Ilastik classification, tracking, and manual editing workflow

Braden Stephens Keiser

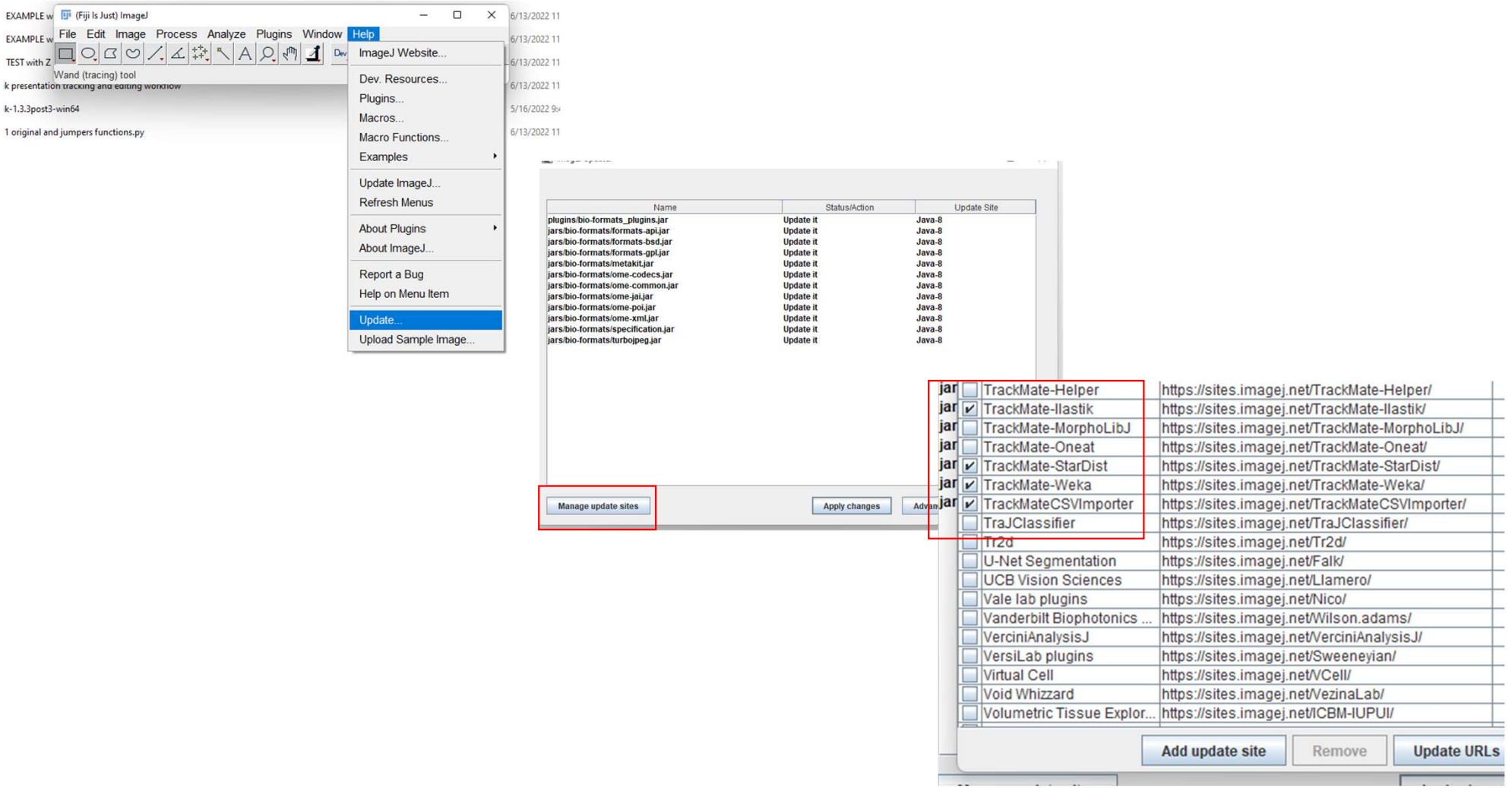
2022/06/13

Pixel Classification

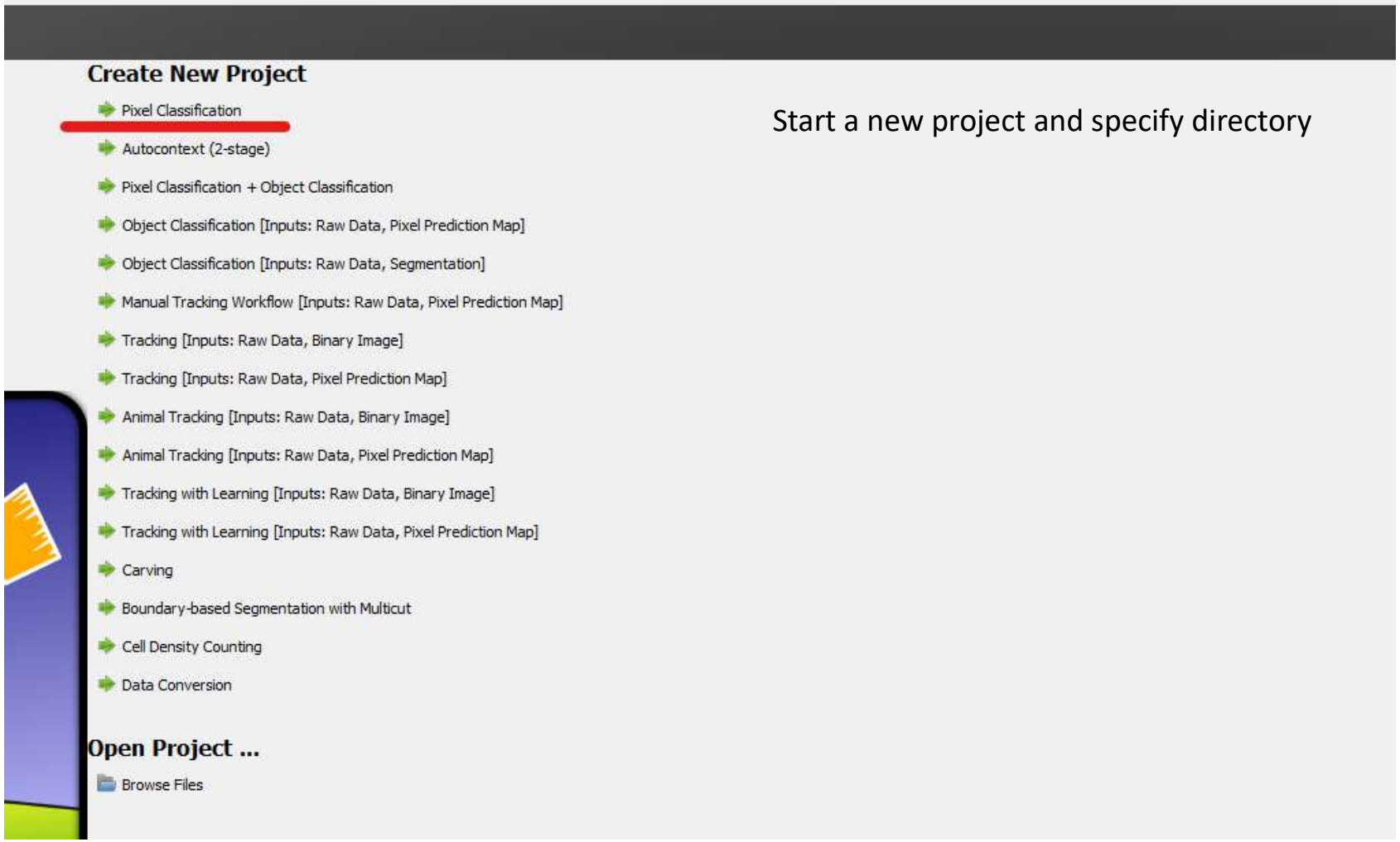
Tracking with Learning (Pixel Prediction Map)

Manual Tracking Editing with Custom Python Script

Install Ilastik from this file folder
Install TrackMate CSV Importer



Puncta
Pixel Classification

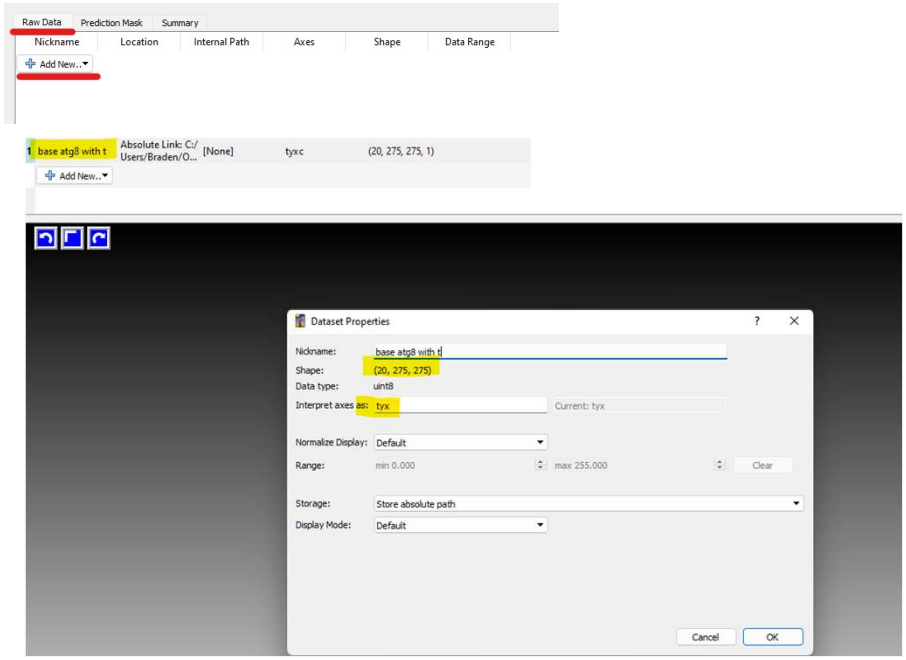


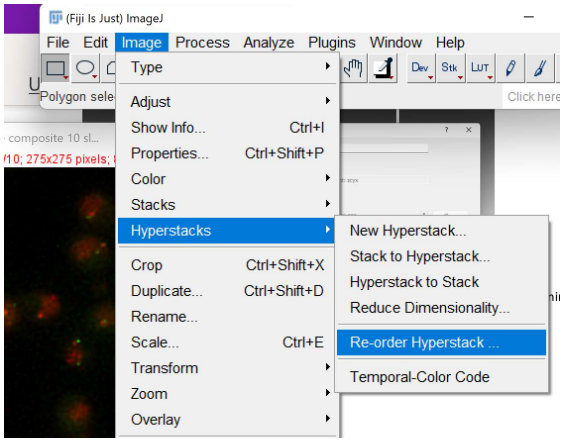
Start a new project and specify directory

