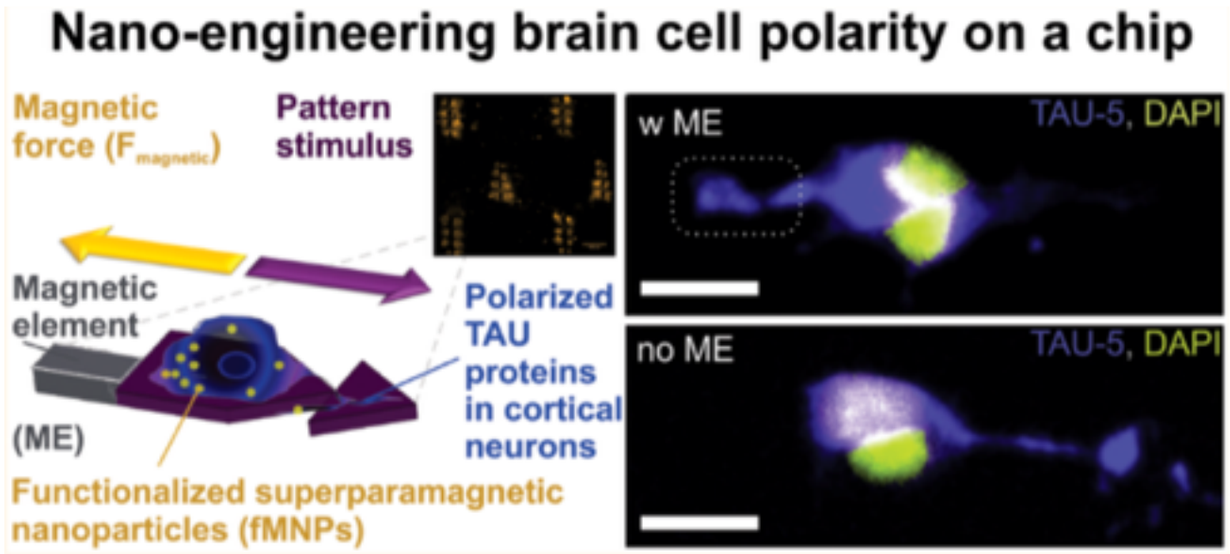
particle tracking

MATH 199

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.



motivation

- 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.

- 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.

state of the art

- Different particle detection methods have been developed depending on the domain and the subject that is being tracked. Template matching, and other image processing techniques are specifically implemented to fit the given circumstances.

- Algorithms such as the Kalman Filter are also in use for particle matching between frames. The Kalman Filter is beneficial when the object being tracked has predictable movement and generally constant characteristics (velocity, acceleration, etc.)

state of the art

The current object tracking algorithm consists of two major elements:

Particle Detection:

- Particle detection is done on a frame by frame basis in Matlab.
- The detection Algorithm primarily uses the Laplacian of the Gaussian filter to find the edges of the vesicles in each frame.
- It then uses simple thresholding and morphological techniques to record unique points at the locations of corresponding vesicles.

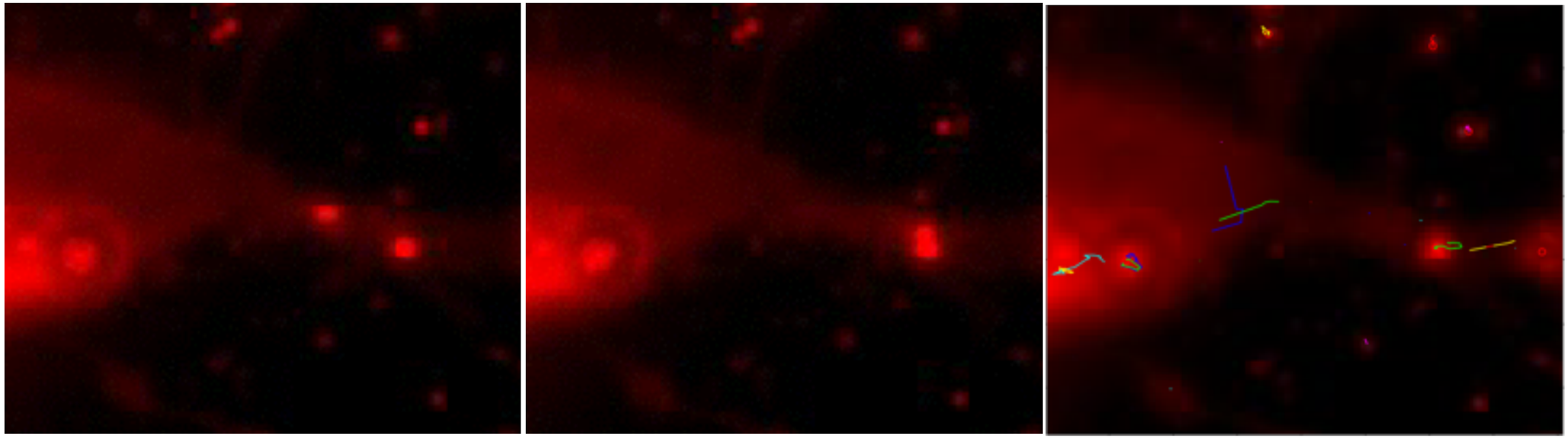
Particle Matching:

Particle matching aims to link vesicles from one frame, to their corresponding position in the next frame.

- Step 1: Compute a cost matrix with the estimated distances from each detected vesicle in the previous frame, to each detected vesicle in the current frame.
- Step 2: Input in the cost matrix to Munkres Algorithm, which efficiently pairs detected vesicles from the subsequent frames according to the **lowest total distance (one to one matching).**
- **Check:** f a detected vesicle in the previous frame was not assigned to a vesicle in the current frame, we give the vesicle a “strike” (after x number of strikes, the vesicle gets deleted)
- **Check:** If a detected vesicle in the current frame is not assigned, we consider the vesicle as a new object to track and add it to our list.

current obstacles

- vesicle collisions
- unpredictable vesicle movement
- noisy particle detection



proposed solutions

- obstacle 1: vesicle collisions

Vesicles sometimes move very close to each other, 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 a vesicle is close enough to a vesicle that has already been matched, using the **Nearest Neighbor method**.

- obstacle 2: unpredictable vesicle movement

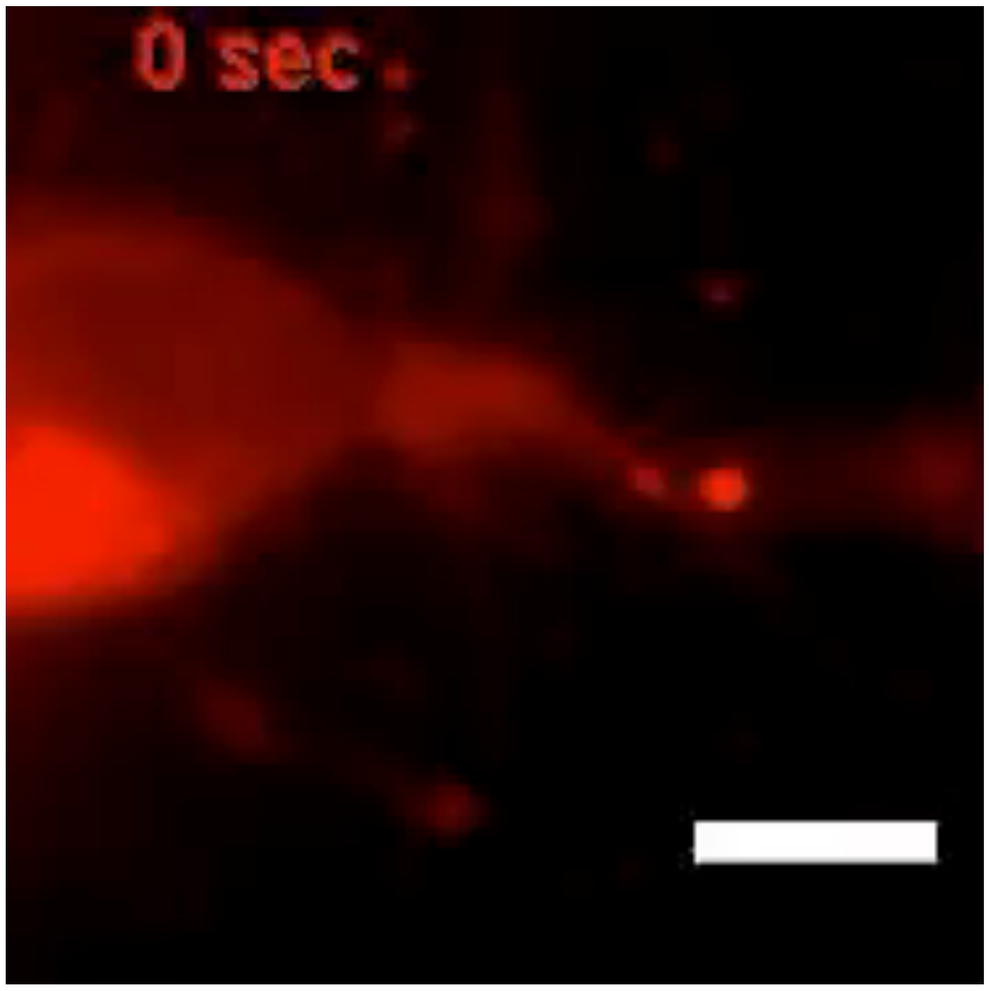
The vesicle movement is very unpredictable, and is different for each vesicle being tracked. Therefore, it is hard to use methods such as the KALMAN FILTER, which assume a specific type of movement for each vesicle in the frame. However, experimenting with the **Bayesian Approach** to vesicle tracking might be a solution to this obstacle.

