Application of Multiple-Object Tracking Algorithms
To accurately trace the trajectories of magnetized cortical vesicles in the Neuron

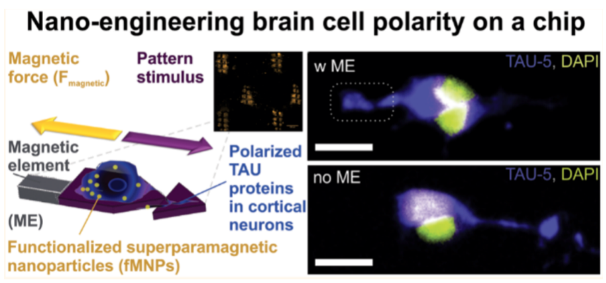
June 8, 2016

1 Abstract

Object-tracking is a well studied problem that has many applications in different domains such as Biology, Traffic, and Security. However, the field continues to be studied because each of these domains and sub-domains have unique features that pose different obstacles and drivers for the development of the object tracking algorithms. This report focuses in the Biological domain, of developing, testing, and comparing efficient methods of tracking multiple cortical vesicles in the neuron. The report is specifically interested in further developing the well-known Bayesian and Kalman filter, to tackle the obstacles of object collision, unpredictable vesicle movement, and noisy vesicle detection.

2 Introduction

In the brain, pathologically oriented neurons are often the cause for disordered circuits, severely impacting motor function, perception, and memory. Current research is focusing on developing highly paralyzed nanomagnets on a chip to exert local mechanical stimuli on cortical neurons. This magnetic field directed displacement repositions the cortical neurons against their preferred positions, in a way that has the potential to restore disordered neural circuits.


Nano-engineering brain cell polarity on a chip
Magnetic force ($F_{magnetic}$)
Pattern stimulus
Magnetic element
(ME)
Polarized TAU proteins in cortical neurons
Functionalized superparamagnetic nanoparticles (fMNPs)
w ME
no ME
TAU-5, DAPI
TAU-5, DAPI

In research, it is common to analyze particle trajectories using Mean Squared Displacement, describing the average extent of space explored by a particle as a function of time:

$$MSD = \langle (x - x_0)^2 \rangle = \frac{1}{T} \sum_{t=1}^{T} (x(t) - x_0)^2$$

In order to compute the Mean Squared Displacement, and evaluate other properties of the magnetic forces being applied to the corresponding cortical neurons, it is essential that we have a tracking algorithm that is able to efficiently detect the vesicles of interest and track their movement from frame to frame. We have obtained video samples of live cell experiments preformed on-chip, with and without magnet application. The cortical vesicles were labeled with a fluorescent tag, and imaged using a wide-filed fluorescent and phase contrast in a life cell incubator on top of an inverted fluorescent microscope. There are many highly developed object tracking algorithms out there; but in this report, we will explore two particular algorithms, the Bayesian and Kalman filter, and fine tune them for our biologicial application. We explore the different adjustments that need to be made to these algorithms in order to overcome the situation specific obstacles of object collision, unpredictable vesicle movement, and noisy vesicle detection. All algorithms have been documented in Matlab.

The report is organized as follows. Section 3 is a description of the core structure and foundation of common object tracking methods. Section 4 outlines the current obstacles that we are facing, and proposes solutions to the current obstacles. Section 5 contains visual examples of the algorithms' performance, illustrating the parameter costs and optimal vesicle movement characteristics for each algorithm. Section 6 concludes with a summary of the pros, cons, and optimal scenarios for each algorithm.

3 Foundations of Object Tracking: Nearest Neighbor Method

In this section, we will explain the basics of the Nearest Neighbor Algorithm, a simple but effective multiple object tracking method. The nearest neighbor method is the foundation of our research, and we will be building on to it in order to address the obstacles presented in the next section.

The Nearest Neighbor Algorithm can be broken down into two major elements, particle detection, and particle matching.

3.1 Particle Detection

Particle detection is the fully automated process of locating the desired particles in a set of files. In our experiments, the particles that we are interested in tracking are cortical vesicles tagged with fluorescence, meaning they are brighter than the rest of the image.

There are different ways to isolate these vesicles, including template matching, hough transform to parse out the circular elements, but our detection algorithm primarily uses the Laplacian of the Gaussian (LoG) filter. The LoG filter is a simple filter in Matlab that essentially takes second order derivatives over the image in order to detect edges of vesicles in each frame. The LoG Filter, sometimes called a Mexican hat

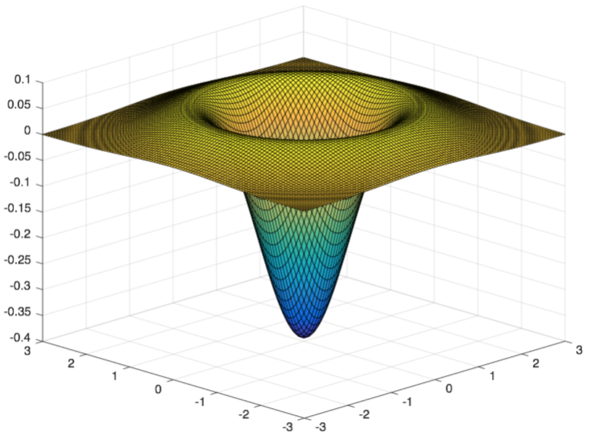


Figure 1: A visualization of the LoG filter template to be convoluted with the image.

filter, can be fine-tuned by adjusting the size (width) and sigma value (breadth of peak). The wider and broader the template is, the less affected by noise the result will be because it operates over a wider area.