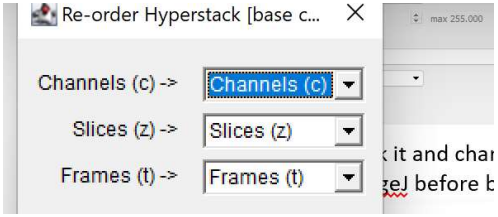
Import your image file

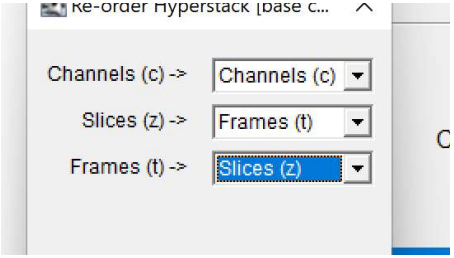
<u>Change Z to T by ‘re-order hyperstack’</u> in ImageJ

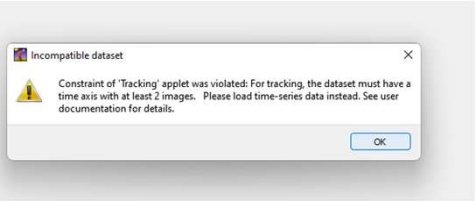
Then, your image should by ‘tyx’ (if 1 channel)



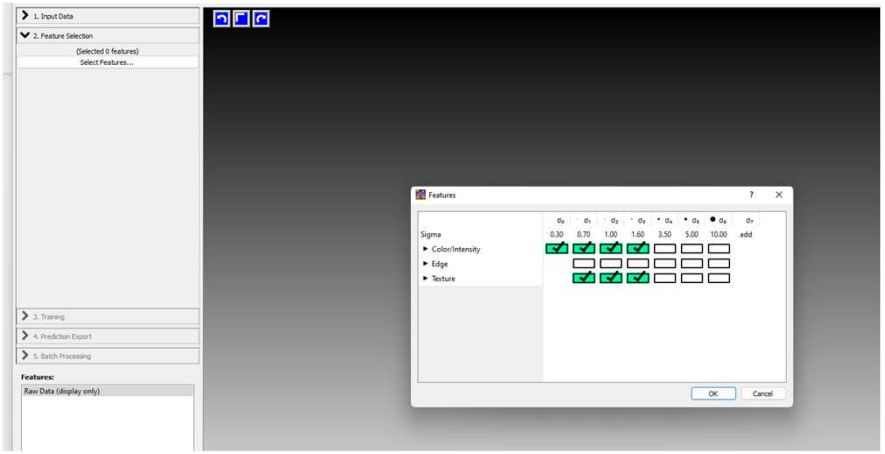






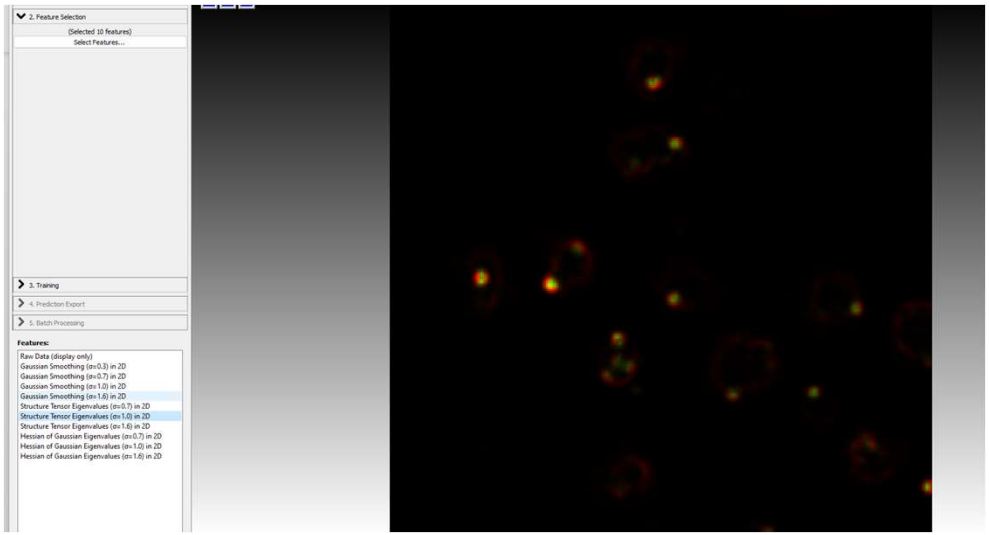


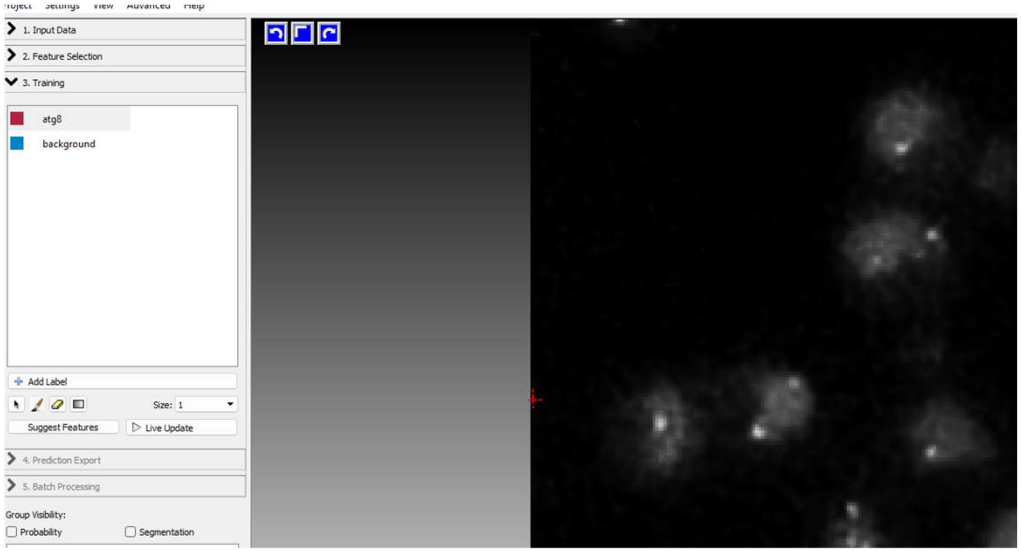
ERROR (in tracking) if you do not change Z→T



Choose **features**:
This relates to the size of your object
Because the **puncta are small**, we choose about just the first half of sigma values: **0.30 – 1.60**
If using **cells**, use **3.50-10.0**

Examine the selected features here. This is what the computer will be using for learning.

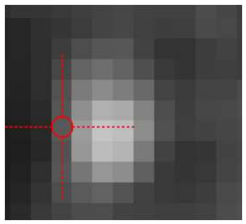


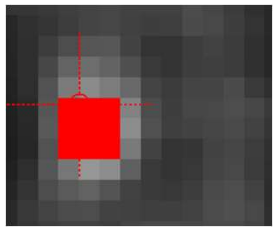


Change color of your objects by double-clicking the color

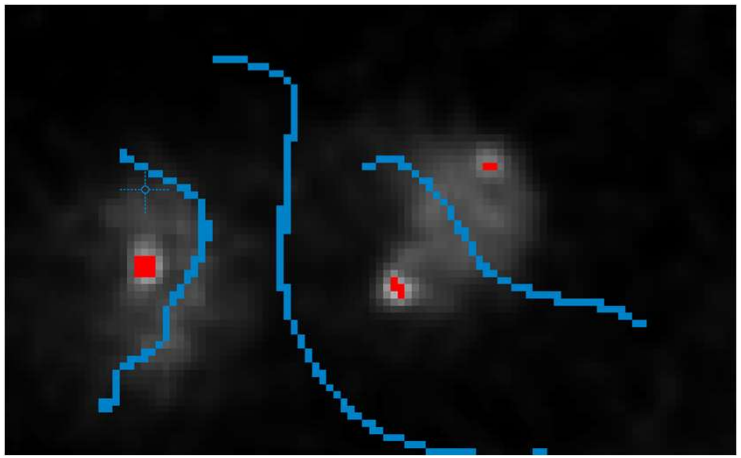
Paint the area that is your ***specific object***

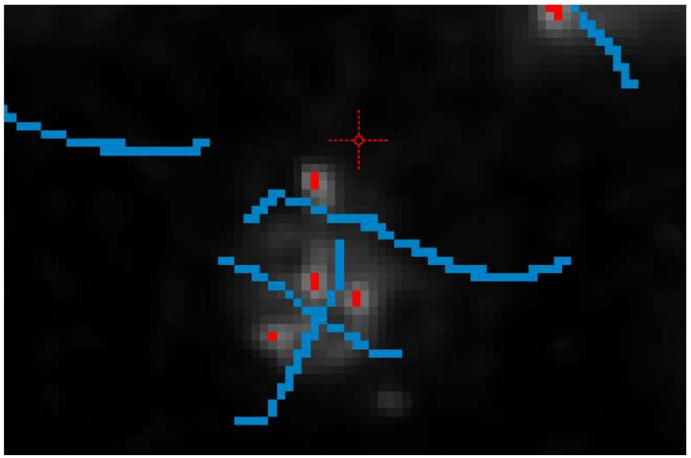
Paint the **background** ***everywhere else***

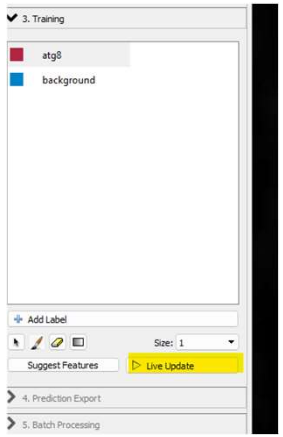


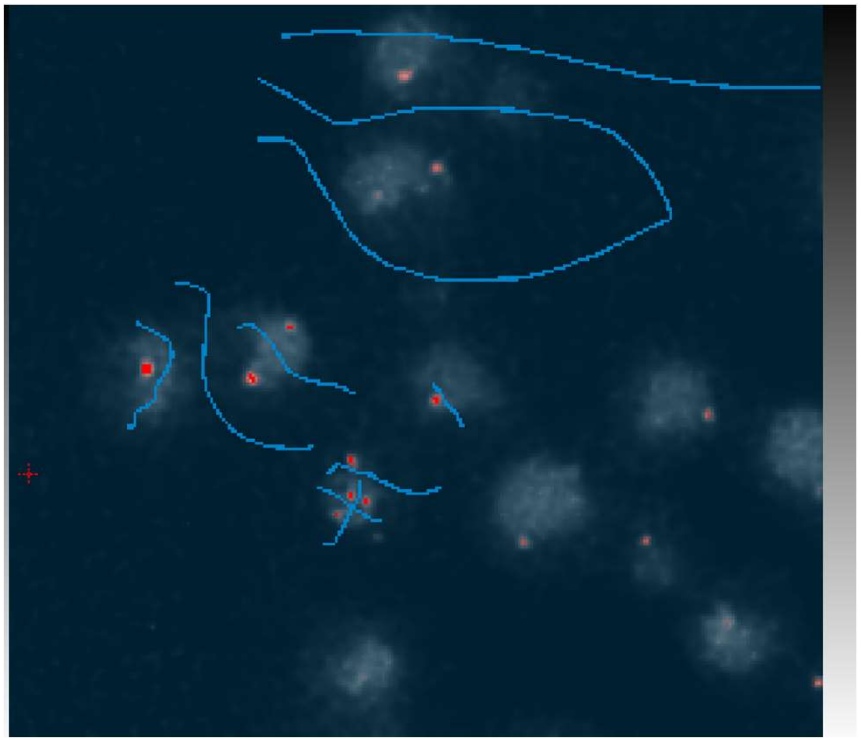


Click ‘live update’ to see the first learning







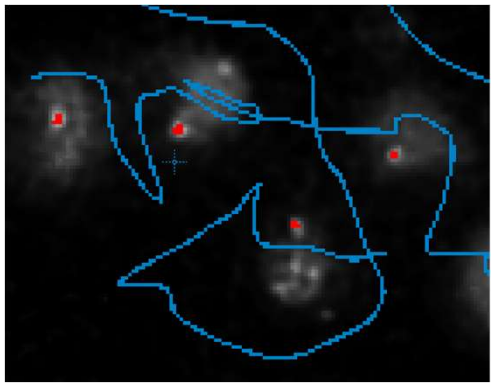


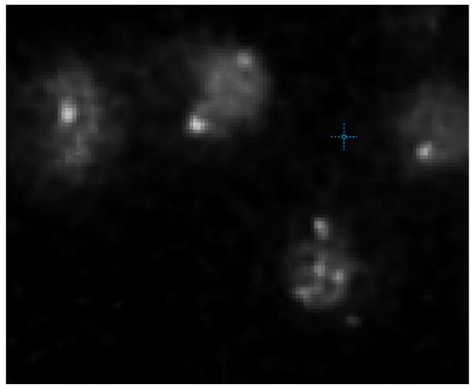
Live Update:
You can keep this always on.