- obstacle 3: noisy particle detection

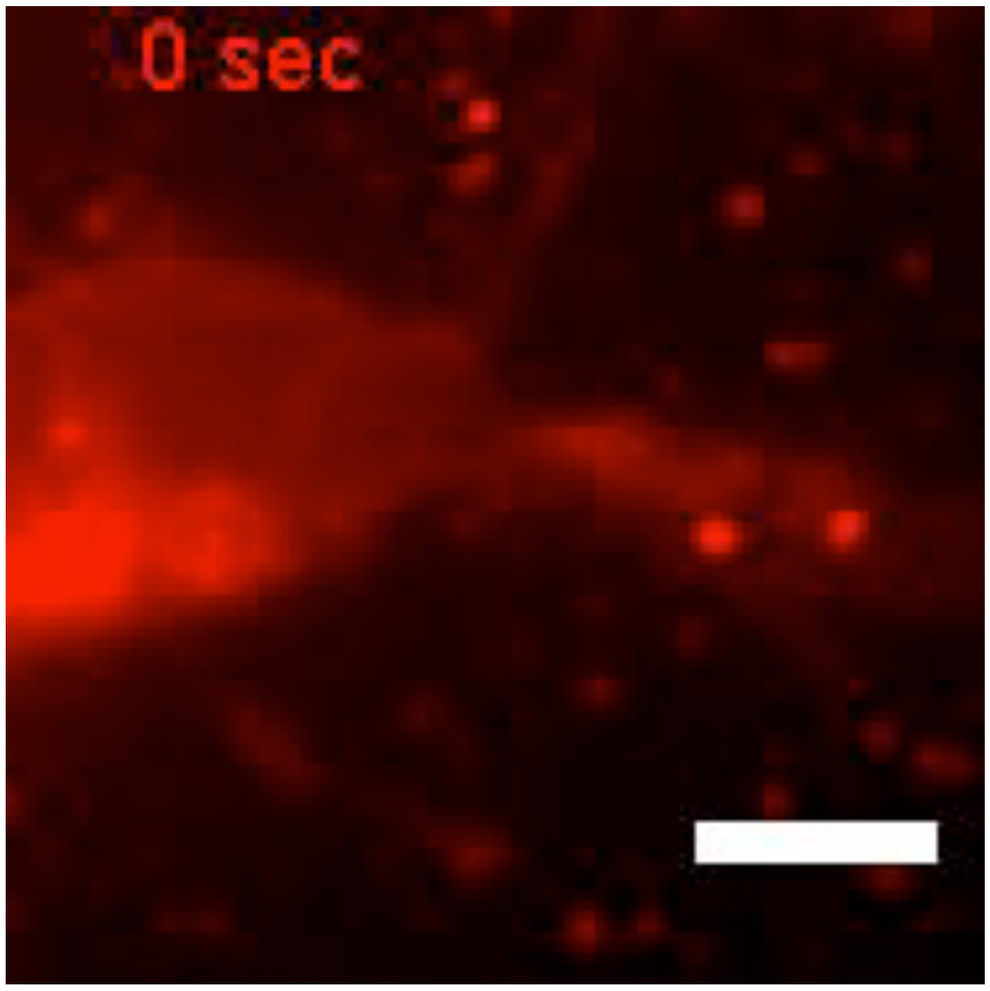
Vesicles are not always easy to detect because sometimes due to optics properties, we observe bright rings around the vesicles after imaging. This in conjunction with the fact that the laplacian of the gaussian filter is not 100% reliable in detecting vesicles in each frame, leads to noisy particle detection. To ease this obstacle, we impose a **Boundary Condition.**

video samples

These are video samples of live cell experiments preformed on-chip, with and without magnet application. The cortical neurons 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.