After getting the edges of vesicles, we threshold the image and use morphological techniques to record unique points at the locations of corresponding vesicles.

Special morphological techniques can be applied if we know that the vesicles are generally spaced out from one another. These techniques include filling in holes, bridging gaps, closing regions, and combining selected areas that are close in pixel distance.

3.2 Particle Matching

Once we have obtained a list of vesicle coordinates in each frame, particle matching aims to link vesicles from one frame, to their corresponding position in the next frame. This can be very tricky when there are multiple objects of interest in each frame, and if the objects do not move in a predictable manner. The basic framework for particle matching, that we will build on throughout the report, is as follows:

Algorithm 1: Nearest Neighbor Multiple Object Tracking

```
   Input: $Q_{est}, Q_{det}, thresh_1, thresh_2$
   Output: set of trajectories for every detected vesicle
1  $Q_{est} \leftarrow$ set of detections from first frame
2  $Q_t \leftarrow$ set of detections from t'th frame
3  for $t = 1$ to $t = lastframe$ do
4      Make a distance cost matrix between $Q_{est}$ and $Q_t$
5      Use Munkres Algorithm to do the assignment
6      If the assigned detection is farther than $thresh_1$ from the observation, reject it.
7      Check 1: Add any particle in $Q_t$ that does not get assigned (new tracking) to the end of $Q_{est}$
8      Check 2: Give a strike to any particle in $Q_{est}$ that does not get matched up.
9      if A particle has a number of strikes is greater than $thresh_2$ then
10         Remove the particle from $Q_{est}$
11     Save results in $Q_{Trajectories}$
12 return $Q_{Trajectories}$
```

It is important to note that Munkres Algorithm (Algorithm 1, line 5) is a famous combinatorial optimization algorithm that solves the lowest-cost assignment problem in polynomial time. Munkres Algorithm outputs one-to-one assignments that will result in the lowest aggregate distance. We also note that the parameters $thresh_1$, and $thresh_2$, allow for some adjustments so that the Algorithm can perform optimally in different scenarios.$Thresh_1$ controls the rejection of Munkres Algorithm matches, so high values are optimal in cases where there are large particle jumps between frames. $Thresh_2$ controls for the number of frames a particle can go without being matched to a detection. $Thresh_2$ should be higher if the detection algorithm does not produce reliable results.

4 Current Obstacles and Proposed Solutions

This Nearest Neighbor method (Algorithm 1) works well for simple multiple-object tracking problems, but vesicle tracking is a domain with additional constraints that need to be addressed. In this section, we will explore and propose solutions to the three major constraints posed by our cortical neural vesicle tracking problem: Noisy Particle Detection, Unpredictable Vesicle Movement, and Vesicle Collisions.

4.1 Noisy Particle Detection

The floursecent vesicles in the neuron images we have obtained are not constant. Their brightness fluctuates, as some vesicles are shown as much brighter than others. Furthermore, the vesicles are of different sizes and sometimes come close to each other to appear as one. Due to optics properties, we observe bright rings around the vesicles that are further below the surface of a cell after imaging. These discrepancies in conjunction with the fact that the laplacian of the gaussian filter is not 100 percent reliable in detecting vesicles in each frame, leads to noisy particle detection. To ease this obstacle, we impose a Boundary Condition.

The boundary approach allows us to ignore the noisy particles that may be mis-detected throughout the image, only focusing on the particles that were detected in the first frame and particles that are inside our area of interest. We assume here that new particles can only be generated inside our area of interest.

The boundary approach consists of two main steps: Determining the boundary, and making sure that new particles can only be added to the algorithm if they lie within the boundary.

Step 1: Read in the image, and determine a threshold (LEVEL pararmeter) based on the intensity of the area of interest. **Step 2**:Using Matlab morphology tools, we convert the image to black and white based on this threshold **Step 3**:Connect any of the selected segments that are within a desired distance for smoothness (DIST parameter). This leaves us with one continuous set of boundary points in the form of a vector of coordinates. **Step 2**:Using the Matlab "inpolygon" function, we add a condition to the algorithm to only accept a particle as new if it is inside the boundary. This can simply be added to Check 1 in Algorithm 1.

We find that adding this boundary condition increases the accuracy of trajectories by lessening the number of false vesicle matches, allowing us to maintain a higher value of $thresh_1$.

4.2 Vesicle Collisions

Vesicles sometimes move very close to each other in our cell observations, and appear to be detected as one vesicle. This proposes a dilemma because the current algorithm uses a one-to-one matching scheme.

A possible solution to this is to add a final check after the one-to-one matching to see if an un-matched vesicle is close enough to a vesicle that has already been matched. This way, two particles have the ability to be matched to the same detection in the occurrence of a collision. We update the Algorithm as follows, adding in a new parameter, $thresh_3$ to control the rejection threshold of acceptable collision matches:

