The thing with the Golgi apparatus

Gert-Jan Both

Supervised by:
P. Sens
C. Storm

Technical university of Eindhoven
January-November 2018

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nam et turpis gravida, lacinia ante sit amet, sollicitudin erat. Aliquam efficitur vehicula leo sed condimentum. Phasellus lobortis eros vitae rutrum egestas. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Donec at urna imperdiet, vulputate orci eu, sollicitudin leo. Donec nec dui sagittis, malesuada erat eget, vulputate tellus. Nam ullamcorper efficitur iaculis. Mauris eu vehicula nibh. In lectus turpis, tempor at felis a, egestas fermentum massa.

Contents

1

Introduction

2
Introduction

3
Data processing pipeline

In this chapter I present the work done on processing the rush movies. Several preprocssing steps hav been undertaken to improve the quality of the fit, and we present all here. Roughly, we can divide the process in four steps:

1. Segmentation and creation of masks
2. Denoising of movies
3. Calculation of spatial and temporal derivatives
4. The actual fitting

Below we describe each step separately.

3.1 Step 1: Segmentation

The images obtained from the rush experiments often contain multiple cells. Furthermore, we can also segment the image into roughly three different types: 1) the background, where nothing of interest happens. No cells are present here, 2) the cytoplasm, which is the area where we want to fit our model and 3) the Golgi itself, where we do not necessarily want to fit. Unfortunately, no bright field images were available, making segmentation significantly harder, as no clear cell boundary can be observed. Further complicating the story is the large dynamic range of the movies due to the fluorescence concentrating in the Golgi. The following procedures we present have been developed to deal with these problems. Note that they are empirical methods, i.e. there's no theoretical background as to why they *should* work. However, in practice they do and I haven't found any other method which was able to.

3.1.1 Voronoi diagram

This method is based on a technique called Voronoi tesselation and doesn't depend on ny measure of the intensity. It was developed after noting that since the cargo is spread throughout the ER in the first few frames and as the ER is roughly circumnuclear, we can use this to determine the centre of the cell (roughly). Voronoi tesselation then allows us to divide the frame into areas with just one point per area, i.e. one cell per area (theoretically). More precise, given n coordinates, voronoi tesselation divides the given area into n pieces, where every point in a piece is closest to one coordinate. In practice this means for us that each point in a cell area is closest to its the given cell centre. Figure **ref** shows this. Each calculated cell centre is a red point and the lines depict the borders between each voronoi cell. Assuming the cells don't move too much, they don't cross the cells and thus we apply the voronoi diagram calculated in the first few frames to the entire movie.

3.1.2 Intensity

For the fi

3.1.2.1 Golgi

3.1.2.2 Cytoplasm

3.2 Denoising

3.2.1 Wavelet filter

3.2.2 Proper orthogonal decomposition

3.3 Derivatives

Taking spatial and temporal derivatives of these images is not an entirely trivial operation due to the discreteness of the system. More specifically, taking numerical derivatives of data is extremely hard to do properly and becomes even harder in the presence of noise. Next to basic finite difference methods, one can for example use a linear-least-squares fitted polynomial, smoothing spline or a so-called tikhonov-regularizer **ref needed**. Each method comes with its strengths and weaknesses, but one particularly nasty thing for our context is that they don't scale well to higher dimensions and quickly become computationally expensive.

Another issue related to discretization is the size of the grid w.r.t. the size of the features. To see this, we plot a 2D-gaussian with $\sigma = 1$ in figure **ref**.

As expected, the derivative is normal to the isolines of the object. Now consider the discretized version of the object. Taking the naive spatial derivate w.r.t. to each direction means only considering a single row or column of and taking the derivative in that direction. Figure **ref** shows the result of this operation. An artifact is clearly visible: instead of a nice uniform derivative, we see a 'cross'. This effect is a cause of the discretization grid being too large for some smaller, often bright, objects.

To remedy this, one can for example artificially upscale the grid, interpolate the values inbetween, and take the derivates from this grid. This is not ideal however, since the upscaling requires a large amount of memory and is computationally expensive. Another solution which is common in image processing is applying a *kernel operator*. The advantage of a kernel operator is that it is extremely computationally cheap, as it involves convolving the original picture with a differentiation kernel. The differentiation kernel is an approximate version of a finite difference scheme. We use and show here the Sobel filter, which is the most commonly used one.

In a simple finite central difference scheme, we set

$$\frac{dx}{dt} \approx \frac{x_{i+1} - x_{i-1}}{2h}$$

where h is the distance between two points. In terms of a kernel operator, this would look like (the h drops out as the distance in terms of pixels is 1):

$$\frac{1}{2} \cdot \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

And applying it by convoluting it to a matrix gives the x-derivative:

$$\partial_x A \approx A * \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

and analogous for the y-direction. However, as we've seen, looking at just a single row introduces cross-like artifacts. To remedy this, we wish to include diagonal pixels as well. However, the distance between the diagonal pixels and the center pixel is not 1 but $\sqrt{2}$ and furthermore we need to decompose is it into $\hat{x}$ and $\hat{y}$, introducing another factor $\sqrt{2}$. Thus, one obtains the classis 3×3 Sobel filter **ref**:

$$\mathbf{G}_x = \frac{1}{8} \cdot \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \mathbf{G}_y = \frac{1}{8} \cdot \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Although not extremely accurate, the Sobel filter seems to do the tricks for us. Several other versions such as Scharr or Prewitt exist, offering several benefits such as rotational symmetry, but we have not pursued these. They just change the coefficients. Although we have shown a 3×3 filter here, the filter can take into account higher order schemes such as a 5×5 or 7×7. The major benefit of the spatial derivatives as a convolution operator is its computational efficiency: convolutional operations are performed parallel and are extremely fast.

For the time derivative, we apply a second order accurate central derivative scheme, while for the spatial derivatives (both first and second order) we apply the 5×5 Sobel filter.

4
Conclusion

Appendix 1: Some extra stuff

Add appendix 1 here. Vivamus hendrerit rhoncus interdum. Sed ullamcorper et augue at porta. Suspendisse facilisis imperdiet urna, eu pellentesque purus suscipit in. Integer dignissim mattis ex aliquam blandit. Curabitur lobortis quam varius turpis ultrices egestas.

5

References