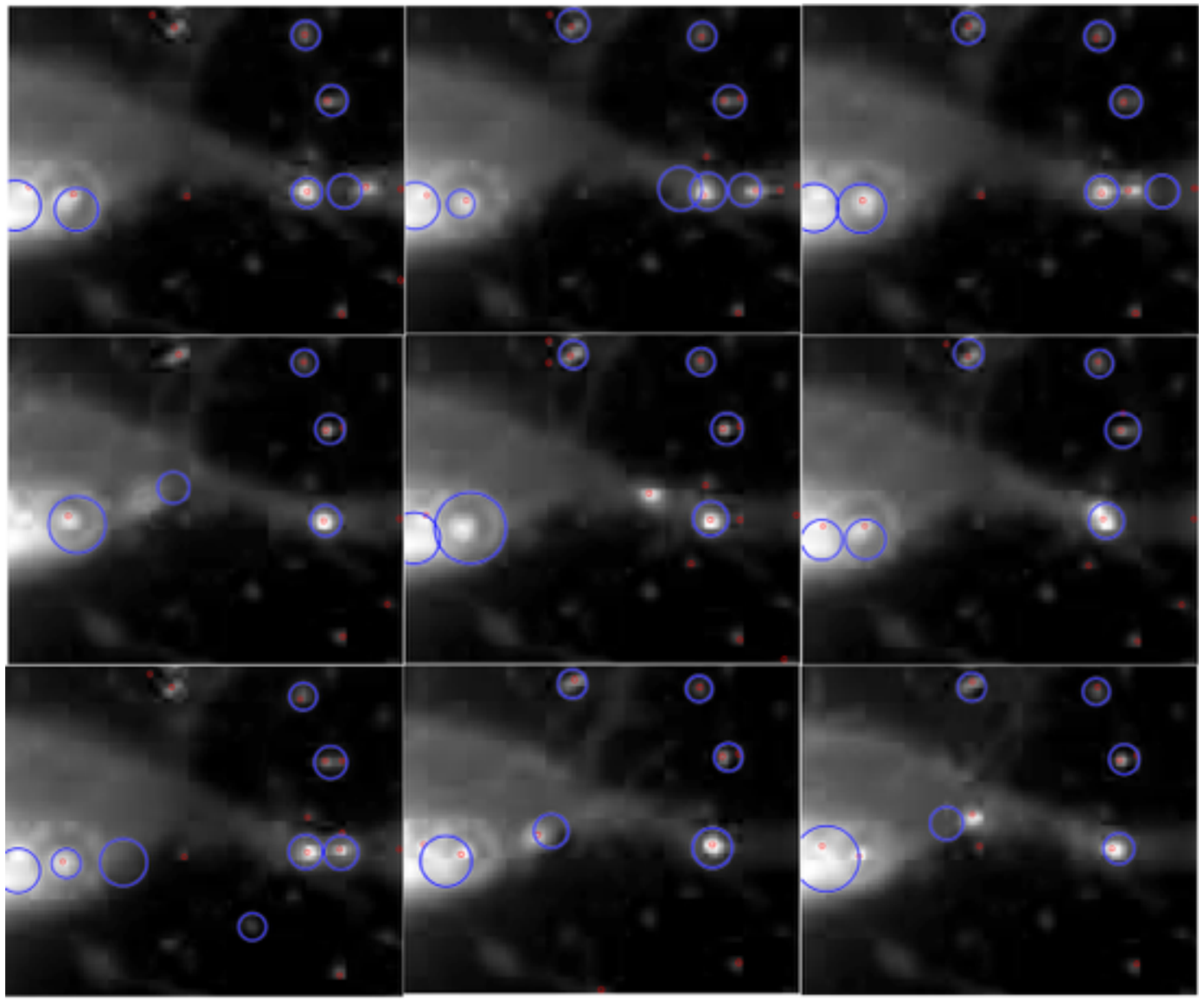


without force



with force

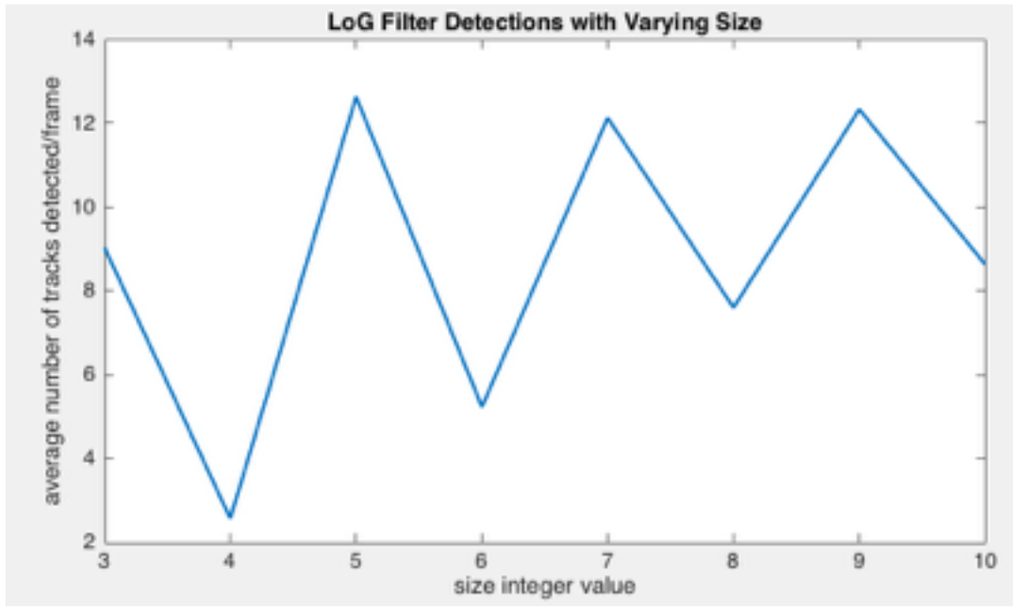
detection results



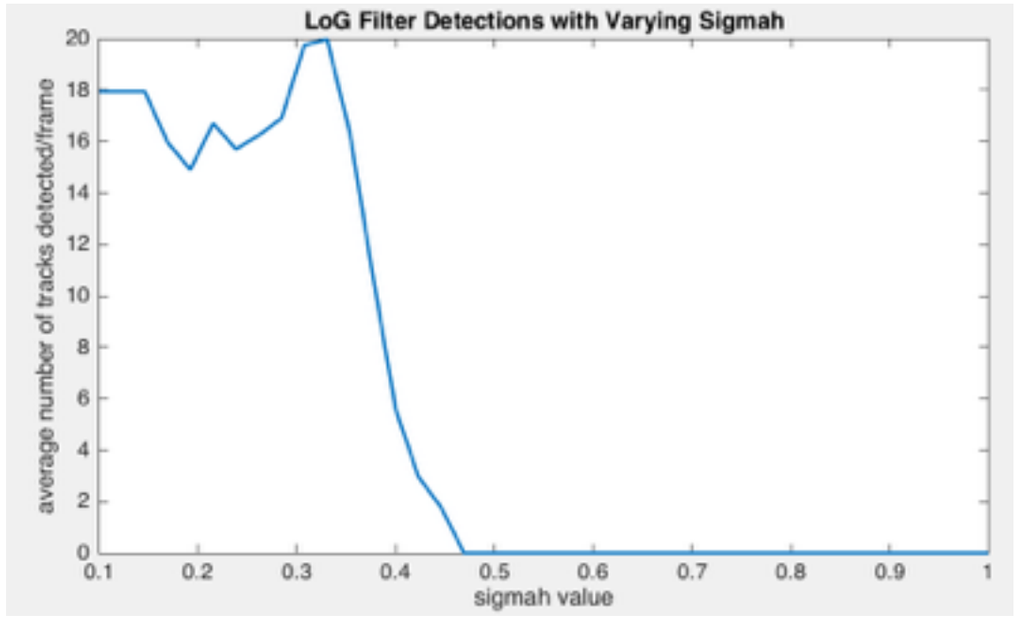
red circles: LoG Filter using sigmah=0.43, size=0.4, **blue circles**: Hough Transform for comparison

detection parameter analysis

Size:



Sigmah:



the bayesian approach

The main motivation for the Bayesian Approach is to better account for the unpredictable movement of the vesicles.

- 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, when we have vesicle movement that is very unpredictable for the Kalman Filter, we run into problems.

- In the **Bayesian Approach,** 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 **Gaussian Distribution,** which are specifically determined for each vesicle according to the vesicle's **recorded velocity** from the previous frame.
    - **When the velocity is higher, the particle will have a flatter Gaussian Distribution.**

- Now 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.

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:

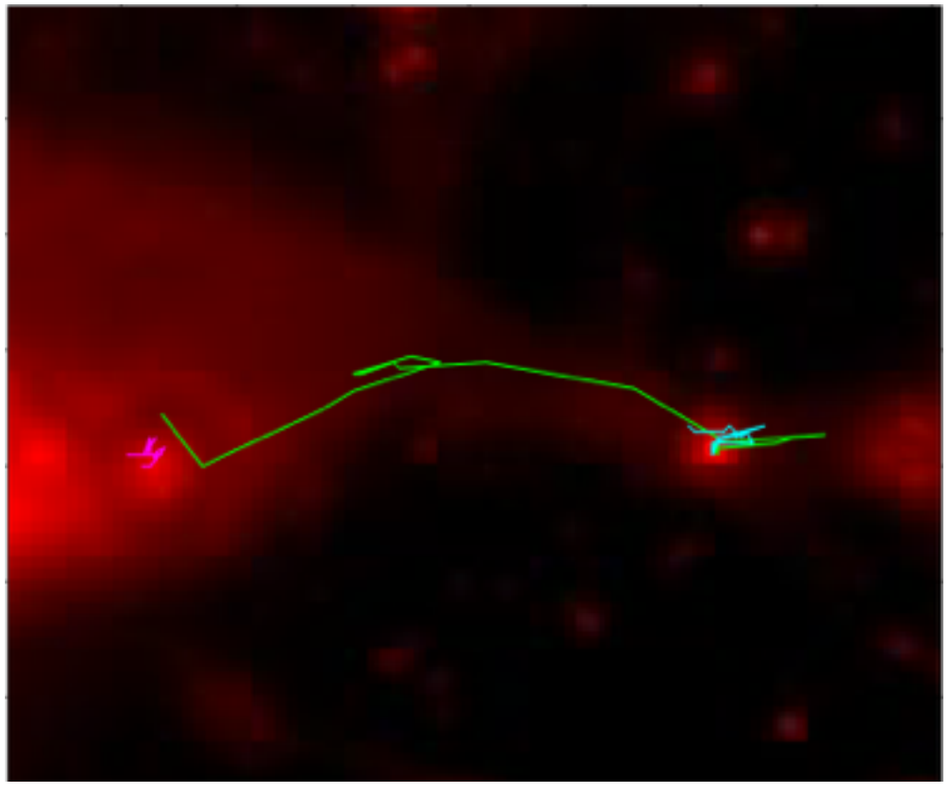
1. Determining the boundary
2. Make sure that new particles can only be added to the algorithm (during **Check 1**) if they lie within a range of this boundary.

Step 1: Thresholding out the area of interest, and using Matlab and morphology tools to get one region and find boundary (in green)

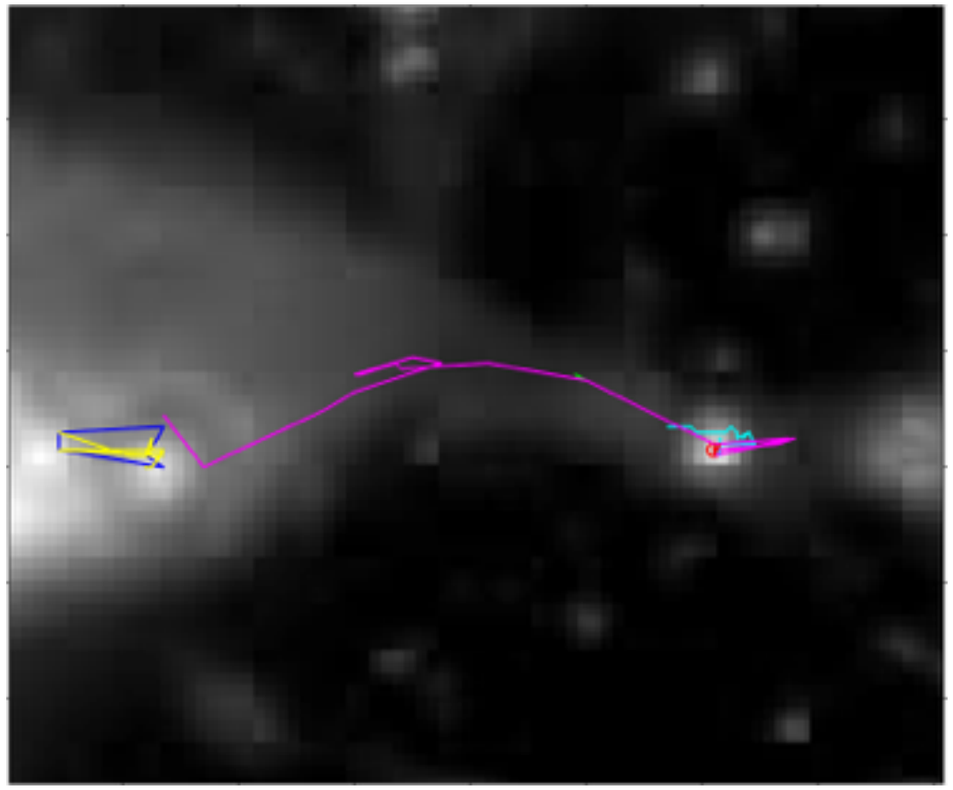
Step 2: Add a condition to the algorithm to only accept a particle as new **if it is inside the boundary**.