Algorithm 2: NEAREST NEIGHBOR MULTIPLE OBJECT TRACKING WITH COLLISION FIX

```
Input: $Q_{est}, Q_{det}, thresh_1, thresh_2, thresh_3$
Output: set of trajectories for every detected vesicle
1  $Q_{est} \leftarrow$ set of detections from first frame
2  $Q_t \leftarrow$ set of detections from t'th frame
3  for $t = 1$ to $t = lastframe$ do
4      Make a distance cost matrix between $Q_{est}$ and $Q_t$
5      Use Munkres Algorithm to do the assignment
6      If the assigned detection is farther than $thresh_1$ from the observation, reject it.
7      If a particle in $Q_{est}$ does not get matched up, but is less than $thresh_3$ from a detection, assign it
       correspondingly.
8      Check 1: Add any particle in $Q_t$ that does not get assigned (new tracking) to the end of $Q_{est}$
9      Check 2: Give a strike to any particle in $Q_{est}$ that does not get matched up.
10     if A particle has a number of strikes is greater than $thresh_2$ then
11         Remove the particle from $Q_{est}$
12     Save results in $Q_{Trajectories}$
13 return $Q_{Trajectories}$
```

4.3 Unpredictable Vesicle Movement

The final obstacle that we face is the obstacle of unpredictable vesicle movement. In a domain such as traffic, car movement is much more predictable because we can reasonably assume that a car will not start moving backwards on a through street. However, this is not the case with vesicles, as there are many forces acting on the vesicles causing them to move in extremely unpredictable directions. Furthermore, each vesicle is unique and has its own particular degree of unpredictable movement!

Therefore, it is hard to use methods to predict the next position of a vesicle (such as the kalman filter), which assume a specific type of constant movement and acceleration. When using the Kalman Filter, the algorithm would use past matches to predict an exact location for where the vesicle should be in the next frame. We then matched up these predictions to the vesicles that were detected in the next frame using the Hungarian Algorithm. However, since we have vesicle movement that is very unpredictable for the Kalman Filter, we run into problems.

Although the direction of movement is unpredictable, we can gain some predictive insight from the speed at which a vesicle is traveling. We take inspiration from the Bayesian tracking algorithms, and develop Algorithm 3 based on probability distributions. We essentially create individual probability distributions for each particle across the entire image. These distributions will tell you the different likelihoods of the particle moving to each pixel of the image. We find that the simplest way to do this is to model the distributions as a Multivariate Gaussian Distribution, with variances specifically determined for each vesicle according to the vesicle's recorded velocity from the previous frame.

When the velocity increases, so does the velocity and the particle will have a flatter Gaussian Distribution.

In order to implement this, we will need a new variable Vel that stores the velocities of every vesicle in the Q_{est} vector of x and y coordinates of vesicles. The velocity will represent our variance term, which we can then use to determine our diagonal covariance matrix , K, for each particle that will help us determine its unique probability distribution.

We define $K(i)$ as a 2x2 covariance matrix for the i-th particle being tracked in Q_{est} as :

$$K(i) = \begin{pmatrix} Vel(i)^2 & 0 \\ 0 & Vel(i)^2 \end{pmatrix}$$

Because we are assuming Multivariate Gaussian Distributed movement, we can calculate the probability that the point $Q_{est}(i)$ is matched to the point $Q_{det}(j)$ by the following equation.

If we define $Dist(i,j)$=$Q_{det}(j)$-$Q_{est}(i)$:

$$P(match) = \frac{1}{\sqrt{(2\pi)^2 det(K(i))}} e^{\frac{-1}{2} Dist(i,j) K^{-1} Dist(i,j)}$$

In algorithm 3, at every frame, we can create a cost matrix with probabilities (instead of distances), and match up each particle with the detection that has the highest probability of being the next particle position. Here, the Bayesian Approach helps us to better track particles that switch directions unexpectedly, because the Gaussian Distribution is spread in all directions. Furthermore, recording the velocity of the particles helps the program better predict particles that have been stationary vs. moving quickly.

Algorithm 3: PROBABILITY MULTIPLE OBJECT TRACKING

```
   Input: $Q_{est}, Q_{det}, Vel, thresh_1, thresh_2, thresh_3$
   Output: set of trajectories for every detected vesicle
1  $Q_{est}$ ← set of detections from first frame
2  $Q_t$ ← set of detections from t'th frame
3  $Vel$ ← vector of ones (initial velocity) the size of $Q_{est}$
4  for $t = 1$ to $t = lastframe$ do
5      Use $Vel$ to calculate the velocity matrix for each particle
6      Make a probability cost matrix between $Q_{est}$ and $Q_t$, assuming Gaussian distribution
7      Use Munkres Algorithm to do the assignment
8      If the assigned detection is farther than $thresh_1$ from the observation, reject it.
9      If a particle in $Q_{est}$ does not get matched up, but is less than $thresh_3$ from a detection, assign it
       correspondingly.
10     Check 1: Add any particle in $Q_t$ that does not get assigned (new tracking) to the end of $Q_{est}$, and
       increase the size of $Vel$
11     Check 2: Give a strike to any particle in $Q_{est}$ that does not get matched up.
12     if A particle has a number of strikes is greater than $thresh_2$ then
13         Remove the particle from $Q_{est}$
14     Save results in $Q_{Trajectories}$
15 return $Q_{Trajectories}$
```

5 Experiments

In this section, we test Algorithm 2 and Algorithm 3's ability to accurately track particle trajectories on various video samples of live cell data. We will begin by documenting the particle detection step, and optimal parameters found. Then, we will experiment with the particle tracking algorithms 2 and 3, recording their threshold values and performance. These experiments together will give us insight on the circumstances that each algorithm performs well, and where they still need improvement.

