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

Abstract i

1 Introduction

1.1 Quantitative work on the Golgi so far . . . . . .

1.2 RUSH system . . . . . .

2 Introduction

3 Data processing pipeline

3.1 Step 1: Segmentation . . . . . .

3.2 Step 2 - Denoising . . . . . .

3.3 Step 3 - Derivatives . . . . . .

3.4 Step 4 - Fitting . . . . . .

4 Results data analysis

4.1 Analysis of time derivatives . . . . . .

4.2 Analysis of fit . . . . . .

5 Physics Informed Neural Networks

5.1 Neural Networks . . . . . .

5.2 Physics Informed Neural Networks . . . . . .

5.3 Conclusion . . . . . .

6 Conclusion

Appendix 1: Some extra stuff

7 References

1

Introduction

1.1 Quantitative work on the Golgi so far

1.2 RUSH system

1.2.1 Man II

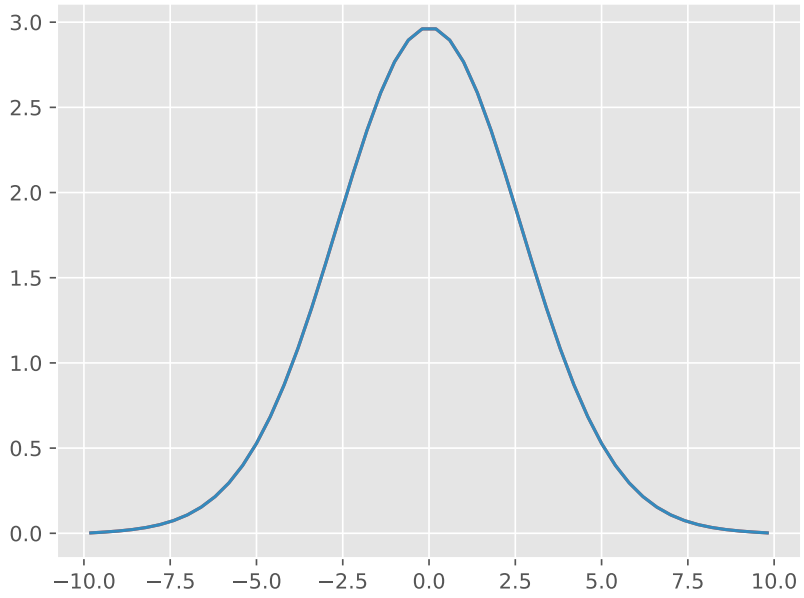

Figure 1.2.1: text

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 fitting however we wish to make a slightly better approach than a voronoi diagram. As stated, we can't find the exact delineation of the cell, but looking at the intensity, we can see an 'area' of interest, separating background from the cell. Since the Golgi is quite bright in het last 200 or so frames, we consider only the intensity for the Golgi, while for the cytoplasm we consider both the intensity and its time derivative. Thus we have two analog but different processes. For the Golgi we do the following:

1. Renormalize the concentration C between 0 and 1.
2. Sum all frames. One then obtains an image such as figure **ref**

$$x \sum_{frames} C(x, y, t)$$

3. This image is thresholded, either through an otsu threshold or a manual one, until the mask roughly matches what we want. Note that extreme precision isn't required, since we just want the rough area. THis results in figure **ref**

For the cytoplasm we follow the same procedure only now we take the log of sum of the product of the intensity and its time derivative:

$$\log \left(\sum_{frames} C(x, y, t) \cdot \partial_t C(x, y, t) \right)$$

We thus obtain a complete mask for the movie as shown in figure **ref**

3.2 Step 2 - Denoising

In order to accurately calculate the derivatives and generally improve the quality of fitting, we wish to denoise and smooth the obtained movies. Denoising and

smoothing is a subject about which many books have been written and there are hundreds of approaches. One oft-used technique is to Fourier transform the signal, cutoff all coefficients above a cutoff frequency and retransform back into the real domain. Next, a Savitzky-Golay filter can be used to finally smooth the result. However, a big issue with all these methods is their non locality. Since our movies have different scales, this is a big problem. Furthermore, they often smooth out sharp peaks. After evaluating several methods, I have settled on a relatively new method presented in **ref**.

The so-called WavinPOD method combines two well-known filtering techniques, known as wavelet filtering and Proper Orthogonal Decomposition. Below we explain each separately. Our explanation is adapted from **ref** and **ref**.

Wavelet filter

A wavelet filter is not really the appropriate name, as its more of a transform.

