Lecture 34: Visualizing tracking results and 2D curve-fitting

University of Colorado Boulder

MCDB/BCHM 4312/5312
Fall 2020

Outline

- Visualizing tracking results
- 2D curve-fitting
- Discussion on nearest-neighbor tracking (if we have time this week)

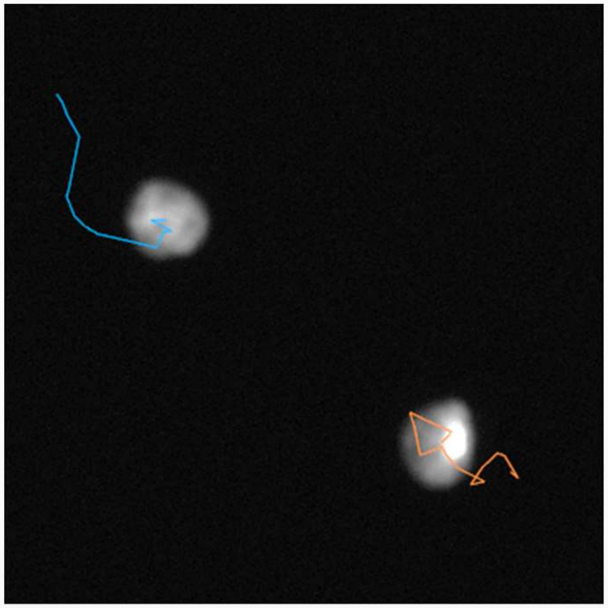
Visualizing tracking results

- You can combine multiple types of plots in MATLAB using the function `hold`

- Basic usage:
  1. Plot the first graph or image using imshow
  2. `hold on`
  3. Plot other graphs (but not images as imshow will cover old plots)
  4. `hold off`

Task

- Download the example tracking code from Canvas (or use your own from last week) and run the code
- Plot an image that shows the last frame of the movie, combined with line plots showing the position of the objects



Note: Using hold on, MATLAB will plot lines in different colors

You can use this figure to check that the tracks look correct

Analyzing time-series data

Analyzing time-series data

- Output of tracking code:

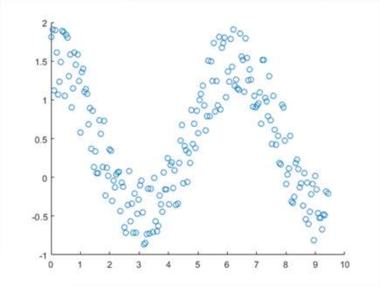
```
tracks(1).MajorAxisLength = [1 2 3 4 5];
tracks(2).MajorAxisLength = [4, 3, 2, 3, 2];
```

- The output is just matrices, which means you can use all the techniques we've learnt this year for analysis, e.g. plots, histograms, mean, std etc…

Some plot types you might find useful

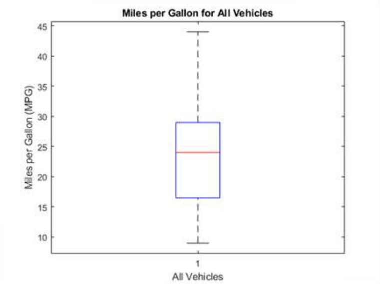
Scatter plot

```
scatter(xdata,
ydata)
```



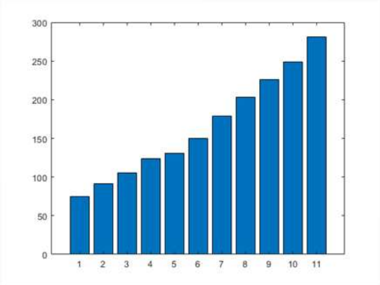
Box-and-whisker plot

```
boxplot(xdata)
```