5.1 Particle Detection Analysis

We will use the particle detection methods described in section 3.1 to locate the vesicles in our data samples. In figure 2, we can see that with size=4 and sigma=0.43, the LoG filter recognizes the areas tagged with a red circle as vesicles of interest. For a comparison, we also show the less accurate results of using a hough transform in blue. We observe that the LoG filter does a good job of detecting the bright vesicles that have the most contrast with the background elements. However, when the vesicles start to merge together, it is harder for the LoG filter to differentiate the vesicles. After experimenting with different filter parameters, we find that a size parameter of 4 and a sigma parameter of 0.43 is able to provide the most accurate detections. Figure 3 shows us how sensitive the LoG filter is when we change the sigma and size parameter values. Since we expect to see around 2.5 average vesicles detected per frame for this data sample, we can use these plots to pinpoint the optimal sigma and size values of 0.43 and 4, respectively.

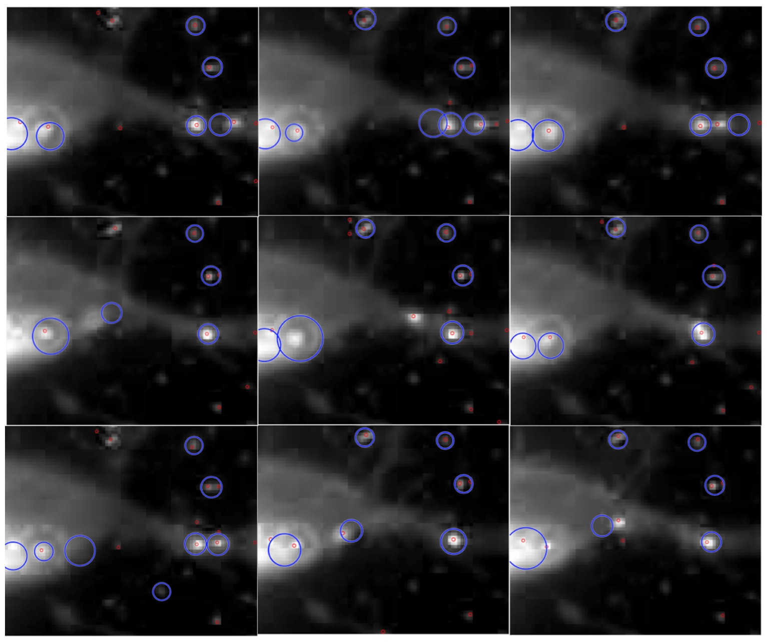
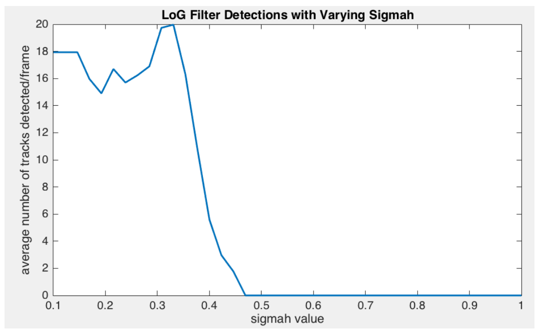
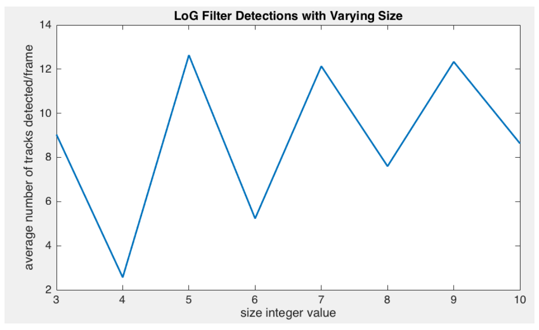


Figure 2: Figures illustrating the different particle detection methods used on the cortical vesicles in the neuron video. The LoG filter (size=4 and sigma=0.43) recognizes the areas tagged with a red circle as a vesicle, while the Hough Transform recognizes the areas tagged with a blue circle as a vesicle.



(a) This plot illustrates the number of average detections/frame at varying sigmah levels for the LoG Filter, holding size constant at its optimal value of 4.



(b) This plot illustrates the number of average detections/frame at varying size values for the LoG Filter, holding sigmah constant at its optimal level of 0.43.

Figure 3: LoG Filter Analysis

5.2 Experiment 1

Our first test is on the original video file of cortical vesicles in a neuron, in its natural state without any magnetic force field application. This data sample was the driving force of our Algorithm development, as it presented the three obstacles of Noisy Particle Detection, Vesicle Collisions, and Unpredictable Vesicle Movement. In order to address the Noisy Particle Detection, we implemented the boundary condition approach as described in Section 4.1.

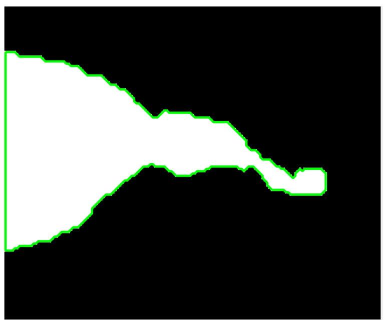


Figure 4: The green boundary outlines the cell, our area of interest, after thresholding the image in Matlab as outline in section 4.1. Using LEVEL=0.3 and DIST=2.

This video sample also contains a vesicle collision (reference figure 3), and therefore does not form complete trajectories if tracked with Algorithm 1 that uses one-to-one particle matching. We therefore use Algorithm 2 that addresses vesicle collisions, outlined in section 4.2, and see improved results.

