D projection.

The following Diderot code generates an image using the MIP method:

```
// volume dataset
field#0(3)[] FF0 = tent  image("vol-0.nrrd");
field#0(3)[] FF1 = tent  image("vol-1.nrrd");

// set camera and image parameters

vec3 camEye = [-5586.59, 7053.3, -22285.7];
vec3 camAt = [2354.57, 1866.76, 511.564];
vec3 camUp = [-0.566504, -0.824, 0.00986904];
real camNear = -700.0;
real camFar = 700.0;
real camFOV = 5.02;
```

```
int imgResU = 600;
int imgResV = 430;

real rayStep = 2;

// boilerplate computation of camera and light info
real camDist = |camAt - camEye|;
real camVspNear = camNear + camDist;
real camVspFar = camFar + camDist;
vec3 camN = normalize(camAt - camEye);
vec3 camU = normalize(camN  camUp);
vec3 camV = camN  camU;
real camVmax = tan(camFOV*/360.0)*camDist;
real camUmax = camVmax*real(imgResU)/real(imgResV);

// RayCast(ui,vi) computes rendered color for pixel (ui,vi)
strand RayCast (int ui, int vi) {
    real rayU = lerp(-camUmax, camUmax, -0.5, real(ui), real(imgResU)-0.5);
    real rayV = lerp(-camVmax, camVmax, -0.5, real(vi), real(imgResV)-0.5);
    vec3 rayVec = (camDist*camN + rayU*camU + rayV*camV)/camDist;

    real rayN = camVspNear;
    real mip0 = 0.0;
    real mip1 = 0.0;
    output vec3 rgb=[0,0,0];

    update {
       vec3 pos = camEye + rayN*rayVec;
       if (inside (pos,FF0)) {
          mip0 = max(mip0, FF0(pos));
          mip1 = max(mip1, FF1(pos));
       }
       if (rayN > camVspFar) {
          stabilize;
       }
       rayN = rayN + rayStep;
```

```
    }

    stabilize {
       rgb=[mip1, 0.7*mip0, 0.5*mip1];
    }
}

initially [ RayCast(ui, vi) | vi in 0..(imgResV-1), ui in 0..(imgResU-1) ];
```

First, the image is read in and stored in-memory. Other initialization parameters set up the viewing space of the image, such as camera/viewer position, viewer direction, and viewer orientation. Additionally, the view space is specified by the positions of the near and far clipping planes and the field-of-view.

From here, the initial position of each ray with respect to the viewing plane (coordinates U and V) is determined by linearly interpolating with respect to the viewing space and image resolution. Then, each ray is traced using a small step size through the viewing space, approaching the far clipping plane. At each step, the image data is sampled, and the intensity maxima are iteratively updated. Upon reaching the far clipping plane, a final computation is made to assign an RGB color value based on the maximum intensity values encountered by the ray during the tracing process.

Another effective means of volume rendering, initially presented by Marc Levoy, involves a similar process of ray casting, but computes opacity and shading of a voxel in independent steps, combining them in the image plane using back-to-front compositing [6]. The opacity assigned to a voxel by the transfer function seeks to avoid aliasing artifacts by assigning voxels near the chosen isovalue to have an opacity that scales inversely with the magnitude of the local gradient vector. The opacity is assigned as below:

$$\alpha(\vec{x_i}) = \alpha_v \cdot \begin{cases} 1 & \text{if } |\bigtriangledown f(\vec{x_i})| = 0 \\ & \text{and } f(\vec{x_i}) = f_v \\ 1 - \frac{1}{r}\left|\frac{f_v - f(\vec{x_i})}{|\bigtriangledown f(\vec{x_i})|}\right| & \text{if } |\bigtriangledown f(\vec{x_i})| > 0 \\ & \text{and } f(\vec{x_i}) - r \cdot |\bigtriangledown f(\vec{x_i})| \leq f_v \leq f(\vec{x_i}) + r \cdot |\bigtriangledown f(\vec{x_i})| \\ 0 & \text{otherwise} \end{cases}$$

where α_v is the chosen opacity, f_v is the chosen isovalue, and r is the desired thickness in voxels.