progress

Combo Algorithm:



Bayesian Approach:

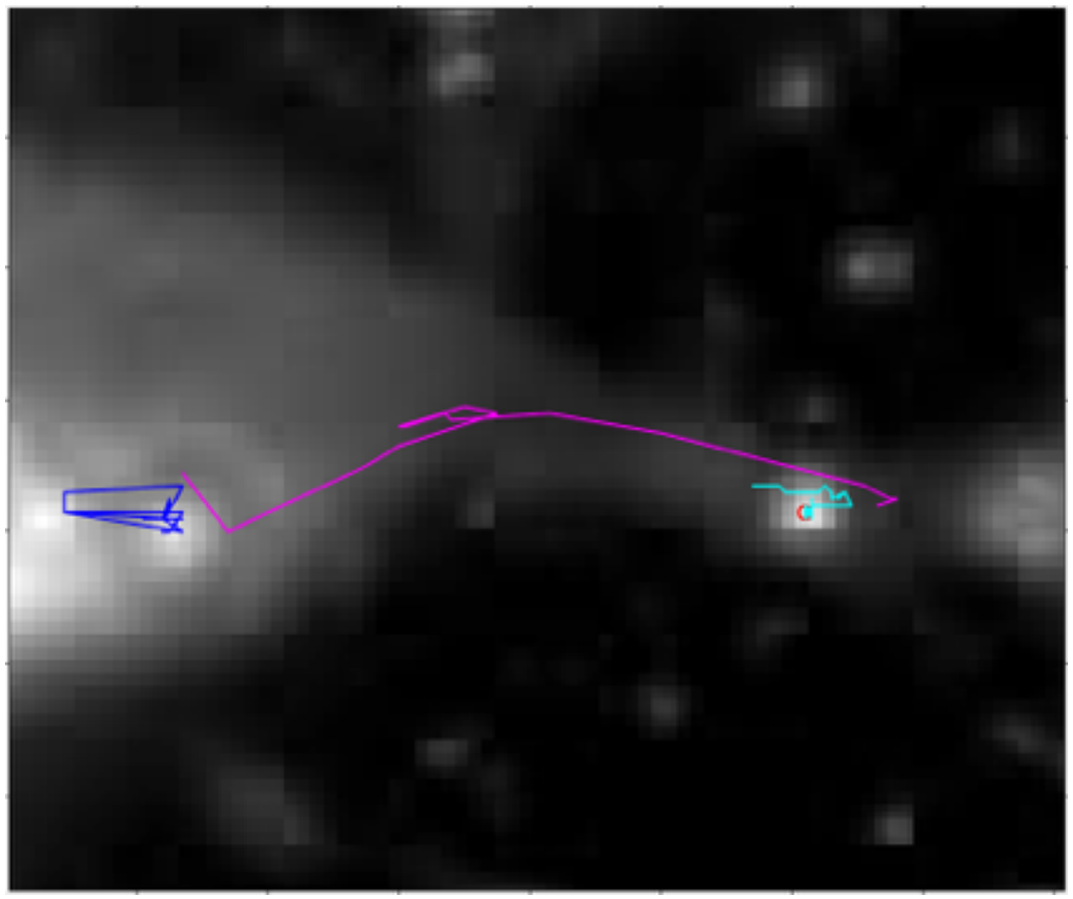


without magnetic force

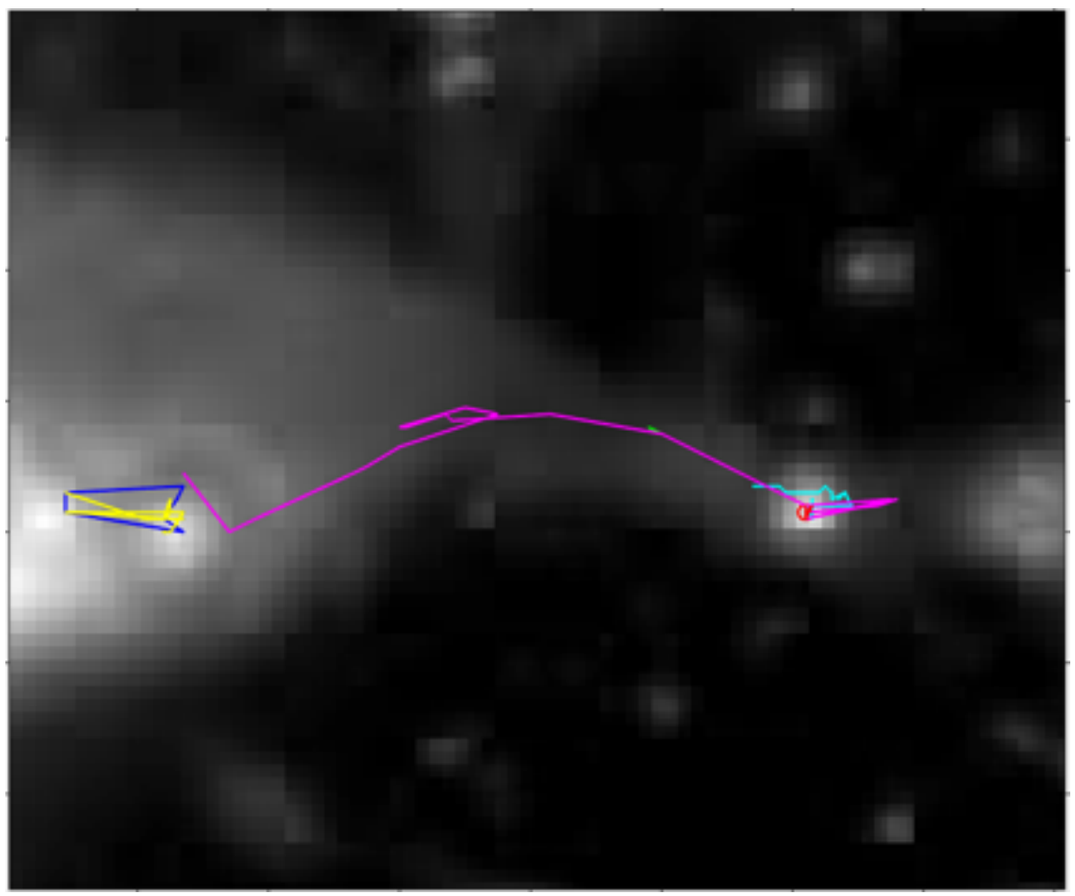
for comparison..

if we fail to incorporate the nearest neighbor method into the algorithm, we observe that the tracking does not accurately pick up the trajectory at the time of collision

without including N.N.



with N.N.

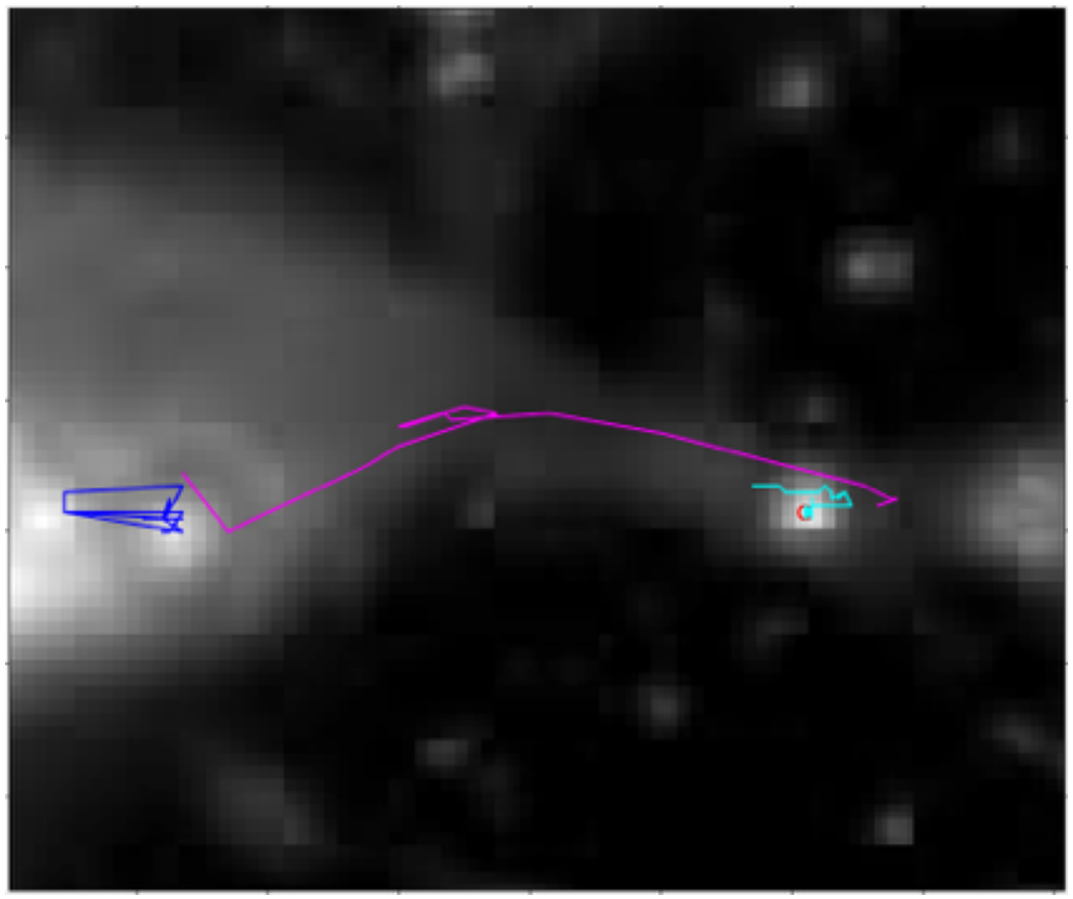


we are able to obtain the correct trajectory by increasing the nearest neighbor threshold, but this comes with a cost of extra noise in the surrounding particles