Figure 4 shows us the improvement of our updated Algorithm 2. We can see that in Algorithm 1, the pink trajectory jumps over the area of the collision, while in Algorithm 2, the pink trajectory follows the correct path. Because this video sample had extremely fast movements between frames, we have to use very high threshold values. In order for complete trajectories to be formed, we need to have a threshold value that is essentially as large as the largest particle jump between frames. Furthermore, since this fast movement is actually very close to the collision, we need our collision parameter, $thresh_3$, to be large as well. Any value lower than 25 for $thresh_3$ would result in an incompleted pink trajectory. Because this specific video sample

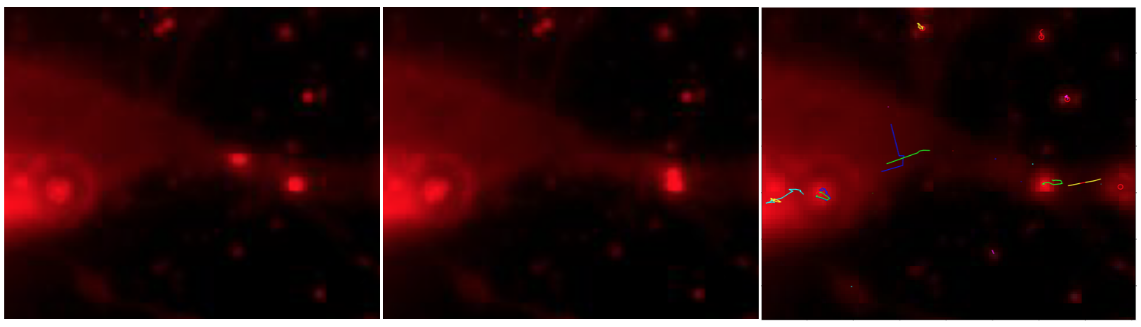


Figure 5: An illustration of a vesicle collision, and the corresponding break in trajectory when using Algorithm 1.

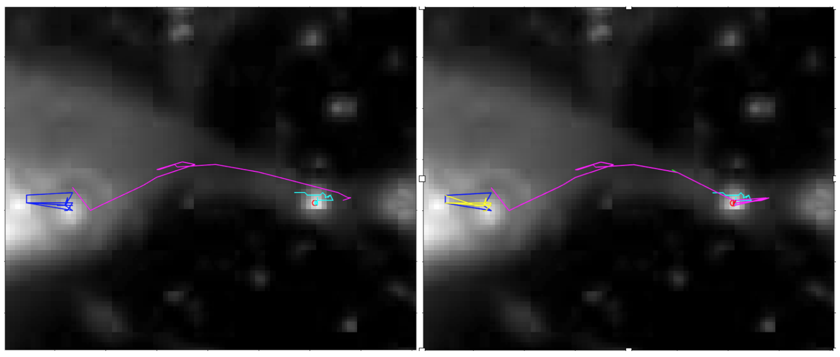


Figure 6: A comparison of trajectories from Algorithm 1(left) and Algorithm 2, tested on the blank video sample .Using $thresh_1 = 30$ and $thresh_3 = 25$

has very few and spaced out particles of interest, it is not very harmful to have such high threshold values. However, we will see in successive experiments that high threshold values can sometimes lead to unwanted particle matches.

We test Algorithm 3 on this data, with the same parameters, and notice that it produces the same results as Algorithm 2 (figure 5). Recall that we developed Algorithm 3 in order to address the obstacle of unpredictable vesicle movement. Although this particular video sample does have very unpredictable vesicle movement, this movement is isolated in an area that is not crowded by other particles, so the nearest neighbor method will still suffice, allowing Algorithm 3 to perform just as well as Algorithm 2.

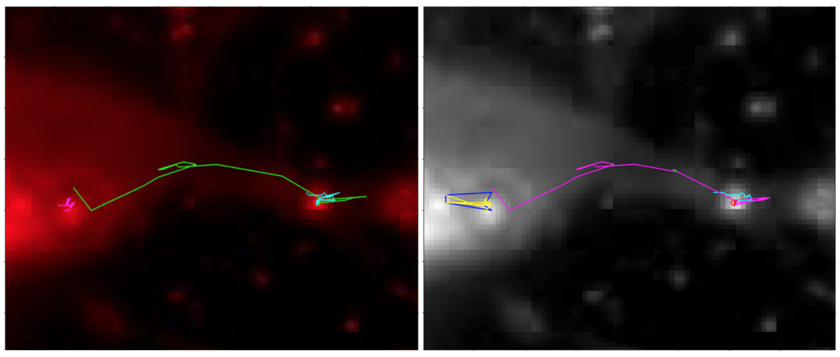


Figure 7: A comparison of trajectories from Algorithm 1(left) and Algorithm 2, tested on the blank video sample. Using $thresh_1 = 30$ and $thresh_3 = 25$

5.3 Experiment 2

We now test Algorithm 2 and 3 on a video sample that parallels that of Experiment 1. Experiment 1 was a data set of cortical vesicles in a neuron without force, while Experiment 2 is on a data set of cortical vesicles in a neuron with a magnetic force applied. This force seems to relatively stabilize the neurons we are tracking, and actually leads to minimal mean squared displacement of the vesicles. A histogram plot of the average pixels traveled per unit of time shows us the distribution of particle "velocities" in the video sample. Keeping in mind that our unit of time is one frame, we see that all of the tracked vesicles are moving at an average speed of less than 4 pixels/frame. We observe that when there is minimal movement in the data, both Algorithm 2 and Algorithm 3 have mirroring results. This is because when the vesicle movement is low, it is generally predictable and the next position is close to the current position. When successive positions are so close to each other, both Algorithm 2 and Algorithm 3 will work very well. We use $thresh_1 = 11$ because the particles in this sample do not move more than 11 pixels a frame. Any $thresh_1$ value lower than 11 will not have complete trajectories, while most values higher than 11 will work fine because the video sample has few and spaced out particles of interest.