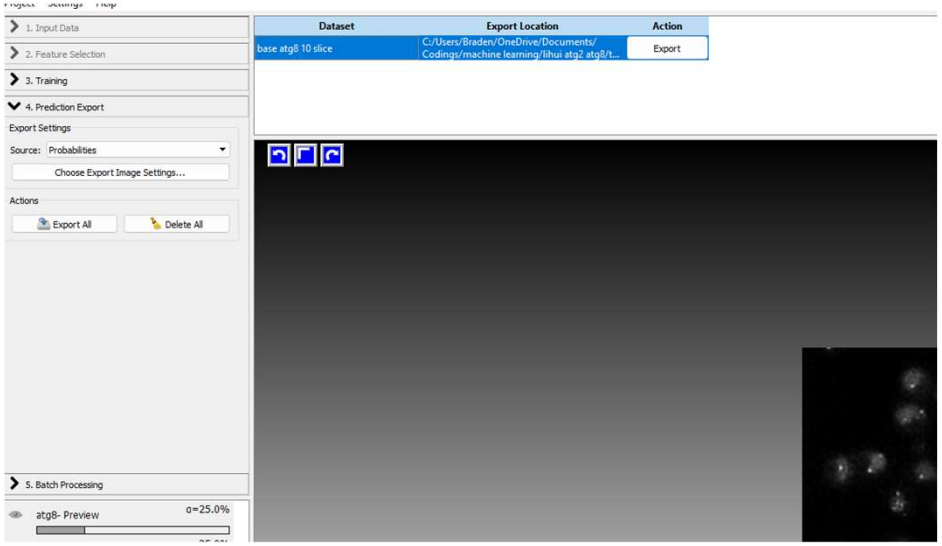
L

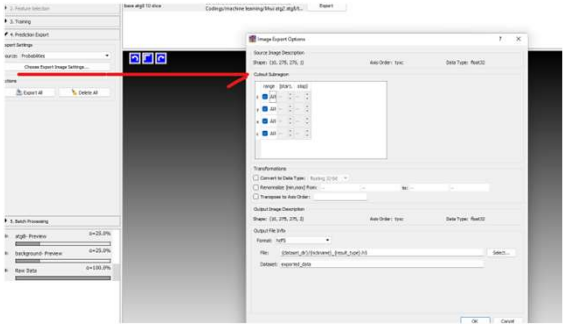
N B E T

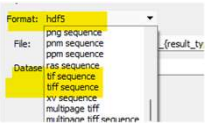
Switch between original and annotated: i

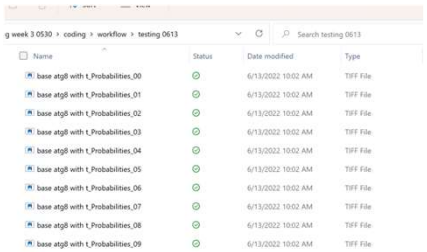












Export:
'Probabilities' = 'Pixel-Prediction Map'
'Simple Segmentation' = 'Binary Image'

Hdf5 = BEST format for ilastik (and large datasets)

Tiff sequence = only good for viewing image in imageJ—exports the image ONE by ONE

HDF5 export format

HDF5 is a flexible cross platform binary data format. Using Python, you can access the data inside the `h5` files using the h5py library.

The `h5` file contains two items at the root level, `images` and `table`. Using h5py, you can access them like a python dictionary:

```
import h5py
data = h5py.File("path/to/data.h5", "r")
data.keys()
# <KeysViewHDF5 ["images", "table"]>
```

In the `images` group you will find a subgroup for each object (identifiable by `object_id`) which contains two datasets: `labeling` (cutout of the segmentation showing the object) and `raw` (cutout of the raw data showing the object).

```
data["images"].keys()
# <KeysViewHDF5 ["0", "1", "2", "3", ...]>
```

Access individual the `raw` and `labeling` images for each object using the key for the object:

```
data["images"]["1"].keys()
# <KeysViewHDF5 ["labeling", "raw"]>
data["images"]["1"]["labeling"]
# <HDF5 dataset "labeling": shape (28, 26, 13), type "|u1">
# stored as a numpy array, access the data using numpy indexing:
data["images"]["1"]["labeling"][:]
```

[ilastik - Object Classification]

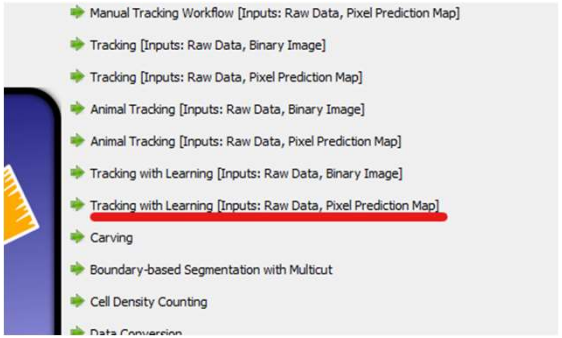
[https://www.ilastik.org/documentation/objects/objects.html]

(ctrl + f : 'hdf5')

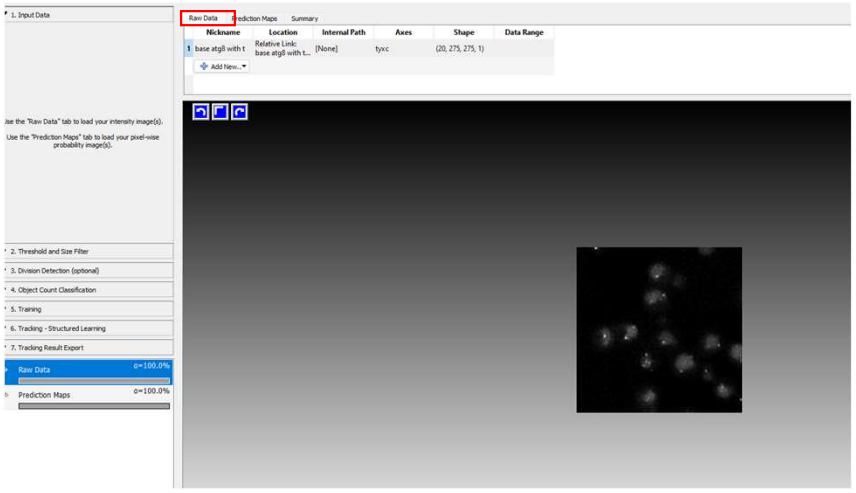
This is not necessary to use now.
Later, we can see if it's helpful.

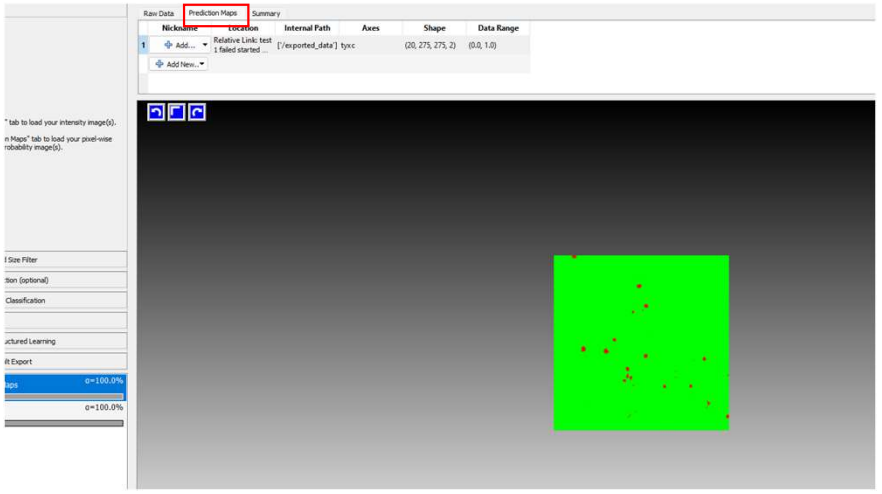
Working with CSV is easiest and convenient for all lab members.