Shading is computed using the Phong shading method, which is an interpolation technique that interpolates surface normals across a surface locally and computes pixel colors using these normals and a reflection coefficient based on the angle at which a specified light vector strikes the surface.

The following Diderot code implements this volume rendering method:

```
// volume dataset
field#1(3)[] FF0 = bspln3  image("vol-0.nrrd");
field#1(3)[] FF1 = bspln3  image("vol-1.nrrd");

// set camera and image parameters

input bool camOrtho = false;
vec3 camEye = [-5586.59, 7053.3, -22285.7];
vec3 camAt = [2354.57, 1866.76, 511.564];
vec3 camUp = [-0.566504, -0.824, 0.00986904];
real camNear = -600.0;
real camFar = 700.0;
real camFOV = 5.02;
int imgResU = 600;
int imgResV = 430;



real rayStep = 1.5;
vec3 lightVsp = normalize([-2.0, -5.5, -2.0]);

// boilerplate computation of camera and light info
real camDist = |camAt - camEye|;
real camVspNear = camNear + camDist;
real camVspFar = camFar + camDist;
vec3 camN = normalize(camAt - camEye);
vec3 camU = normalize(camN  camUp);
vec3 camV = camN  camU;
```

```
real camVmax = tan(camFOV*/360.0)*camDist;
real camUmax = camVmax*real(imgResU)/real(imgResV);
vec3 light = lightVsp[0]*camU + lightVsp[1]*camV + lightVsp[2]*camN;

input real isoval = 1500;
input real thick = 20;
function real alpha(real v, real g) = 0.1*(1.0 if v > isoval else clamp(0, 1, 1 -
    |v-isoval|/(g*thick)));

// RayCast(ui,vi) computes rendered color for pixel (ui,vi)
strand RayCast (int ui, int vi) {
    real rayU = lerp(-camUmax, camUmax, -0.5, real(ui), real(imgResU)-0.5);
    real rayV = lerp(-camVmax, camVmax, -0.5, real(vi), real(imgResV)-0.5);
    vec3 rayVec = camN if camOrtho
                    else (camDist*camN + rayU*camU + rayV*camV)/camDist;
    vec3 rayEye = (rayU*camU + rayV*camV if camOrtho else [0,0,0]) + camEye;

    real rayN = camVspNear;
    vec3 rgb = [0, 0, 0];
    output vec4 rgba=[0,0,0,0];
    real transp = 1;

    update {

       vec3 pos = rayEye + rayN*rayVec;

       if (inside (pos,FF0)) {

          real val1 = FF1(pos);

          vec3 grad1 = - FF1 (pos);
          real aa = alpha(val1, |grad1|);

          if (aa > 0) {
              aa = 1 - pow(1-aa, rayStep);
              real depth = lerp(1, 0.2, camVspNear, rayN, camVspFar);
```

```
            real shade = lerp(0, 1, -1, normalize(grad1)light, 1);
            vec3 color = [0.4,1.0,0.4];
            rgb += transp*aa*depth*shade*color;
            transp *= 1 - aa;
         }

      }


      if (transp < 0.01) {
         transp = 0;
         stabilize;
      }

      if (rayN > camVspFar) {
         stabilize;
      }
      rayN = rayN + rayStep;
   }

   stabilize {
     rgba = [rgb[0], rgb[1], rgb[2], 1-transp];
   }

}

initially [ RayCast(ui, vi) | vi in 0..(imgResV-1), ui in 0..(imgResU-1) ];
```

As in the MIP code, the image is loaded into memory and initialization parameters are set to define the viewing space of the image. Ray positions are then interpolated, and each ray is traced according to the step size through the viewing space. The image data is sampled at each step, and the image value and local gradient are used to compute opacity, `aa`, using the method described above, with the isovalue having been chosen by visual inspection using a range of input values for our particular data.

Opacity is then recomputed to be weighted in terms of the step size of the ray. From

here the depth of the ray, relative to the near and far clipping planes, is computed and Phong shading computation is performed. The RGB color assigned to the given voxel is computed and compositing is performed. The `transp` value is decreased by a factor of `1-aa` to facilitate the next iteration's compositing computation, reflecting the increasing opacity of the voxels as the ray continues to penetrate the volume of interest. Tracing of a given ray may terminate upon passing below the given transparency threshold or upon exiting the view space.