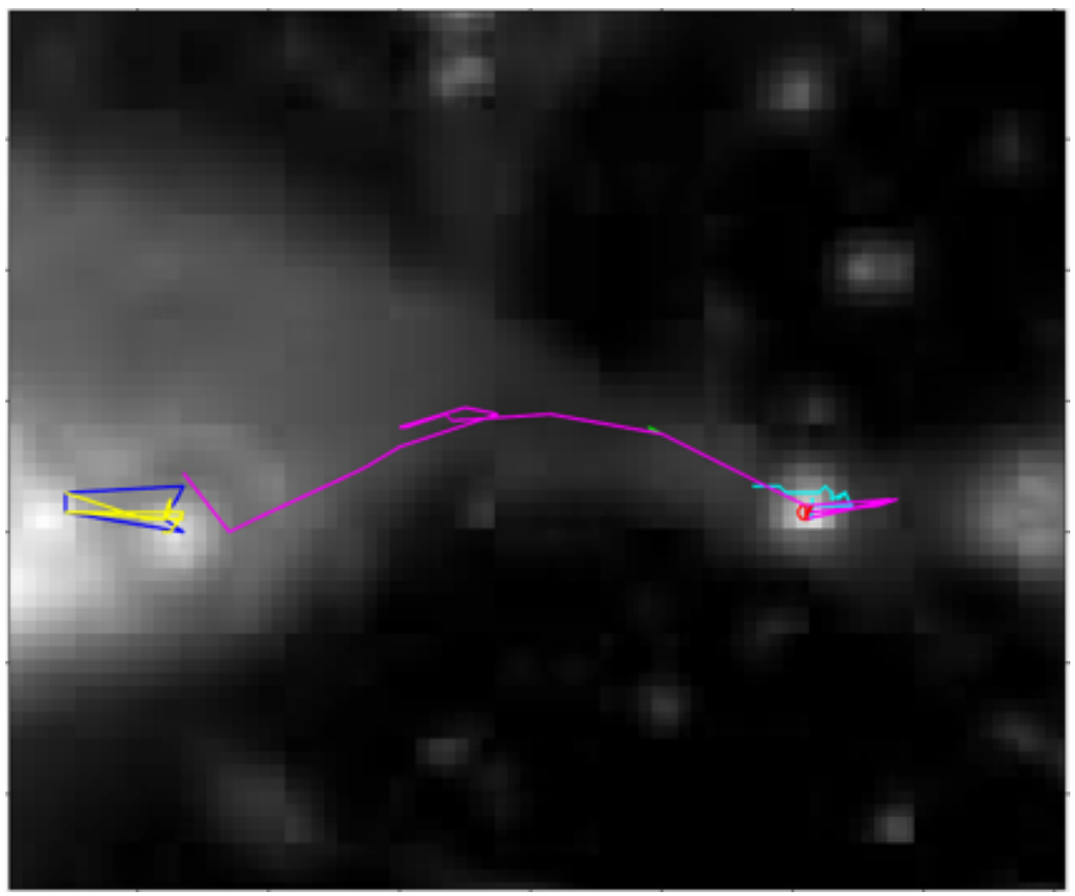
for comparison..

if we fail to incorporate the nearest neighbor method into the algorithm, we observe that the tracking does not accurately pick up the trajectory at the time of collision

without including N.N.



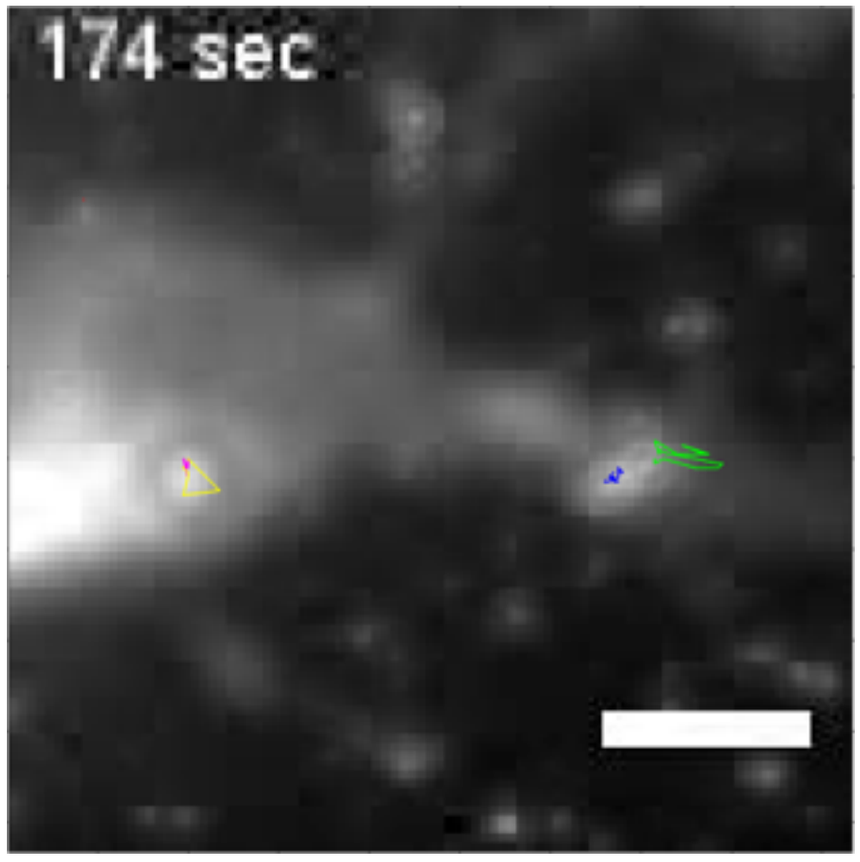
with N.N.



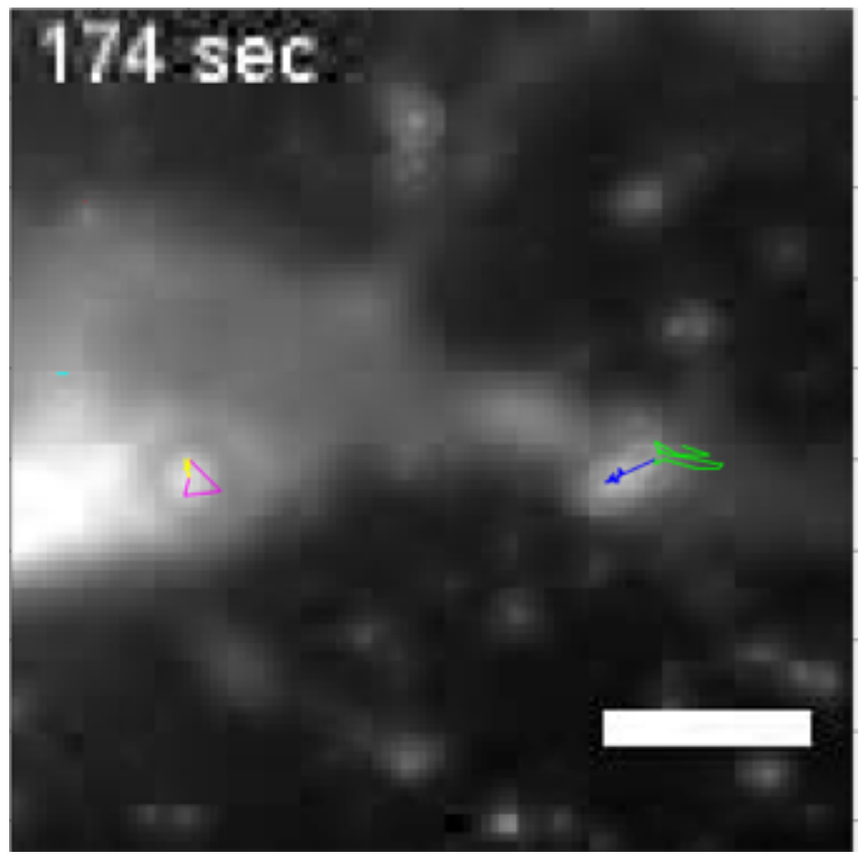
we are able to obtain the correct trajectory by increasing the nearest neighbor threshold, but this comes with a cost of extra noise in the surrounding particles

progress

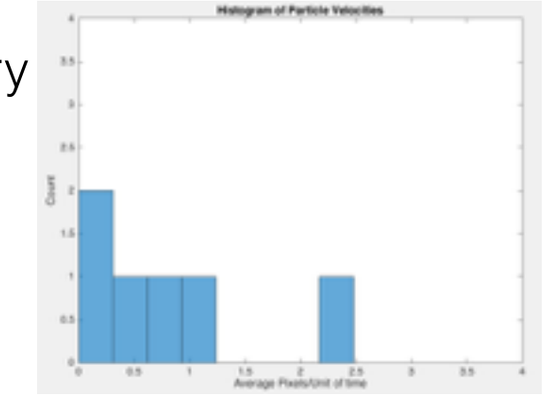
Combo Algorithm:



Bayesian Approach:



with magnetic force..particles are mostly stationary

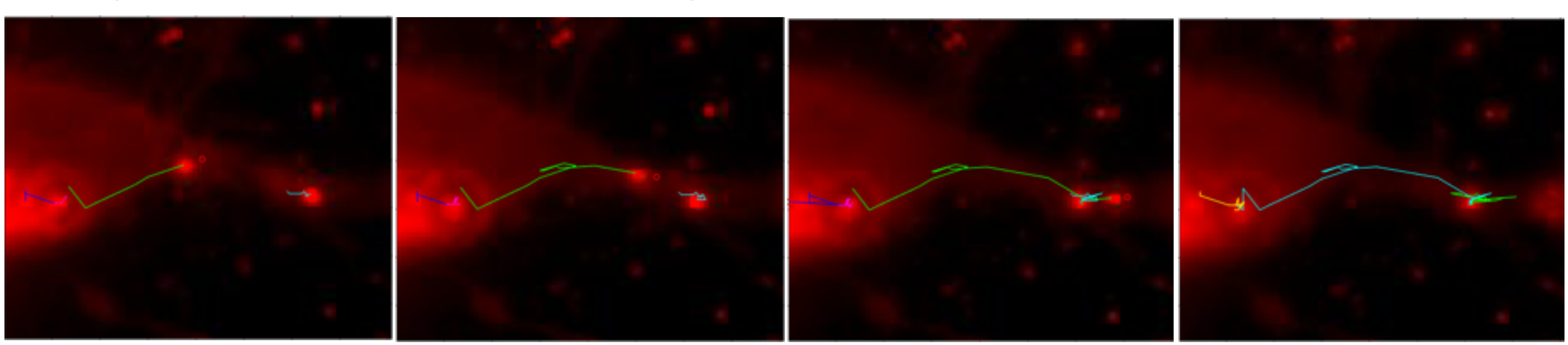


adding a predictive element

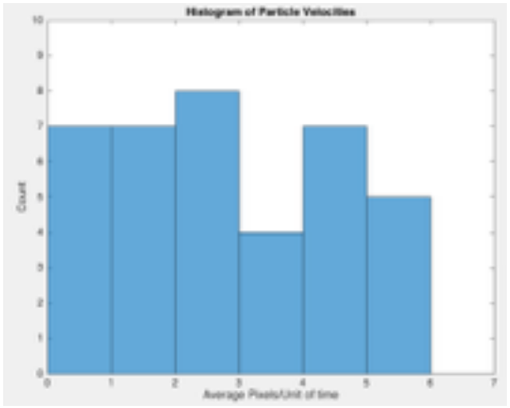
In order to lower some of the distance thresholds (distance from the vesicle in previous frame, to its supposed match in the current frame), we have implemented a predictive element to the algorithm, that follows the following process:

step 1: Estimate the next position of a vesicle. We estimate by taking the average of the previous two displacements, and applying this average to predict the next position.

step 2: Compare the **estimated position** to the **actual detections** in the next frame. And follow through with the original distance cost matrix and matching procedure.

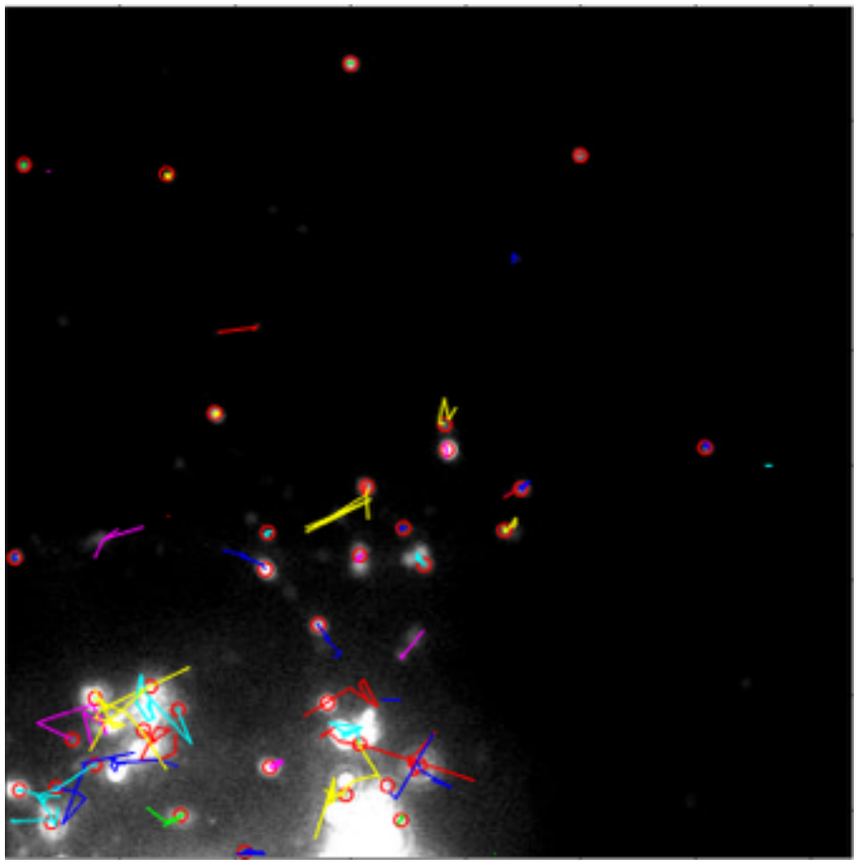


note: adding this element allowed us to decrease thresholds by 7 pixels, and avoid some unnecessary matches. However, also note that the predictive element may be very specific to this data set

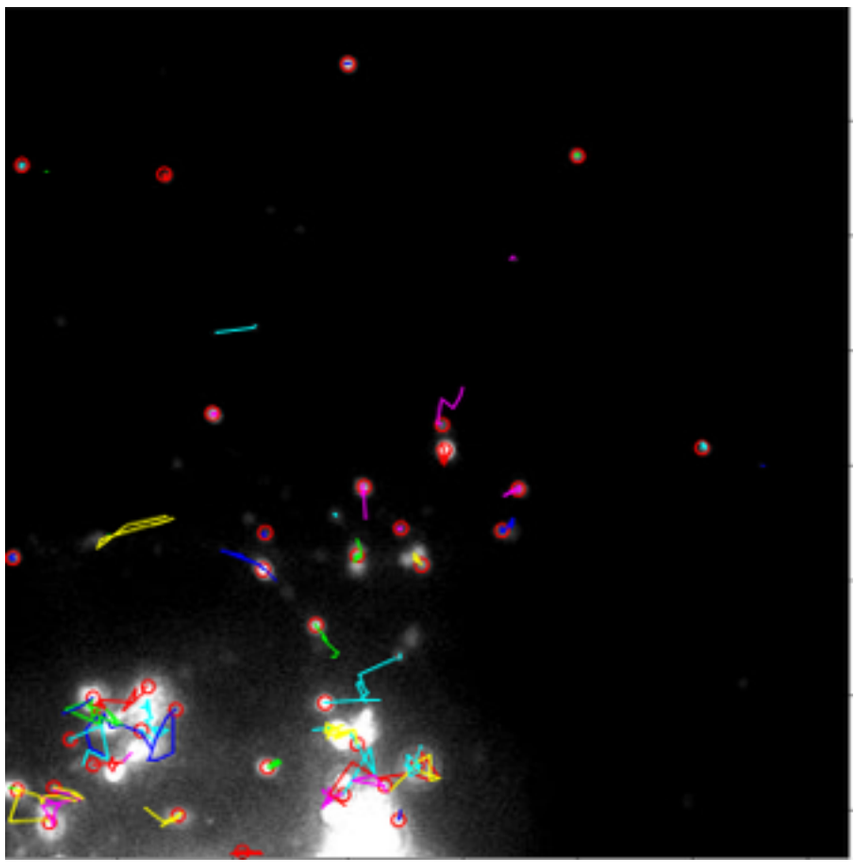


progress

Combo Algorithm:

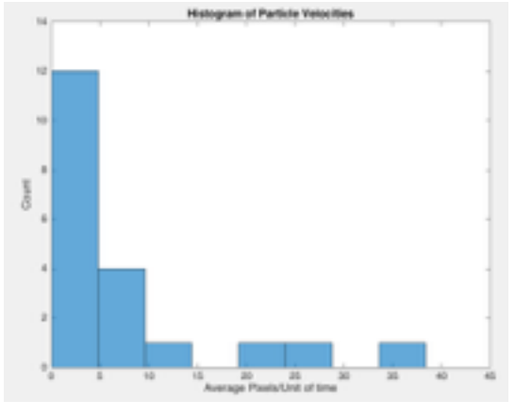


Bayesian Approach:



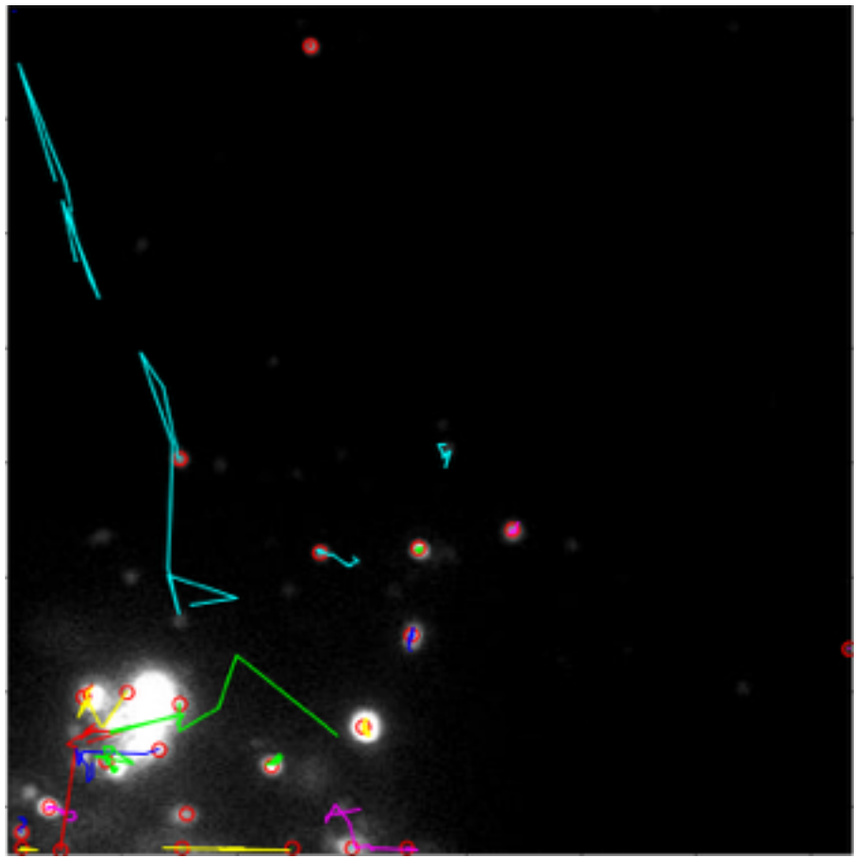
Notes:

- The Bayesian approach is better able to capture the movement of the particles because it accounts for the fact that some particles are more active than others
- Again, with magnetic force particles are mostly stationary

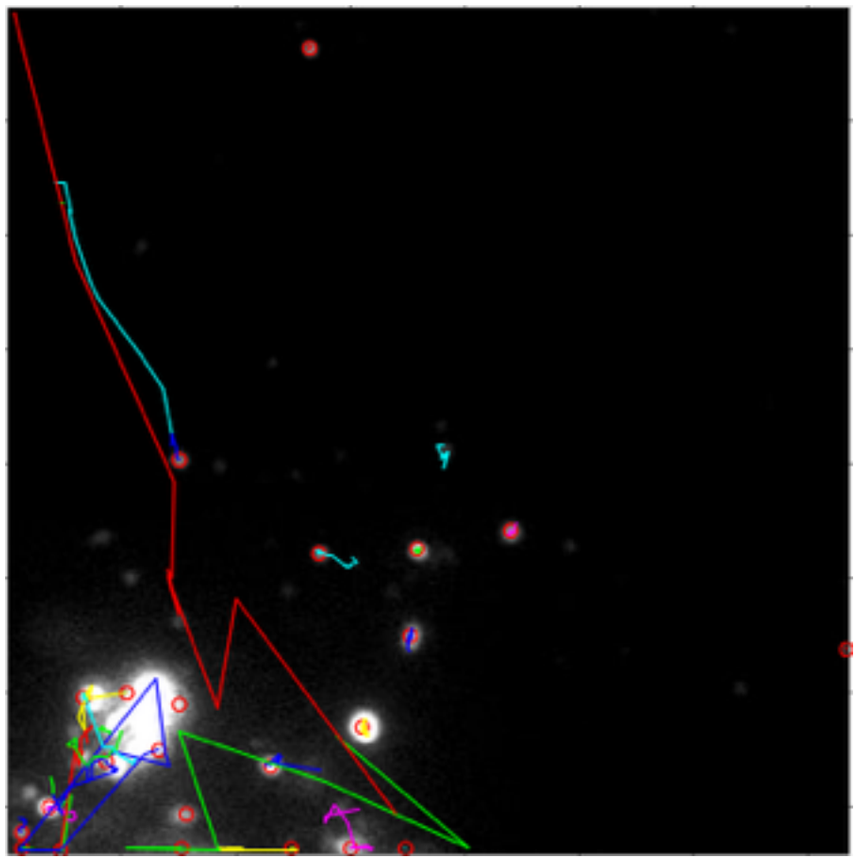


progress

Combo Algorithm:



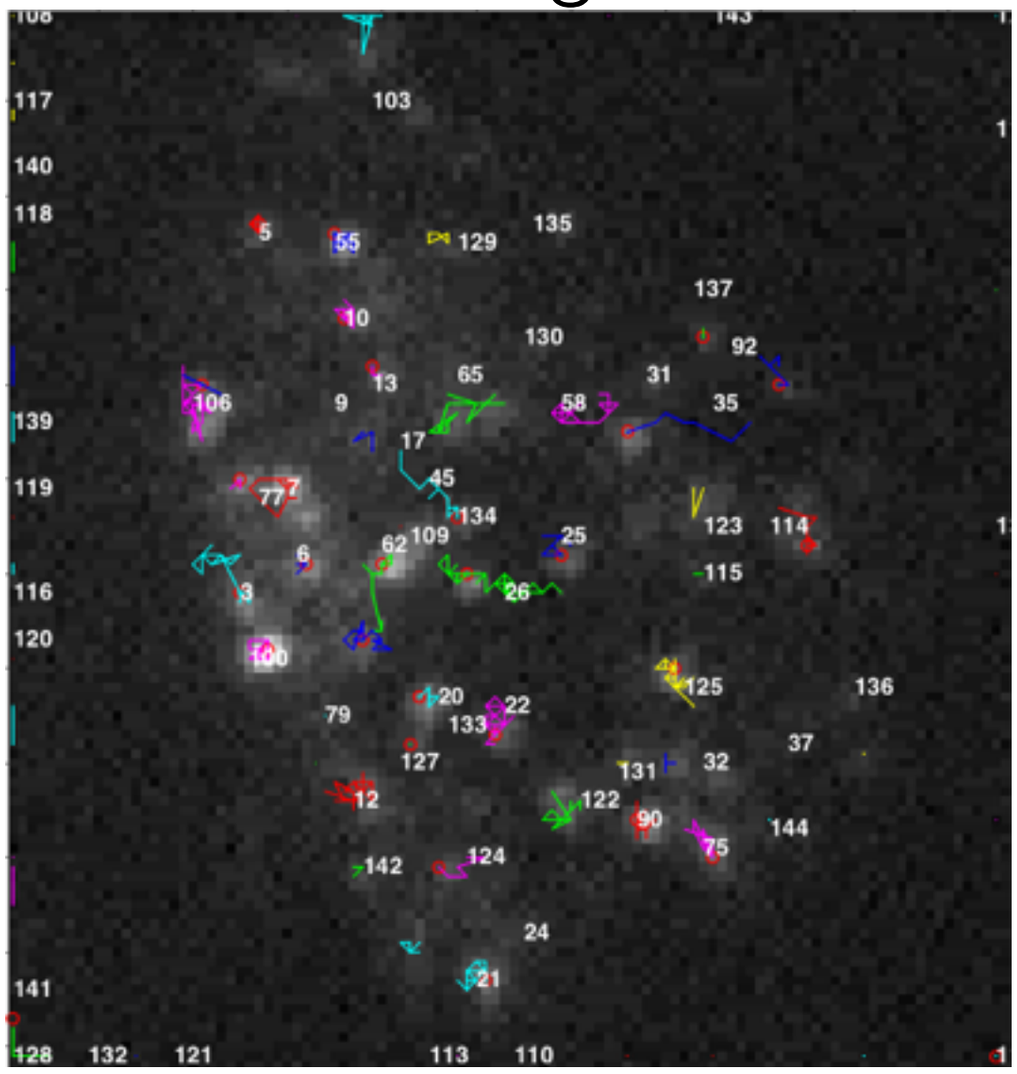
Bayesian Approach:



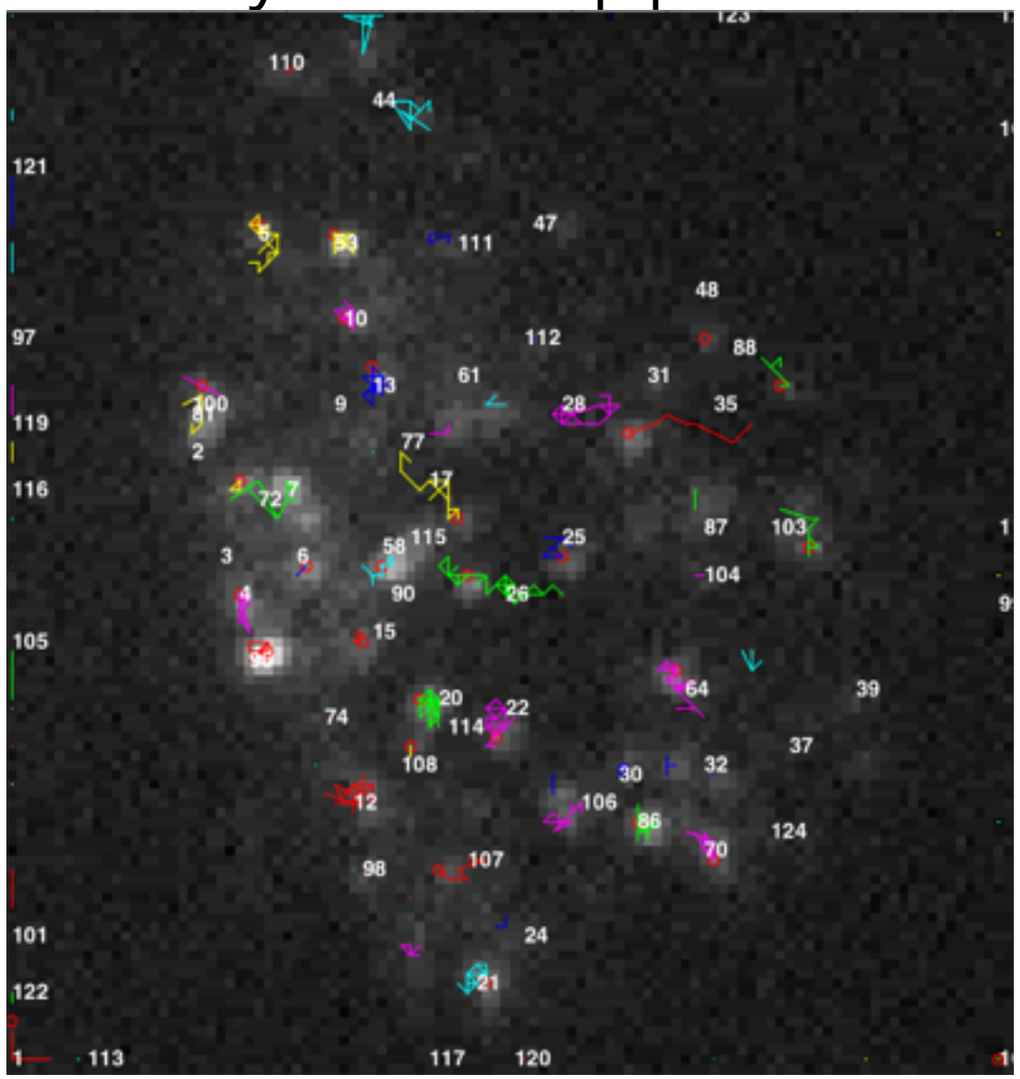
Notes:

- The Bayesian approach is captures the correct movement of the vesicles (in red), but at the cost of very high threshold needed to account for the swift and jumpy movement.
- This high threshold causes a lot of noise and errors in the other particle trajectories.
- The combo algorithm evidently is not able to correctly link the vesicle movement, but also does not need high thresholds and therefore has a less noise in the bottom left corner.

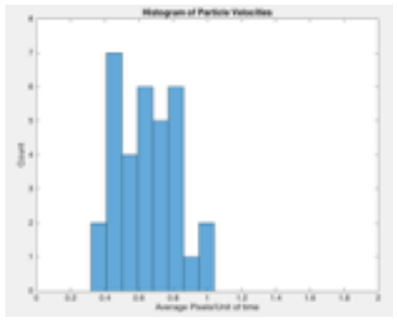
Combo Algorithm:



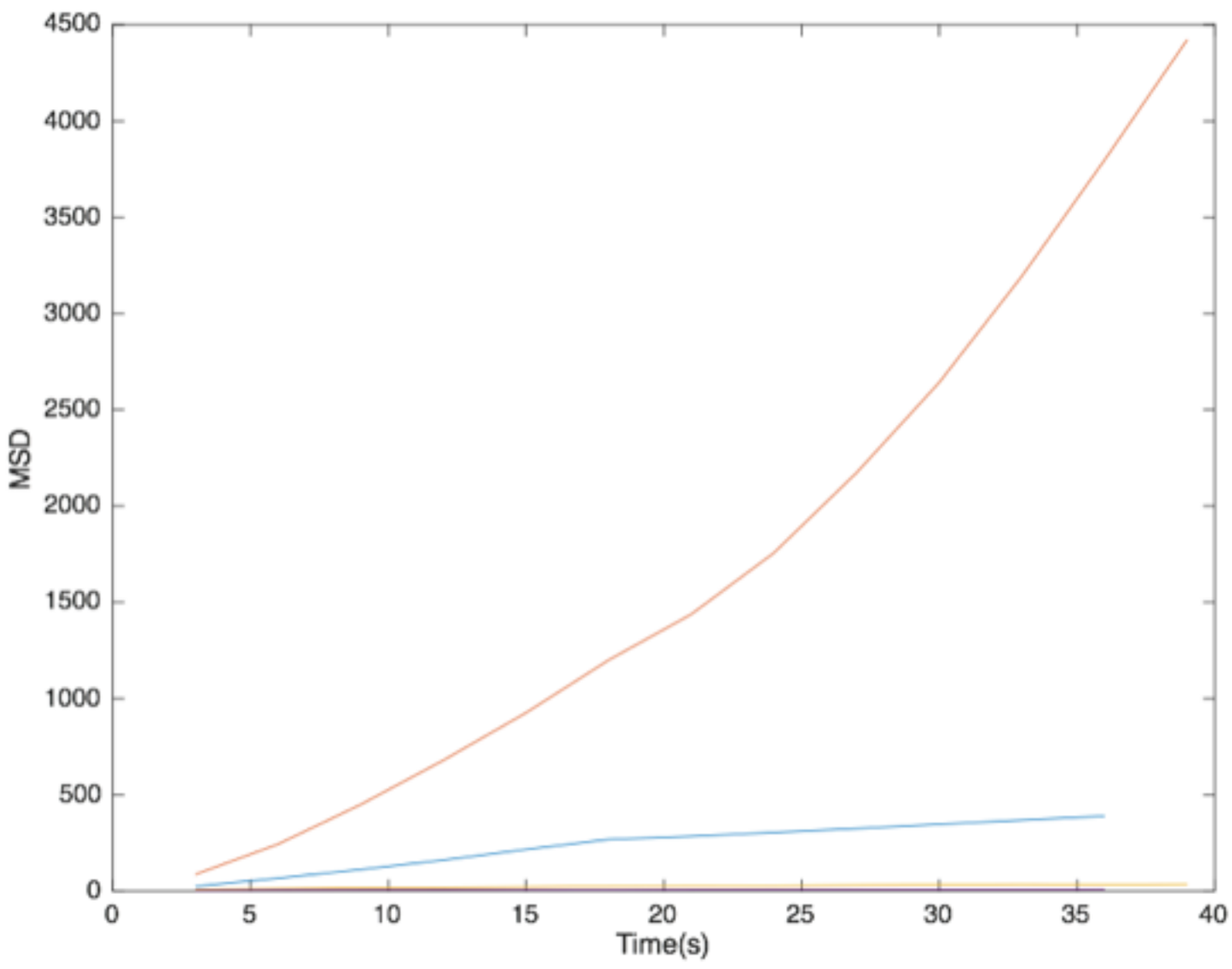
Bayesian Approach:



Video 4 (2.7 MB)
Secretory granule motion observed by evanescent wave fluorescence microscopy. A PC-12 cell was cotransfected by NPY-GFP and the Rab binding domain of MyRIP (MyRIP-RBD) and imaged 3 d later by EW-FM. False color is used. Images were collected at 2 frames/s for 60 s, and the sequence is accelerated 5 times on the video (10 frames/s).



mean squared distance plot



parameter summary

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
thresh_1	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 Thresh_1, we reject them in our algorithm. High values of Thresh_1 are optimal in cases where there are large true particle jumps between frames. Thresh_1 essentially corresponds to the farthest true jump a particle makes.	5-35
thresh_2	In the particle matching step, if a particle has not been matched up for Thresh_2 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 Thresh_2.	4-8
thresh_3	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 Thresh_3 distance away from each other. Thresh_3 should only be set higher if we anticipate collisions, or have noisy data. Thresh_3 essentially corresponds to the farthest jump a particle makes into a collision. Furthermore, Thresh_3<Thresh_2.	0-15

outlook

- have more reliable particle detection, by using more advanced image processing techniques(template matching?)
- improving the velocity calculations for the bayesian approach