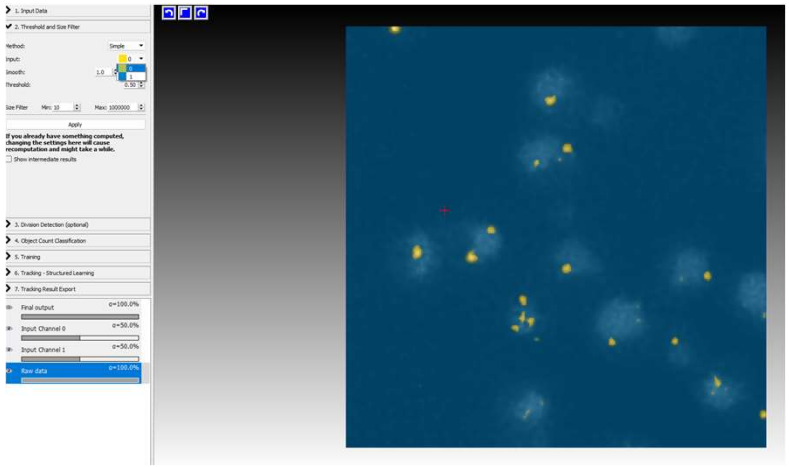
Tracking with Learning



Import your previous image
Import your exported prediction/probability image





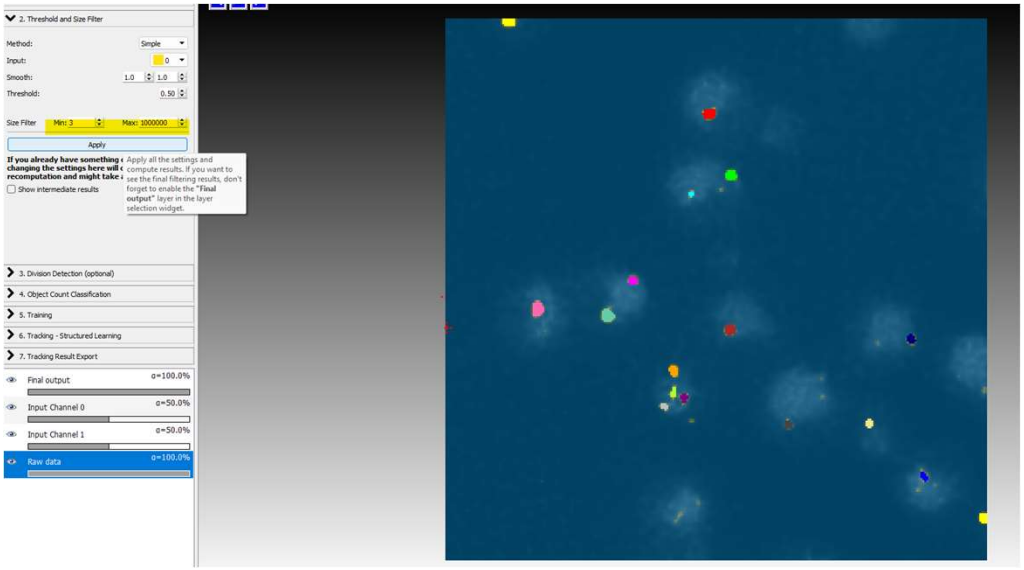


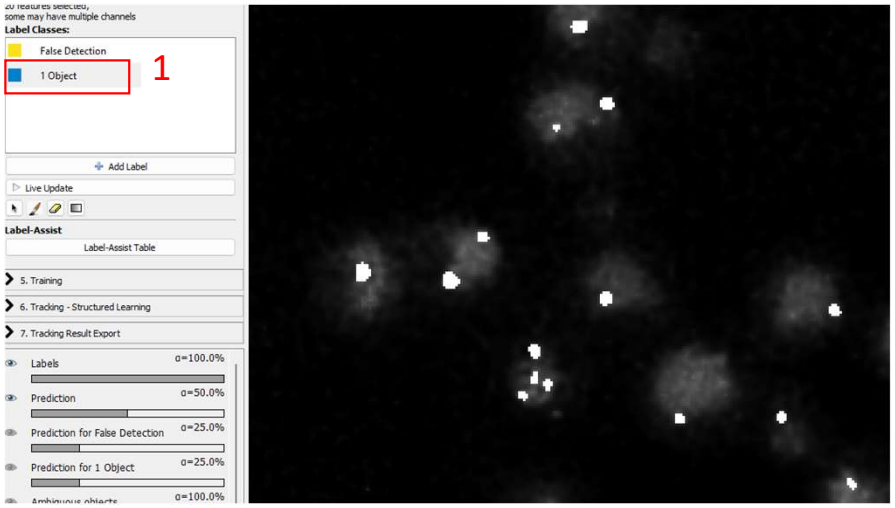
You will see an “input” selection. These are your background and puncta. We will only focus on the puncta color.

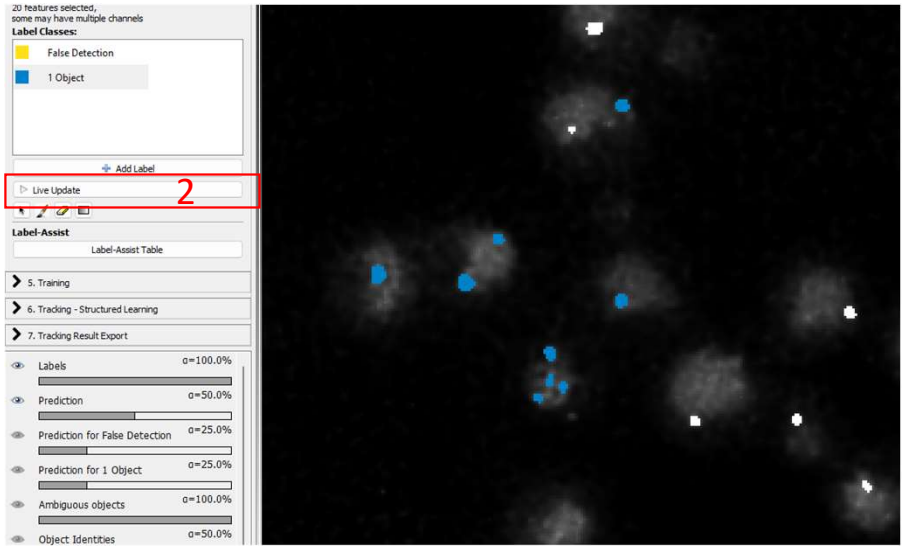
I started a new test, so my atg8 is now yellow instead of red.

Select a **<u>low minimum value</u>**. These **<u>puncta</u>** are only a few pixels. So, I set the min to just **<u>3 pixels</u>**. Then, click “apply”, and we are finished with this part.

SKIP “Division Detection”
<u>GO TO “Object Count Classification”</u>

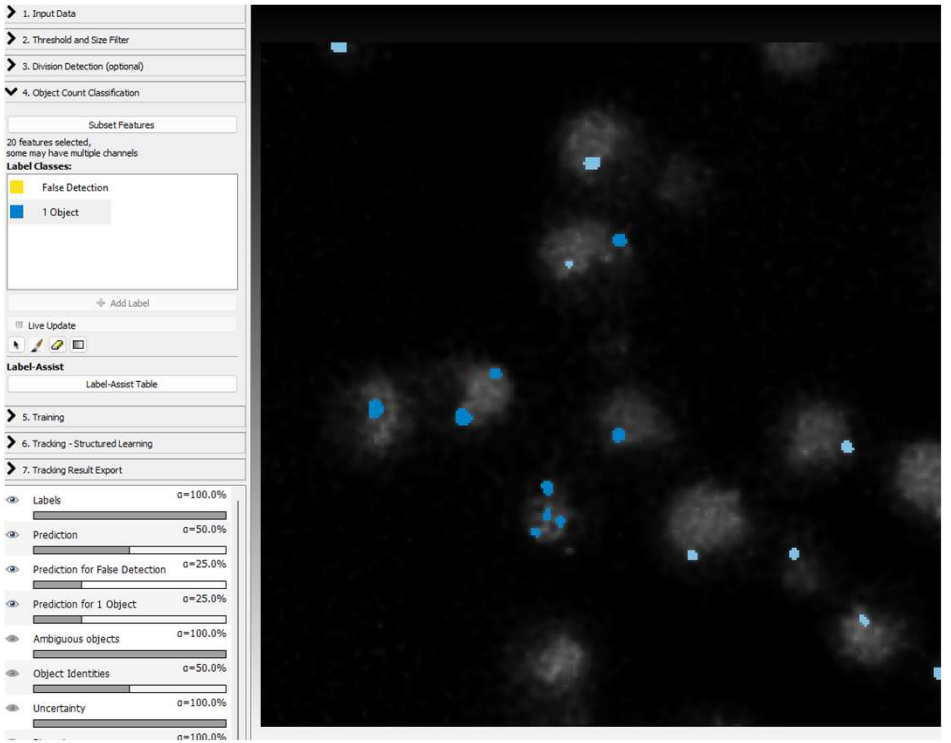


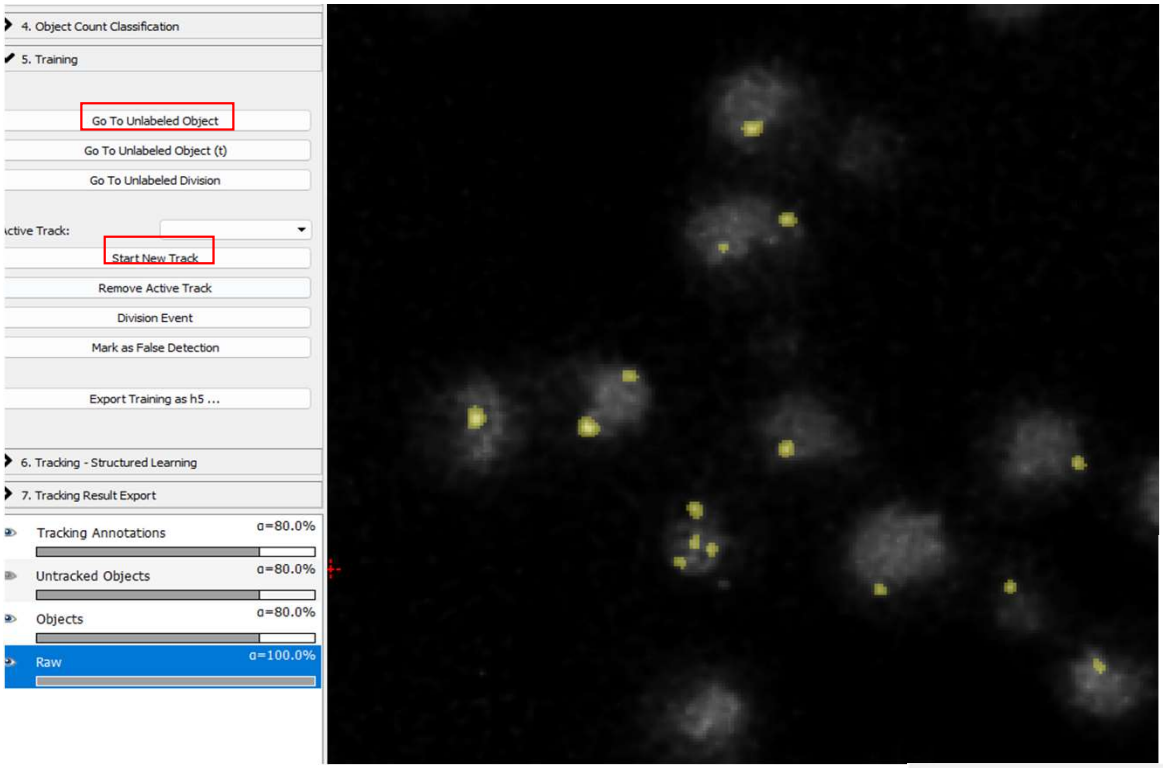




Select a few objects, and then do the “live update”.
Now, we can move to “training”.