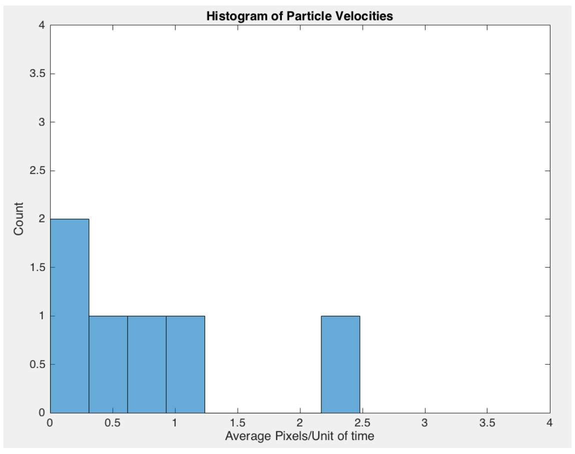


Figure 8: A histogram plot of the average pixels traveled/frame for each tracked particle in the blank video sample, with magnetic forces applied.

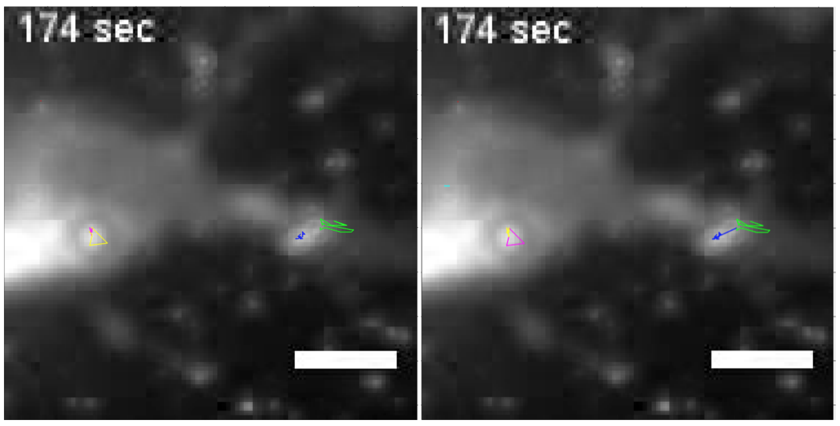


Figure 9: A comparison of trajectories from Algorithm 2(left) and Algorithm 3, tested on the blank video sample, with magnetic forces applied. Using $thresh_1 = 11$ and $thresh_3 = 10$.

5.4 Experiment 3

The next set of videos that we test are another data sample of cortical vesicles in the neuron, tagged with fluoresence and using the same imaging techniques as Experiments 1 and 2. However, these video samples pose unique constraints because there are a greater number of particles, more particles clustering, and rapid particle movement. By examining the histogram plot of average pixels traveled/frame, we see that most of the tracked particles are moving at less than 5 pixels/frame. However, we do have a couple of outliers that are moving at a drastic 20-40 pixels/frame! This scattered distribution of velocities suggests that Algorithm 3 may do a better job of accurately tracking the vesicles than Algorithm 2.

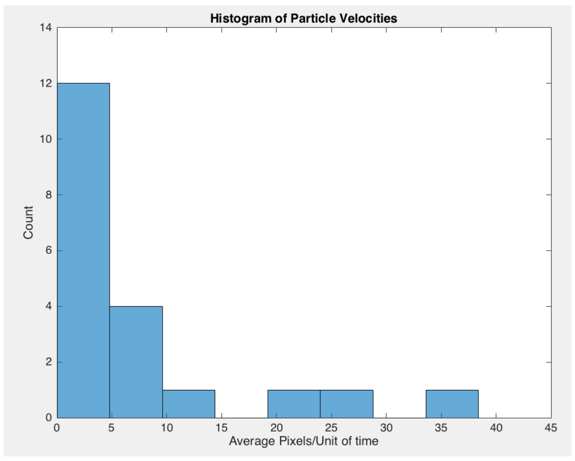


Figure 10: A histogram plot of the average pixels traveled/frame for each tracked particle in the blank video sample, with magnetic forces applied.

Figure 8 shows us that indeed Algorithm 3 does a better job of tracking the correct vesicle trajectories. The vesicles in the top right corner are moving in opposite directions and at different velocities. However the nearest neighbor approach in Algorithm 2 assumes an incorrect matching at the time when the vesicles paths collide. Because Algorithm takes into account the fact that each particle is moving at different velocities, it is able to make the correct match at the time of collision. However, because of the vesicles swift movement, we must use a very large $thresh_1$ parameter of 120 to capture the particle jump. The tradeoff of a large $thresh_1$ value is the unwanted connections occurring at the bottom left of the image.