Proper orthogonal decomposition

Proper orthogonal decomposition is a technique similar to what is known as Principal component Analysis in statistics and falls into the general category of model reduction techniques. It's often used in flow problems to extract coherent structures from turbulent flows. Simply put, in POD we wish to express a function as

WavinPOD

WavinPOD combines these two techniques in the following way. First, we decompose our problem with a POD transformation. This yields a set of temporal and spatial modes. We select the most energetic modes and wavelet filter these, before transforming them back to the real domain. As shown in **ref**, combining these techniques has an advantage over others.

In our case, we select the number of modes to be used by hand (30 in the case of MANII) and apply a 3-level db4 wavelet. We use a slightly higher than necessary level to increase smoothness. In the figure below we show the result for both a pixel in time and one time snapshot. Note that the result is significantly smoother, but that smaller details have been preserved.

3.3 Step 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. We analyze these in the next chapter

3.4 Step 4 - Fitting

Now that we have gathered all our data we can use it to fit. We use a simple least squares method.

4

Results data analysis

Here we present the results from our analysis on the RUSH experiments. We only show the results of MANII because this is the only thing we studied.

textwidth: 5.00098in

linewidth: 5.00098in

4.1 Analysis of time derivatives

Ziet er goed uit chefke.

4.2 Analysis of fit

4.2.1 Diffusion

4.2.2 Advection

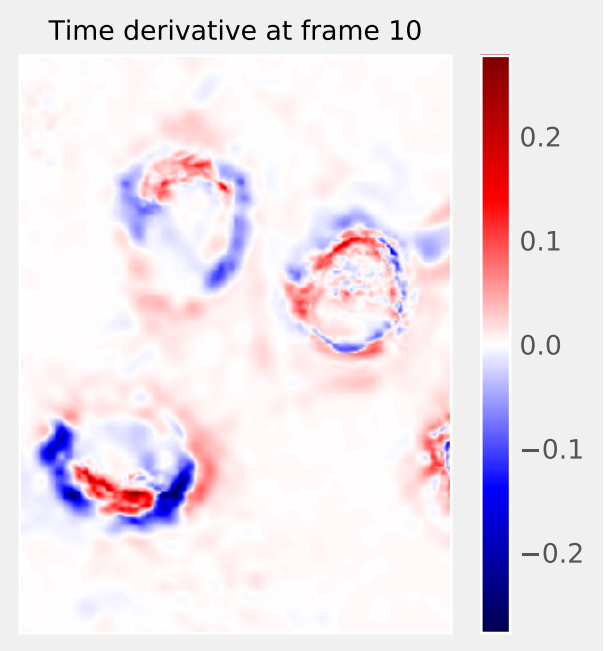

Figure 4.1.1: Wolla zieke subscript.

5

Physics Informed Neural Networks

In previous chapters we showed the difficulty in finding a general fit for a model to spatiotemporal data. Very recently, a set of papers introduced a new technique called Physics Informed neural networks. The set of papers show very powerful and promising results. I've evaluated this technique for use in fitting our model to the RUSH data. Since for many Neural networks are a new technique, this chapter is also a gentle introduction into Neural networks themself. We have three parts:

- Neural Networks
- Physics Informed Neural networks
- Conclusion

In neural networks we introduce the ideas and some mathematics behind neural

networks in Physics informed neural networks we showcase the concept using a simple toy model and in conclusion we present the evaluation of the technique.

5.1 Neural Networks

5.1.1 The basis: the neuron

The basic of each type of neural network is the same: the neuron. Inspired, but not the same as a biological neural network, the neuron has an input x and transforms it to an output y: $x \to y$. This is a three step process. The input x is multiplied by a factor a known as a weight and adds a constant known as the bias b. This serves as an input to the *activation function* g of the neuron, so that

$$y = g(ax + b)$$

the role of the activation function is to introduce non-linearity into the system. Many research is ongoing into different activation, but one of the most used is the $tanh(x)$ function, due to its biological relevance and bounded in and outputs. Several other functions with more fsvourable properties such as lower computtational cost are available as well.

A neuron always has one output, but can have multiple inputs, for example in the case of multitple neurons feeding into another one. One can also have multiple neurons in a single layer. A natural generalization is that a and b become matrices and x and y vectors:

$$\vec{y} = g(A\vec{x} + B)$$

where g is applied element wise. The goal of training a neural network is to find the matrices A and B which make sure that y is closest to the desired outcome.

5.1.2 Network

5.1.3 Training

5.2 Physics Informed Neural Networks

5.2.1 The concept

In my point of view, the power of physics informed neural networks comes from two core concepts: the physics regularization and automatic differentiation.

####*

5.2.2 PINNs in practice

5.2.2.1 Constant D

5.2.2.2 Varying D

5.3 Conclusion

5.3.1 Weak points and how to improve

6

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.

7

References