3

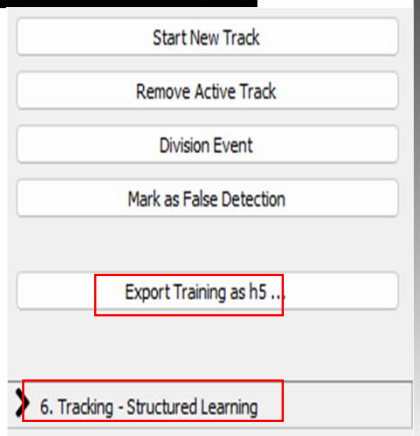


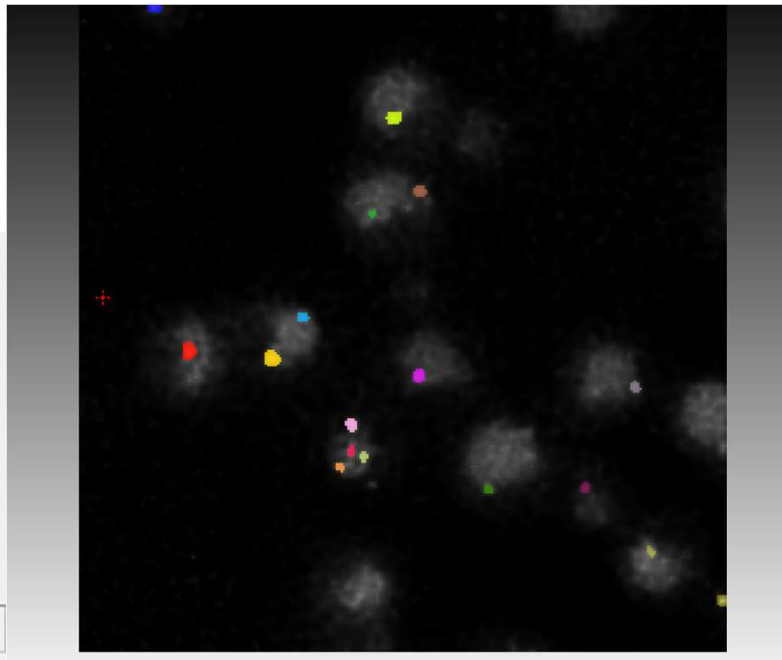


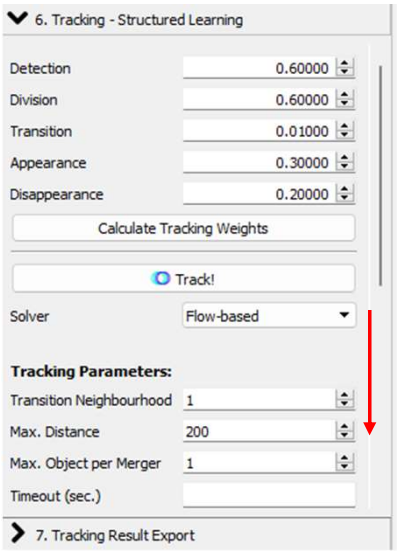
Label your tracks frame by frame
It is good to make about **<u>10-20 tracks</u>** to increase the accuracy of predictions

In 'Tracking', Ilastik will not use these like a binary image. The program will make new tracks based on these training data.

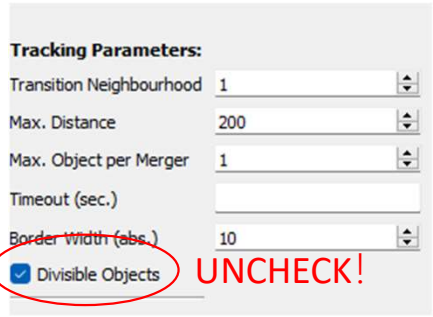
TIP:
press 'S' to start a new track
Press 'E' to turn off labeled tracks
Press 'R' to turn off prediction
*'**I**' has no function here*

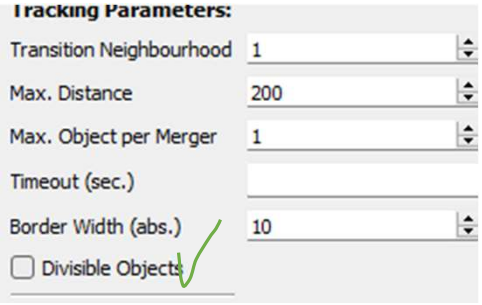


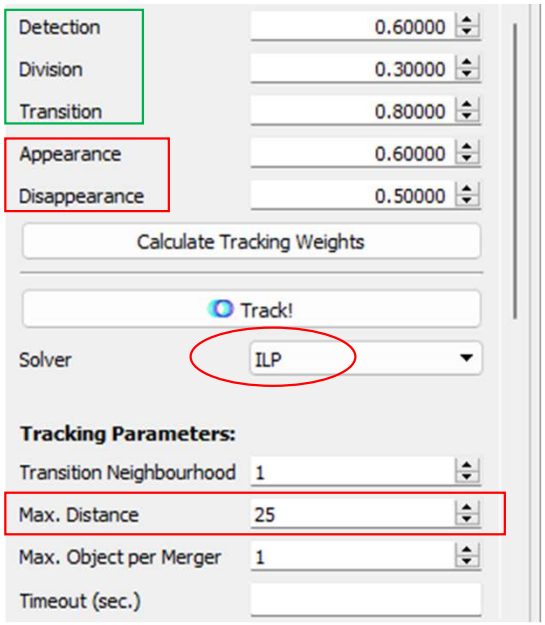




1st: Turn **OUT ‘Divisible Objects’** or the tracking won’t work







These 4 parameters have the most effect.
Appearance: if this is large, it will be **harder to make new tracks**
Disappearance: if this is large, it will be **harder to end a track** (it will try to ‘survive’)
Solver: The type of algorithm for calculating the tracks. This worked better for me than ‘Flow’ with puncta.
Max. Distance: This is the max distance that the puncta can travel between frames before it will be seen as a new track.

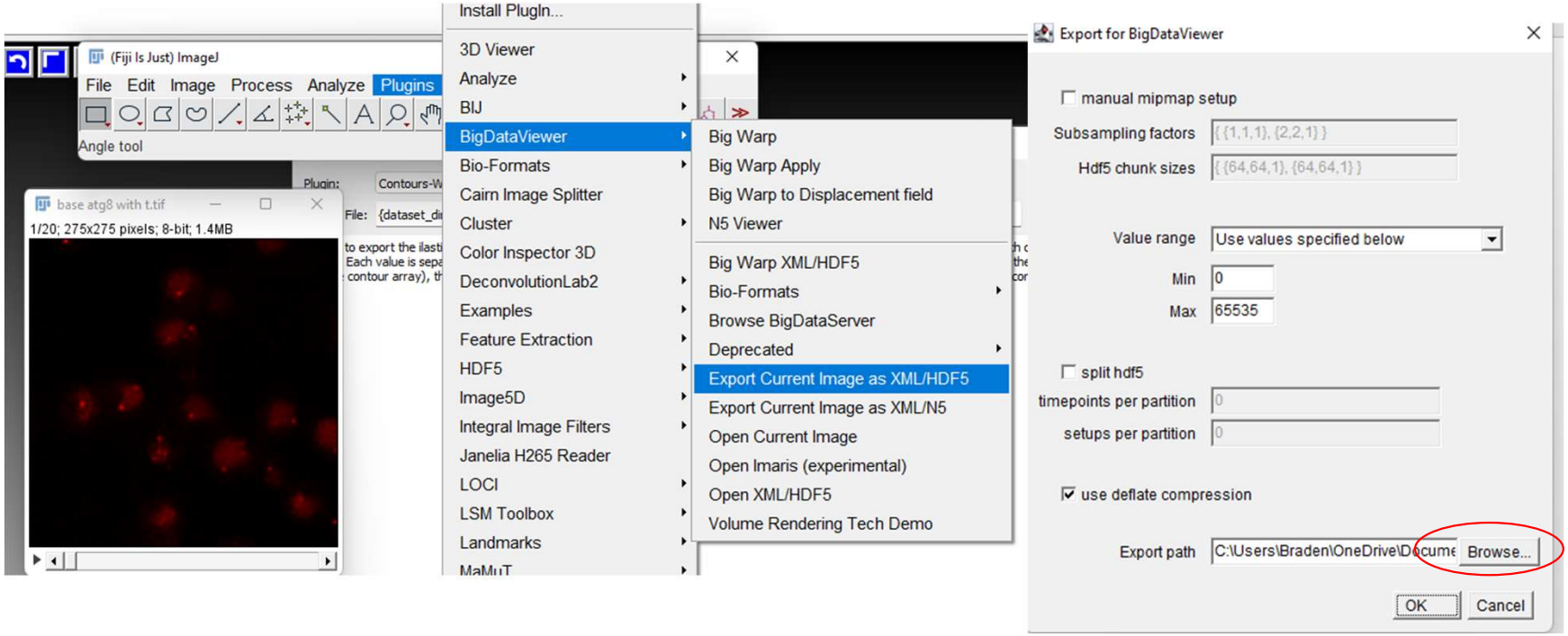
Example:

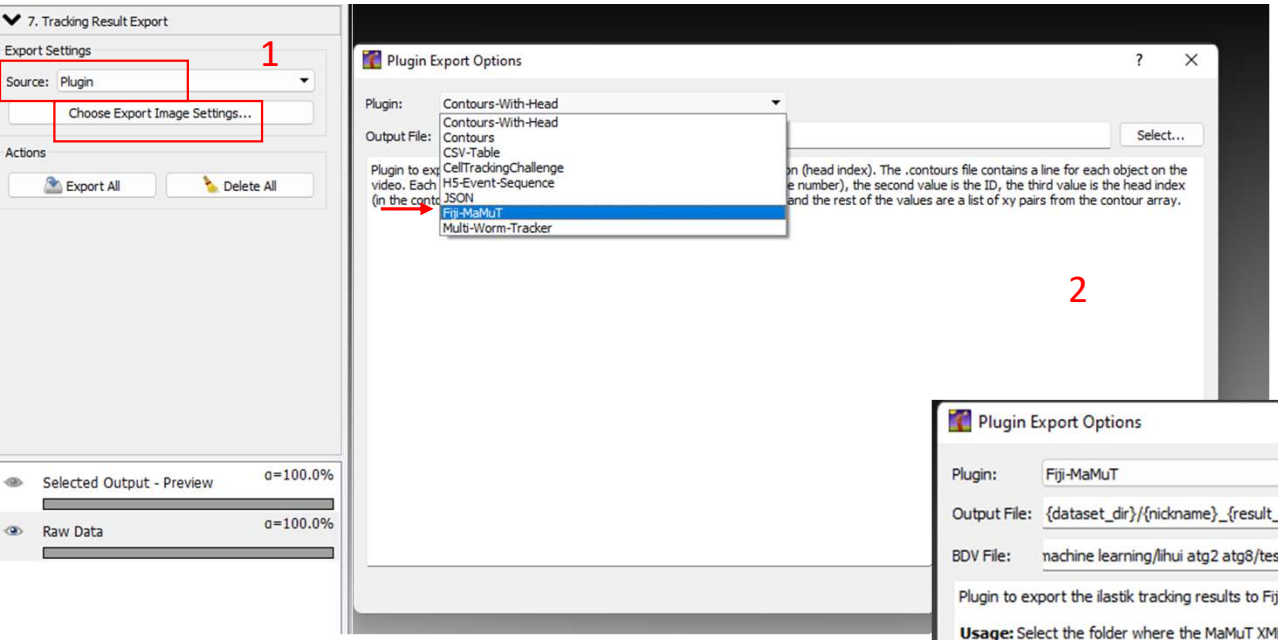
The cell size is about 30-40 pixels
The max. distance is ***at most*** 40
This is usually smaller when we look at the activity in each cell