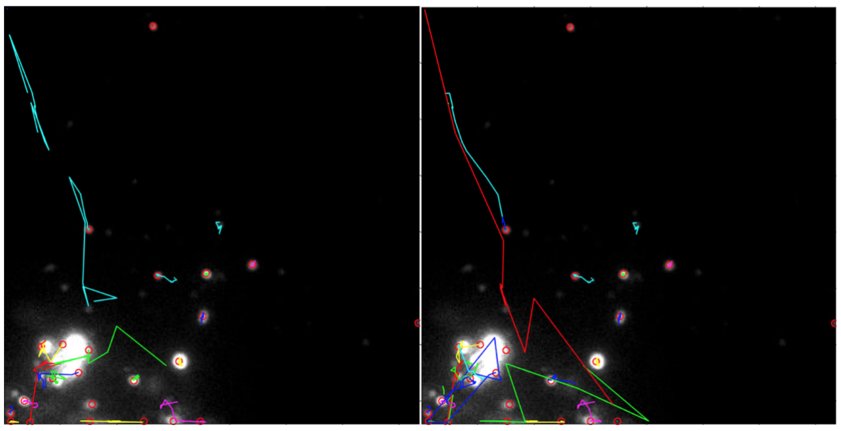


Figure 11: A comparison of trajectories from Algorithm 2(left) and Algorithm 3, tested on the blank video sample, with magnetic forces applied. Using $thresh_1 = 120$ and $thresh_3 = 10$.

5.5 Experiment 4

The next video parallels Experiment 3, but has a magnetic force applied to the cortical vesicles. We can see from the histogram plot that the magnetic force applied in this data sample causes the particles to be relatively stationary. We also note that since there are clumps of vesicles, it is difficult for our detection algorithm to accurately pinpoint the vesicles, leading to noisy trajectories in the bottom left corner.

Because the particles in this data sample all move less than 6 pixels/frame, we expect that Algorithm 2 and Algorithm 3 will perform similarly. Because there is minimal movement, we are able to use a smaller $thresh_1$ value of 20.

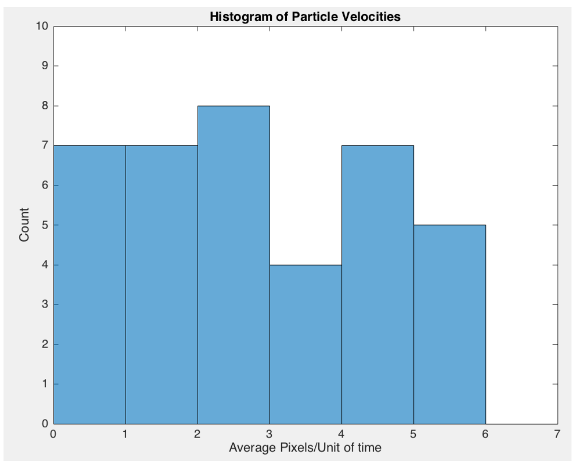


Figure 12: A histogram plot of the average pixels traveled/frame for each tracked particle in the blank video sample, with magnetic forces applied.

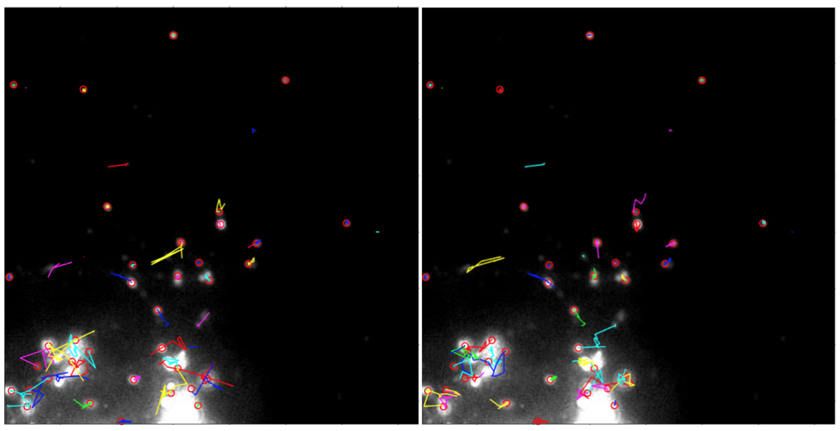


Figure 13: A comparison of trajectories from Algorithm 1(left) and Algorithm 2, tested on the blank video sample, without magnetic forces applied. Using $thresh_1 = 20$ and $thresh_3 = 0$.

5.6 Experiment 5

The last test that we do is on secretory granule motion observed by evanescent wave fluorescence microscopy, taken from reference 2. Characteristics of this data set include many vesicles moving slowly and closely together, so we must use a very small $thresh_1$ value of 5. Since the histogram confirms that all particles move at a similar velocity of less than 2 average pixels/frame, we expect Algorithm 2 and Algorithm 3 to perform similarly.

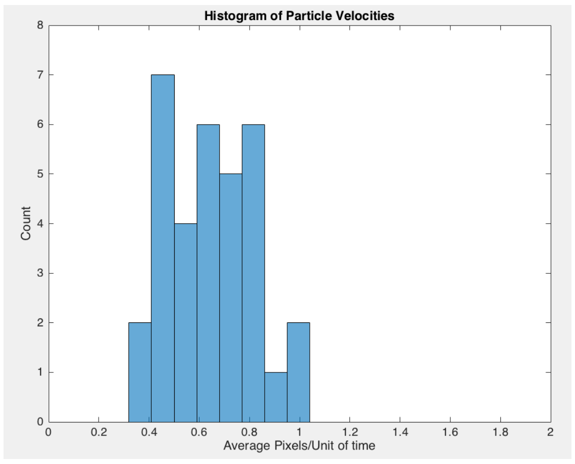


Figure 14: A histogram plot of the average pixels traveled/frame for each tracked particle in the blank video sample.