Chapter 4

Results

... Results ...

Chapter 5

Future Work

... What worked, what didn't ...

In addition to studying the cellular dynamics of the pioneer FBMN, understanding the provenance of all FBMNs during early embryonic development is of particular value to developmental biologists. One important method to support this work is to construct an abstract model of cell lineage. Such a tree-based model could include, at each node, information about position, time, stage of mitosis, etc. The abstract nature of this analysis would enable biologists to study the development of multiple organisms from multiple image acquisitions without needing to observe and decipher the raw image data, enabling them to identify common developmental patterns and possible anomalies in early development.

Chapter 6

Appendix A - `czireader` software

In addition to the image analysis and visualization work presented above, significant effort in pursuit of these results was devoted to the design of an efficient file reader for the `.czi` file format specified by Carl Zeiss Microscopy, GmbH, the manufacturer of the digital light-sheet microscope from which the image data originated.

Though not of particular interest in terms of the results sought after by developmental biologists, work on this file reader presented an opportunity to apply various skills gained during my coursework as an undergraduate in the Department of Computer Science at The University of Chicago. Various aspects of file format reading, such as file I/O; binary representation of data and bitwise operators; and cache performance learned from a myriad of systems courses informed my approach when helping to develop this software. Furthermore, I gained better familiarity with object-oriented design concepts and facility with the C++ programming language through my coursework as a student in the department's Bx/MS program, both of which proved invaluable.

This software was designed with the goal of giving researchers a fast, efficient tool by which to extract image data from `.czi` files, without having to rely on much larger software packages, such as ImageJ, which are designed for a variety of uses. Moreover, we hope to make the `czireader` software available as an open-source project, as there is a great deal of extensibility possible. One such example is outputting image data in formats supported by a variety of tools for image analysis and visualization. The `.nhdr` file extension used by the Teem software package is currently supported, and there are countless similar opportunities for expansion.

Bibliography

[1] Fernando Amat and Philipp J Keller. Towards comprehensive cell lineage reconstructions in complex organisms using light-sheet microscopy. *Development, Growth & Differentiation*, 55(4):563–578, 2013.

[2] Anand Chandrasekhar. Turning heads: Development of vertebrate branchiomotor neurons. *Developmental Dynamics*, 229(1):143–161, Jan 2004.

[3] Charisee Chiw, Gordon Kindlmann, John Reppy, Lamont Samuels, and Nick Seltzer. Diderot: A parallel dsl for image analysis and visualization. In *Proc. Programming Language Design and Implementation (PLDI)*, pages 111–120. ACM, 2012.

[4] Takehiko Ichikawa, Kenichi Nakazato, Philipp J Keller, Hiroko Kajiura-Kobayashi, Ernst H K Stelzer, Atsushi Mochizuki, and Shigenori Nonaka. Live imaging and quantitative analysis of gastrulation in mouse embryos using light-sheet microscopy and 3d tracking tools. *Nat. Protocols*, 9(3):575–585, 03 2014.

[5] Philipp J. Keller, Annette D. Schmidt, Joachim Wittbrodt, and Ernst H.K. Stelzer. Reconstruction of zebrafish early embryonic development by scanned light sheet microscopy. *Science*, 322(5904):1065–1069, 2008.

[6] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics & Applications*, 8(5):29–37, 1988.

[7] Oscar Marín and John L. R. Rubenstein. A long, remarkable journey: Tangential migration in the telencephalon. *Nature Reviews Neuroscience*, 2(11):780–790, 2001.

[8] Jens Rittscher, Raghu Machiraju, and Stephen T. C. Wong. *Microscopic Image Analysis for Life Science Applications*. Artech House, Inc., Norwood, MA, USA, 1 edition, 2008.

[9] Raju Tomer, Khaled Khairy, and Philipp J Keller. Shedding light on the system: studying embryonic development with light sheet microscopy. *Current Opinion in Genetics & Development*, 21(5):558–565, 2011. Developmental mechanisms, patterning and evolution.

[10] Sarah J. Wanner and Victoria E. Prince. Axon tracts guide zebrafish facial branchiomotor neuron migration through the hindbrain. *Development*, 140(4):906–915, 2013.