This effect seems less than above
But: the ***larger*** of these *three* ***values*** will have the ***strongest*** effect
Detection: > Division/Transition will favor detection
Transition: > detection/division will favor transition (change current track to new track)

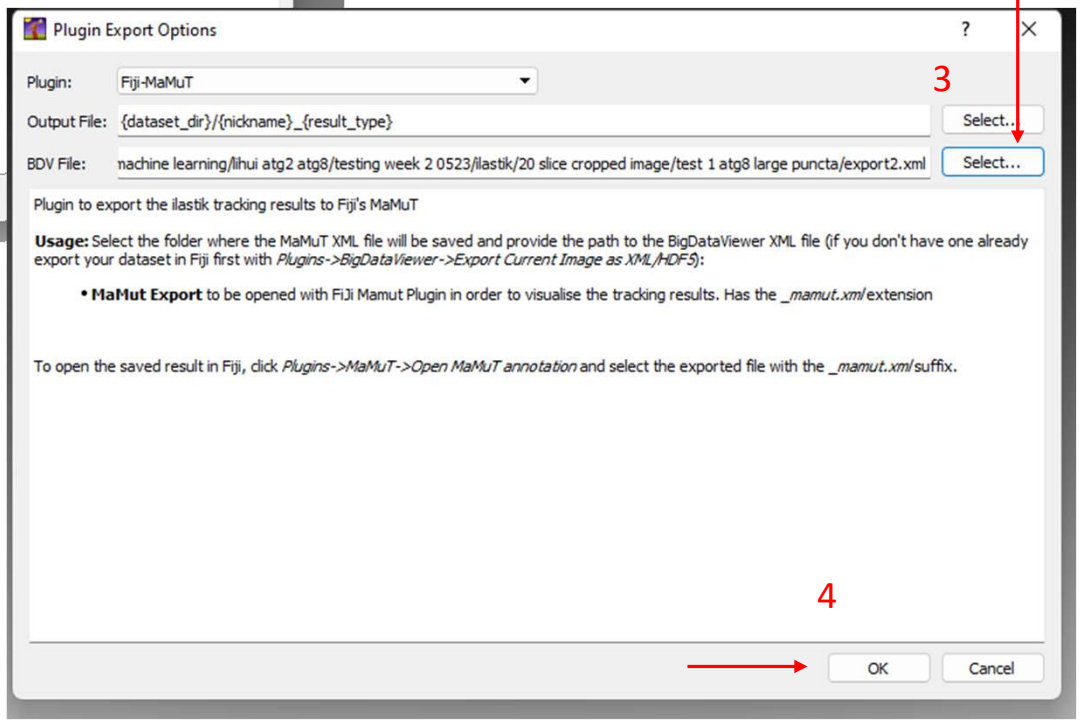
Division: *we don’t use it because no cells are dividing*
e.g.: **transition > detection > division**
****Transition** helps separate nearby tracks—adjusting this can be helpful

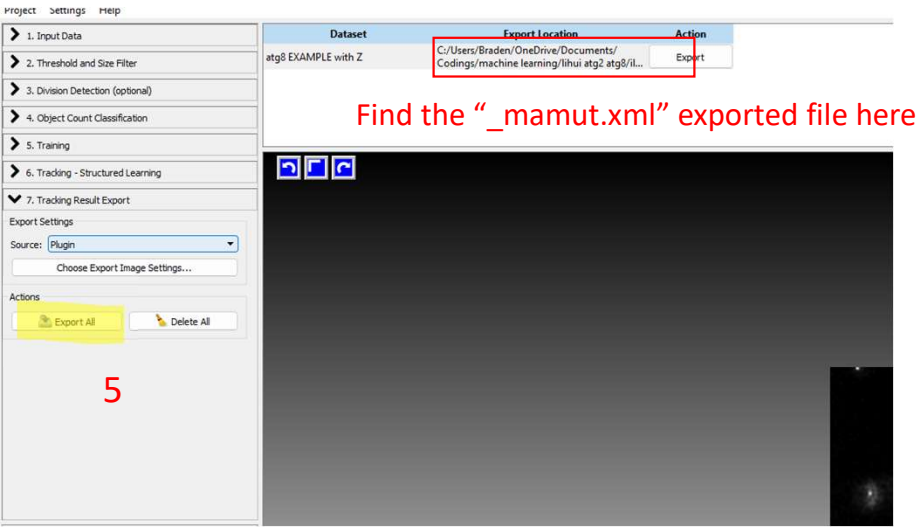
BEFORE EXPORT:
Open original image (**with T**) and export with BigDataViewer as XML/HDF5—choose the location. **File = export.xml**



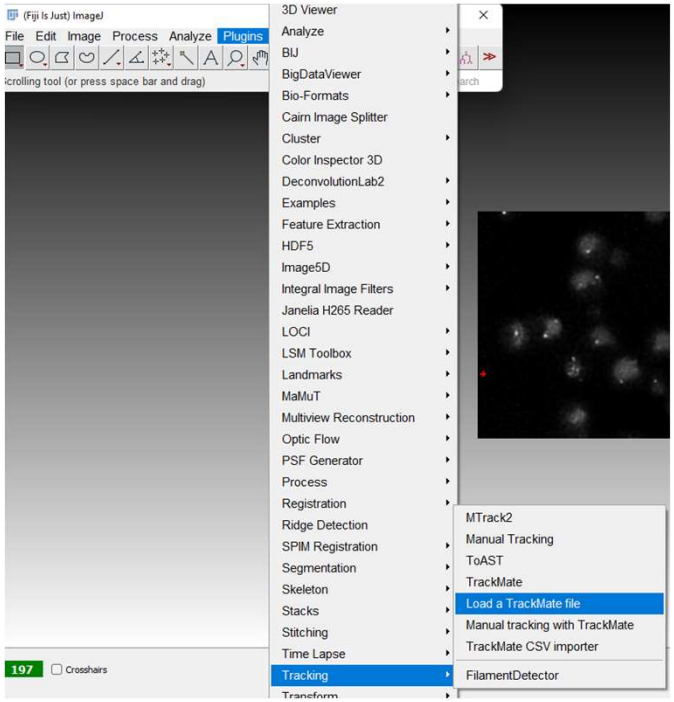


Export Tracking Results:
Source: PLUGIN
Plugin: Fiji-MaMut
BDV file: export.xml of the original image

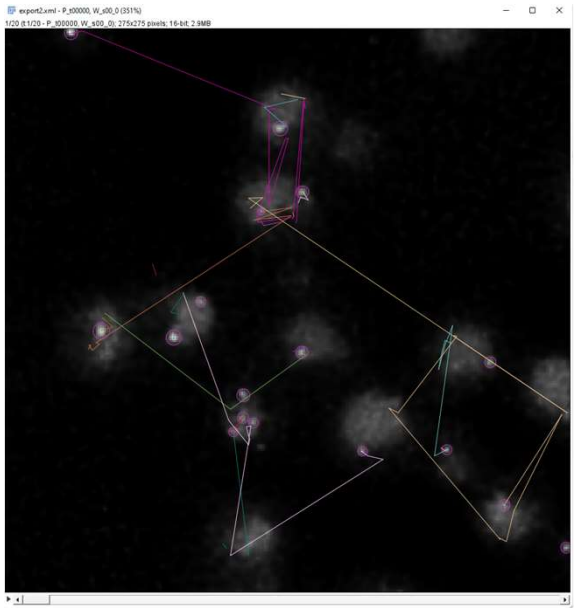


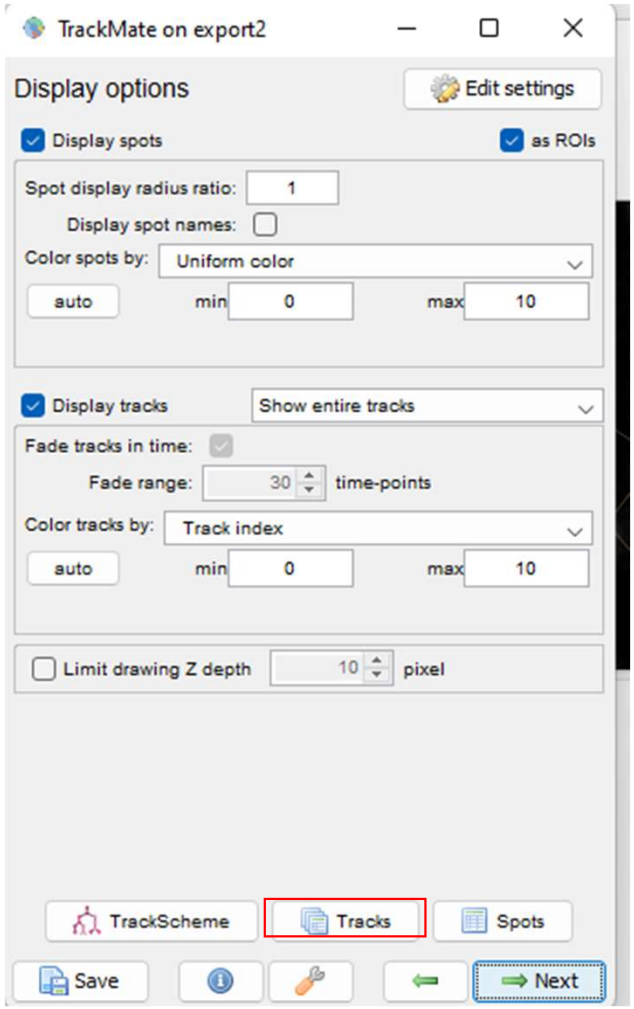


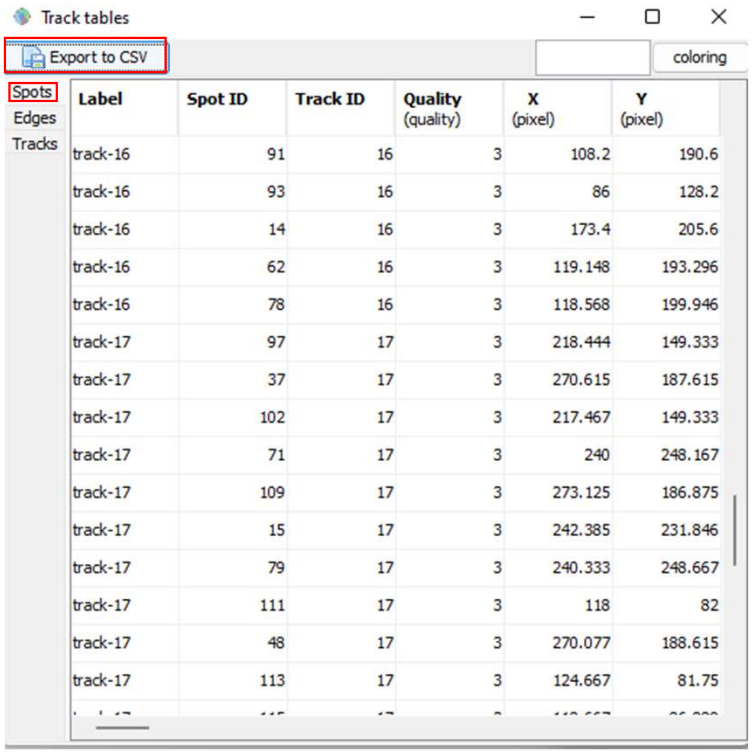
Find the "_mamut.xml" exported file here