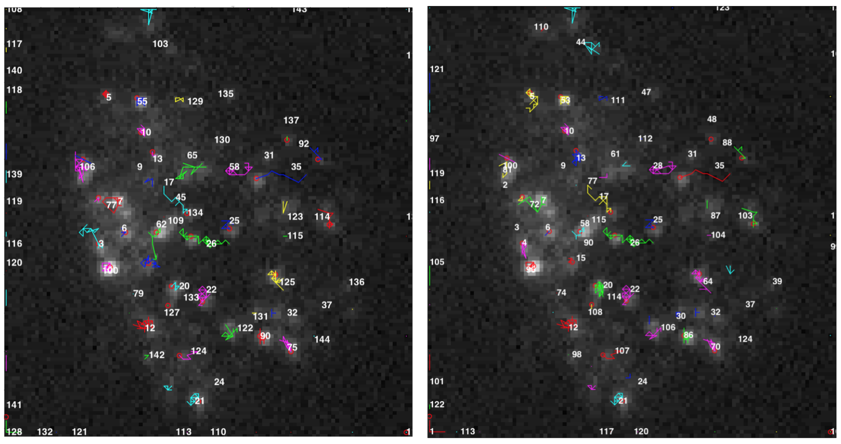


Figure 15: A comparison of trajectories from Algorithm 1(left) and Algorithm 2, tested on the blank video sample, without magnetic forces applied. Using $thresh_1 = 5$ and $thresh_3 = 0$.

5.7 Mean Squared Displacement Analysis

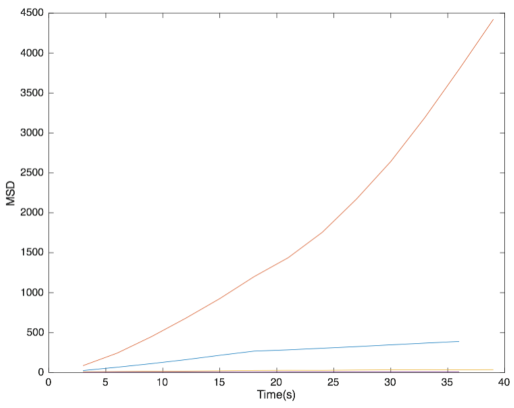


Figure 16: Mean Squared Displacement results from Experiment 1 Tracks.

Figure 16 is the Mean Squared Displacement results from the particles being tracked in Experiment 1. We can observe the different diffusion patterns for each of the particles being tracked, specifically that one particle is super diffusive over time.

6 Conclusion

Our experiments show that our proposed solutions were able to address the initial obstacles of object collision, unpredictable vesicle movement, and noisy vesicle detection. We note that both Algorithm 2 and Algorithm 3 preform similarly when the vesicles we want to track are relatively stationary. However, when there is a wider distribution of vesicle velocities, Algorithm 3 is able to pick up these differences and therefore out performs Algorithm 2. Further analysis should be focused on improving the particle detection step with more advanced imaging tools (such as template matching, etc.), and possibly improving the velocity calculation in Algorithm 3. The following table (Table 1) is a summary of the various parameters we use in the particle detection and tracking steps.

Parameter	Description	**Optimal Range**
size	Controls the width of the LoG filter, that we convolute with the image in the particle detection step. The wider the LoG filter, the more particles are detected.	**3-4**
sigmah	Controls the breadth of the peak of the LoG filter, that we convolute with the image in the particle detection step. The broader the peak, the less affected the the filter is to noise, resulting in less particles being detected.	**0.43-0.45**
thresh1	When using the Hungarian one-to-one matching in the particle matching step, we want to disregard matches that are too far apart. If matches are father apart than Thresh1, we reject them in our algorithm. High values of Thresh1 are optimal in cases where there are large true particle jumps between frames. Thresh1 essentially corresponds to the farthest true jump a particle makes.	**5-35**
thresh2	In the particle matching step, if a particle has not been matched up for Thresh2 number of frames, we assume that the particle is no longer in the image and we remove it from our tracks. If the data is noisier and particles are not detected reliably, we can have a higher value of Thresh2.	**4-8**
thresh3	If there are collisions in the particle trajectories, we allow for two particles to be matched to the same detection in the next frame if the matches are less than Thresh3 distance away from each other. Thresh3 should only be set higher if we anticipate collisions, or have noisy data. Thresh3 essentially corresponds to the farthest jump a particle makes into a collision. Furthermore, Thresh32.	**0-15**

Table 1: Summary of our Algorithm Parameters

References

[1] Anja Kunze, Peter Tseng, Chanya Godzich, Coleman Murray, Anna Caputo, Felix E.Schweizer, Dino Di Carlo "Engineering Cortical Neuron Polarity with Nanomagnets on a Chip", ACSNANO, Vol.9 No.4, pages 3664-3676, 2015.

[2] Claire Desnos, Jean-Sbastien Schonn, Sbastien Huet, Viet Samuel Tran, Aziz El-Amraoui, Graa Raposo, Isabelle Fanget, Catherine Chapuis, Gal Mnasch, Genevive de Saint Basile, Christine Petit, Sophie Cribier, Jean-Pierre Henry, and Franois Darchen, "Rab27A and its effector MyRIP link secretory granules to F-actin and control their motion towards release sites", J Cell Biol 2003 163:559-570. Published November 10, 2003, doi:10.1083/jcb.200302157.