(Fiji Is Just) ImageJ
File Edit Image Process Analyze Plugins
Scrolling tool (or press space bar and drag)
197
Crosshairs
3D Viewer
Analyze
BIJ
BigDataViewer
Bio-Formats
Cairn Image Splitter
Cluster
Color Inspector 3D
DeconvolutionLab2
Examples
Feature Extraction
HDF5
Image5D
Integral Image Filters
Janelia H265 Reader
LOCI
LSM Toolbox
Landmarks
MaMuT
Multiview Reconstruction
Optic Flow
PSF Generator
Process
Registration
Ridge Detection
SPIM Registration
Segmentation
Skeleton
Stacks
Stitching
Time Lapse
Tracking
Transform
MTrack2
Manual Tracking
ToAST
TrackMate
Load a TrackMate file
Manual tracking with TrackMate
TrackMate CSV importer
FilamentDetector


base atg8 with t_Fiji-MaMuT_mamut

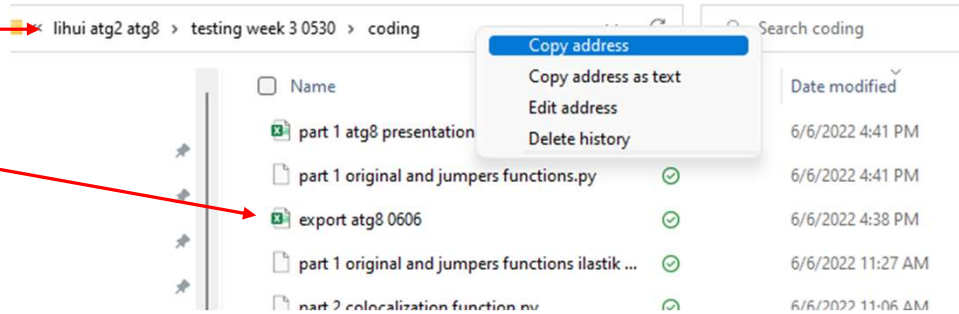






Label	Spot ID	Track ID	Quality (quality)	X (pixel)	Y (pixel)
track-16	91	16	3	108.2	190.6
track-16	93	16	3	86	128.2
track-16	14	16	3	173.4	205.6
track-16	62	16	3	119.148	193.296
track-16	78	16	3	118.568	199.946
track-17	97	17	3	218.444	149.333
track-17	37	17	3	270.615	187.615
track-17	102	17	3	217.467	149.333
track-17	71	17	3	240	248.167
track-17	109	17	3	273.125	186.875
track-17	15	17	3	242.385	231.846
track-17	79	17	3	240.333	248.667
track-17	111	17	3	118	82
track-17	48	17	3	270.077	188.615
track-17	113	17	3	124.667	81.75

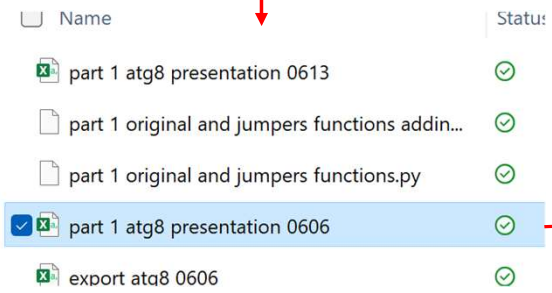
```
dir_in = 'C:/Users/Braden/OneDrive/Documents/Codings/machine learning/lihui atg2 atg8/testing week 3 0530/coding'

path_atg8 = os.path.join(dir_in, 'export atg8 0606.csv')
atg8 = pd.read_csv(path_atg8)

dis = 11

df = atg8[['LABEL', 'FRAME', 'POSITION_X', 'POSITION_Y']]
df = df.drop([0,1,2], axis = 0)

df['POSITION_Y'] = df['POSITION_Y'].astype(float).astype(int)
df['POSITION_X'] = df['POSITION_X'].astype(float).astype(int)
df['FRAME'] = pd.to_numeric(df['FRAME'], downcast = 'unsigned')

df['puncta']=None
```

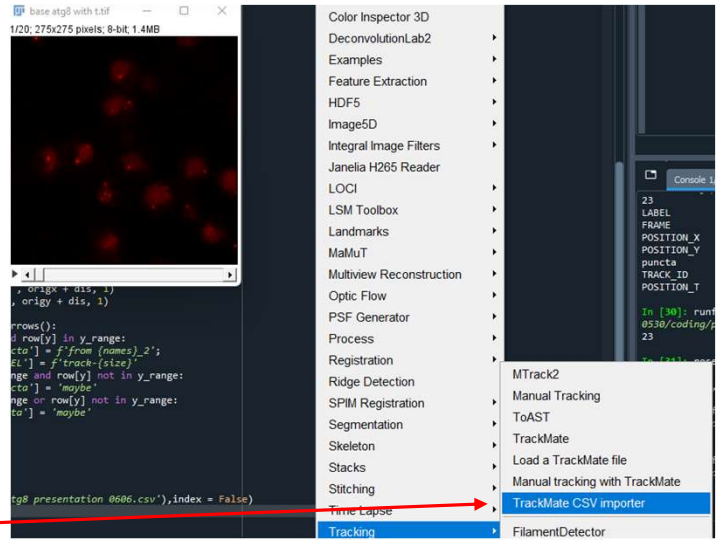


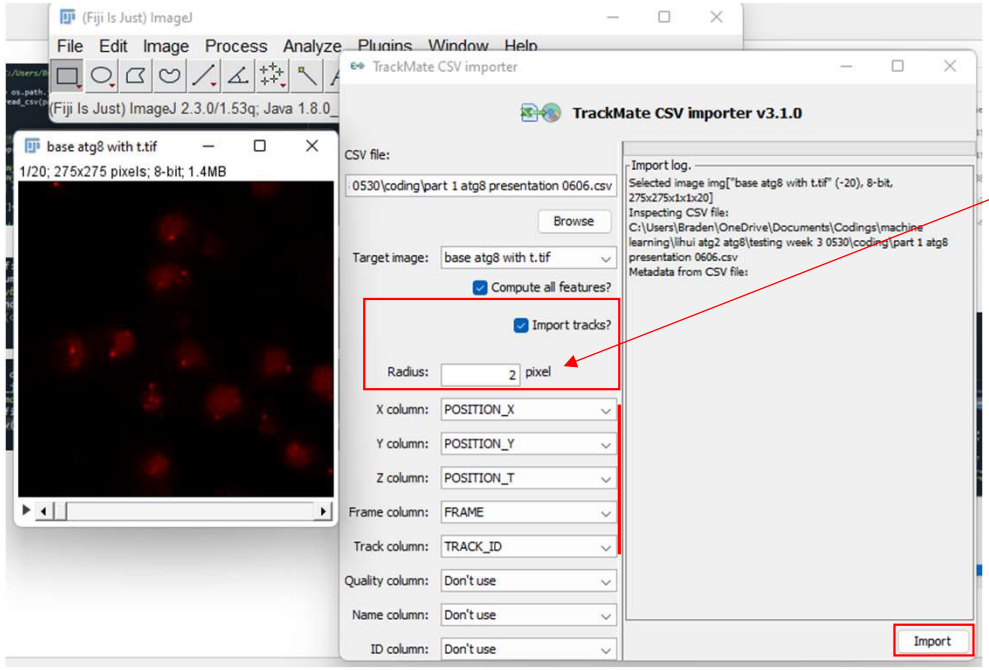
Max. distance track can travel in the next frame

```
for x in df:
    find_jumpers(df, 23)
    if 'maybe' in df['puncta']:
        find_jumpers(df, 23)
#df.to_csv(os.path.join(dir_in,'part 1 atg8 presentation 0606.csv'),index = False)
```

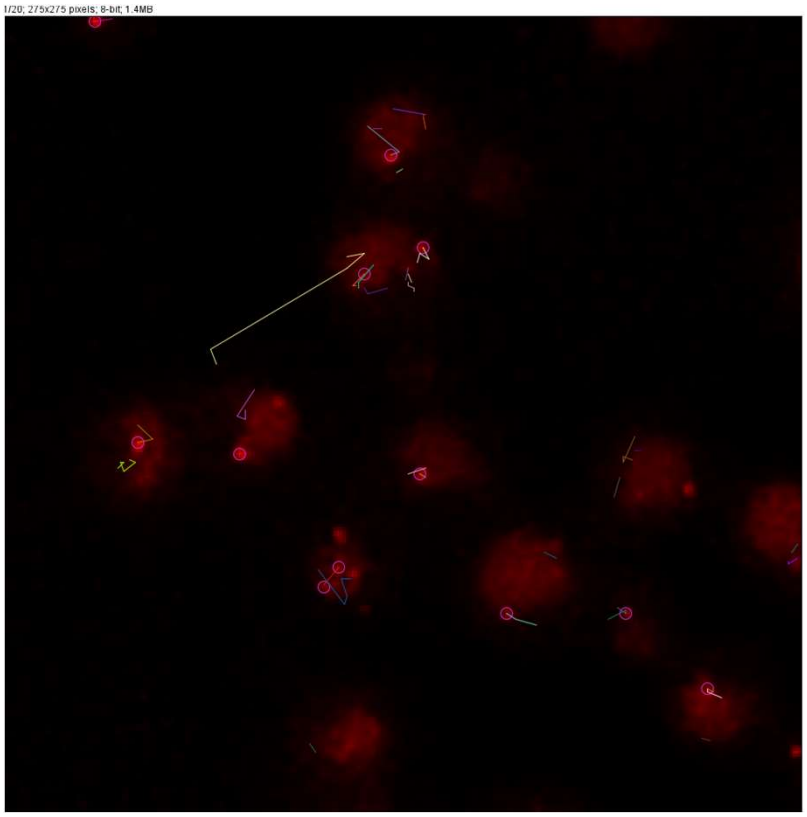
```
                return 'invalid'
for x in df:
    find_jumpers(df, 23)
    if 'maybe' in df['puncta']:
        find_jumpers(df, 23)
df.to_csv(os.path.join(dir_in,'part 1 atg8 presentation 0606.csv'),index = False)
```



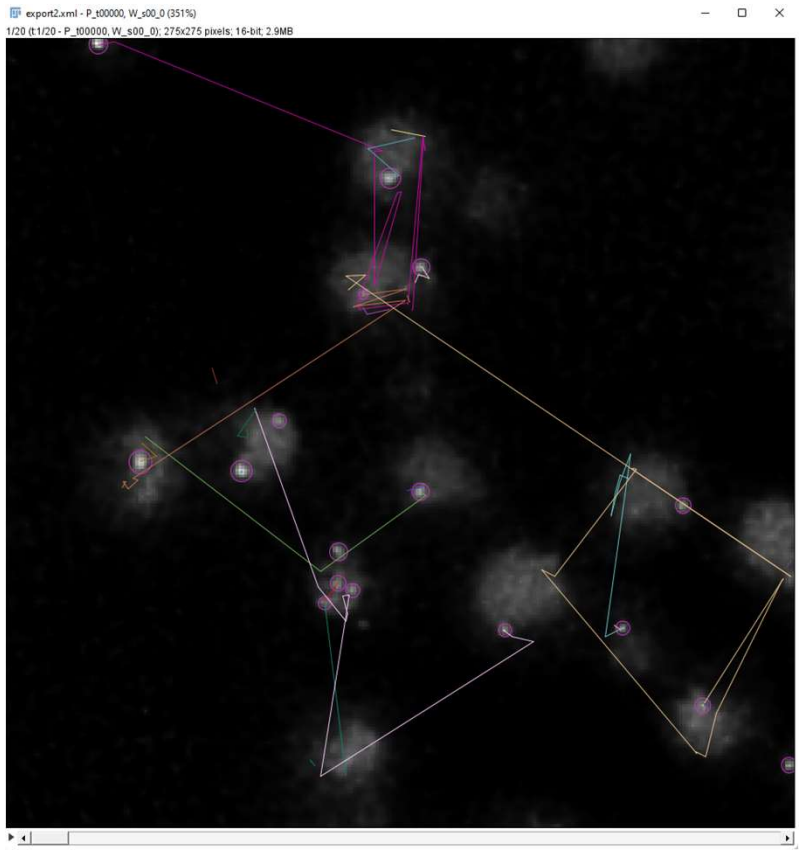




Make sure this is an integer with no decimal value!

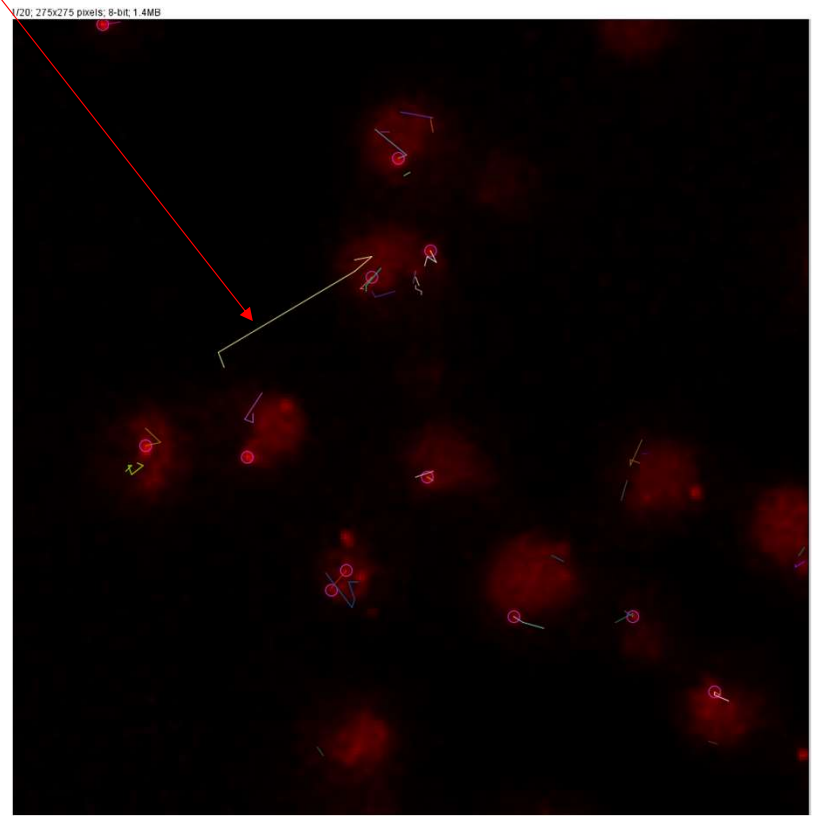


Original, no function



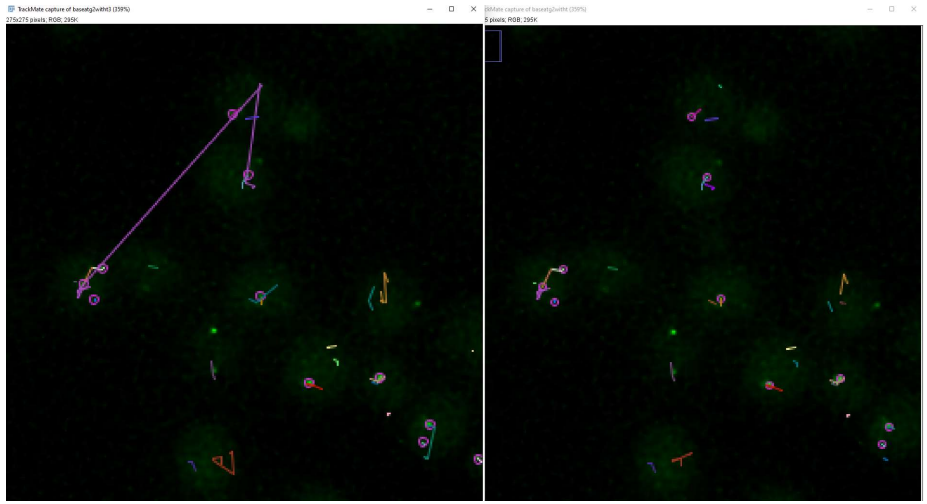
Probably max. distance problem

With function
(original tracking parameters)

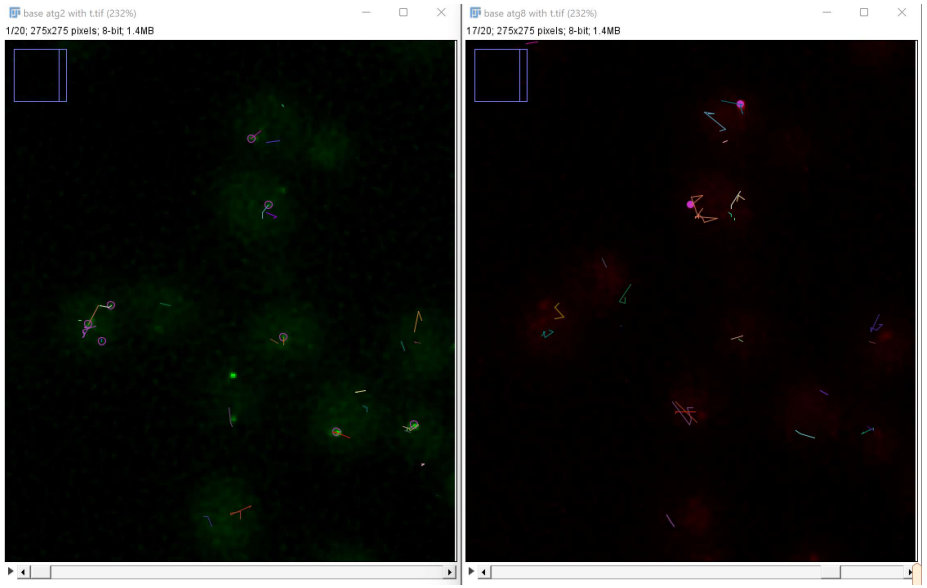


Original, no function

With function
(with modification to tracking parameters)



With function
(with modification to tracking parameters)



Index	LABEL	FRAME	POSITION_X	POSITION_Y	puncta	TRACK_ID	POSITION_T
3	track-32	18	145	33	original	32	18
4	track-32	19	134	31	original	32	19
5	track-33	18	71	114	original	33	18
6	track-33	19	73	119	original	33	19
7	track-2	0	31	1	original	2	0
8	track-43	11	122	93	from track-43	43	11
9	track-39	6	137	52	from track-39	39	6
10	track-36	12	139	91	from track-36	36	12
11	track-39	7	135	53	from track-39	39	7
12	track-48	3	130	38	from track-48	48	3
13	track-36	13	139	92	from track-36	36	13

```
    data['POSITION_T'] = data['FRAME']
    print(tracked)

find_original(df)

def find_jumpers (data, tracked):
    jumpers = data[data['puncta'] != 'original']
    label = jumpers.groupby(by = ['LABEL'])
    size = tracked + 10
    for tracks, series in label:
        track = series.sort_values(by = 'FRAME', ascending = True)
        x, y, t = 'POSITION_X', 'POSITION_Y', 'FRAME'
        indice = track.loc[track['FRAME'] == track[t].min()]
        origy = int(track[y][indice.index])
        origx = int(track[x][indice.index])
        x_range = range(origx - dis , origx + dis, 1)
        y_range = range(origy - dis, origy + dis, 1)
        names = tracks
        for index, row in track.iterrows():
            if row[x] in x_range and row[y] in y_range:
                data.at[index, 'puncta'] = f'from {names}';
                data.at[index, 'LABEL'] = f'track-{size}'
            elif row[x] not in x_range and row[y] not in y_range:
                data.at[index, 'puncta'] = 'maybe'
            elif row[x] not in x_range or row[y] not in y_range:
                data.at[index,'puncta'] = 'maybe'
            else:
                return 'invalid'
```

The dataframe created with the functions fulfills the TrackMate requirements.

‘Puncta’ is helpful to the user. Original = the tracks Ilastik originally found.

‘From--##’ tells us if ilastik was wrong. If ## < ‘size’, then this is the first track that the function gives to new puncta after separating the tracking error from the original. If ## > ‘size’, then this was a second puncta tracking error, and the function created a *second new track*.

Other CSV filetypes:

```
FOR OTHER CSV FILES:
    x = 'POSITION_X'
    y = 'POSITION_Y'
    t = 'FRAME'
    FRAME = 'FRAME'
    TRACK_ID = 'LABEL'
    name = 'LABEL'

and then you can perform the following:
    df = pd.DataFrame(columns = [x, y, t, name, TARCK_ID, FRAME, 'puncta', 'LABEL'])
    df[x] = atg2['CENTER_OF_OBJECT_0'] etc...

    FOR POSITION_X, POSITION_Y, POSITION_T, FRAME we need to CONVERT from STRING to INT:
    workflow:
        df[x] = atg2['CENTER_OF_OBJECT_0']
        df[x] = df[x].astype(float).astype(int)
        etc...

    you then transforms the column data from the original CSV into a format which is more readable to trackmate
    i.e. track_in , position_t, position_x, etc...
    trackmate needs:
        track_id
        position_x
        position_y
        position_t (ideally) or z
        frame

        adding 'puncta' and 'label' is useful for checking the progress while in the python IDE
"""
```

You can use any tracking file; however, you just need to specify the five basic parameter values that TrackMate needs:

POSITION_X
POSITION_Y
POSITION_T
FRAME
TRACK_ID

Ilastik CSV export **(instead of MaMut)**

	N	O	
_H	Object_Center_0	Object_Center_1	Ob
.5	31.608696	1.434783	
48	133.300003	47.933334	
79	144.399994	79.040001	
39	124	88.714287	
5	94.647057	132.529419	

Example:
x = ‘POSITION_X’
y = ‘POSITION_Y’
Etc…

df = pd.DataFrame(columns = [x, y, t, name, TRACK_ID, FRAME, 'puncta', 'LABEL’])

df[x] = input_data[‘Object_Center_0’]
df[y] = input_data[‘Object_Center_1’]