Bar plot

```
bar(xdata, ydata)
```



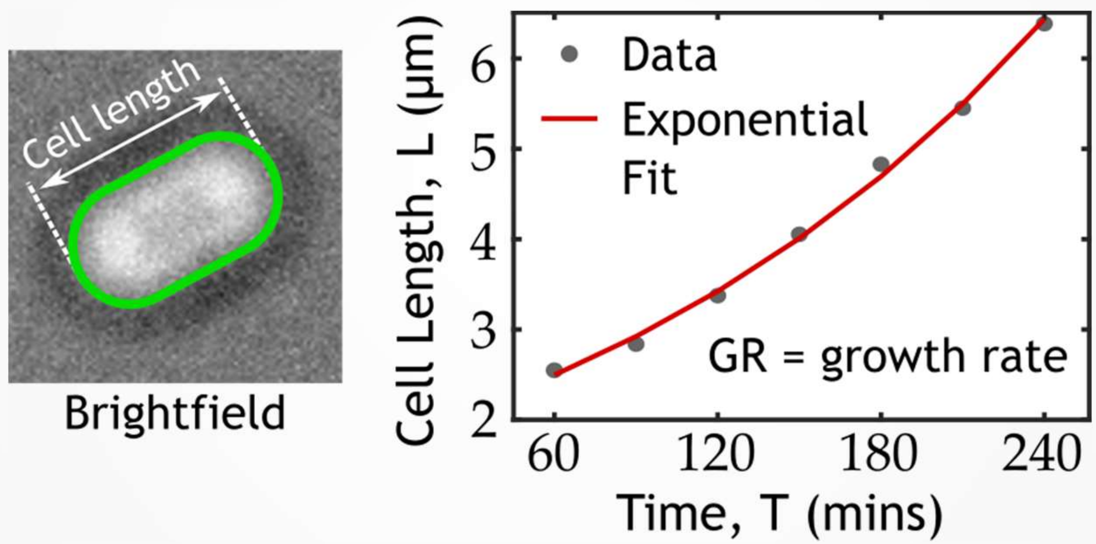
Not on final exam

2D curve-fitting

- Curve-fitting is an important skill in data analysis
- Fits data to a mathematical *model* (function)

Examples of curve-fitting applications

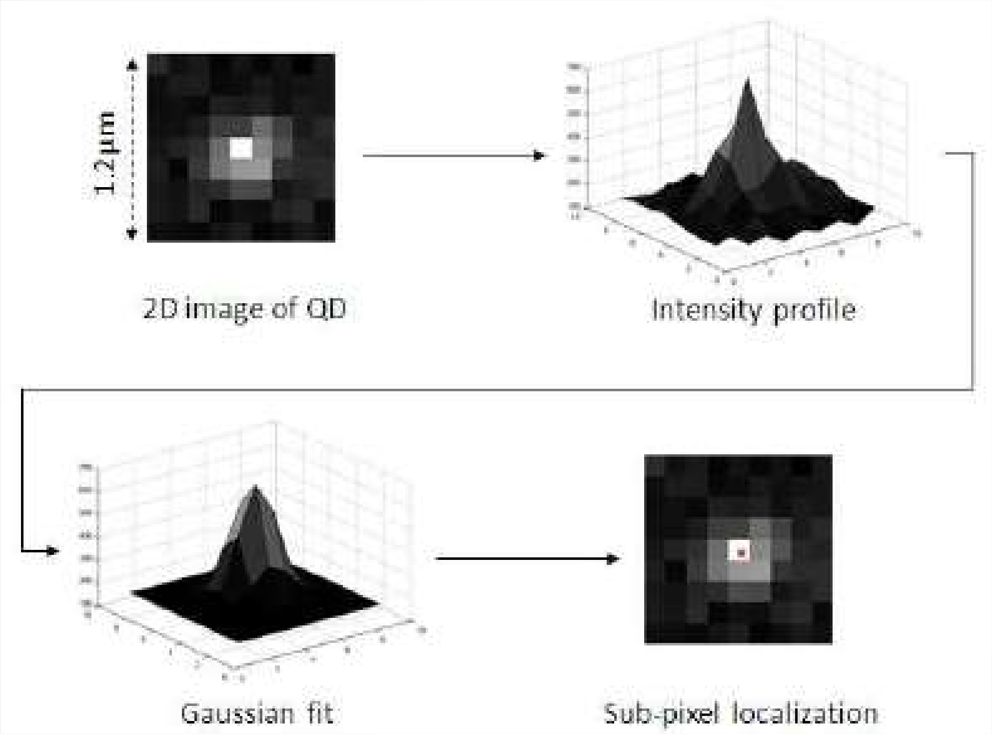
- Measuring the growth rate of cells – exponential function

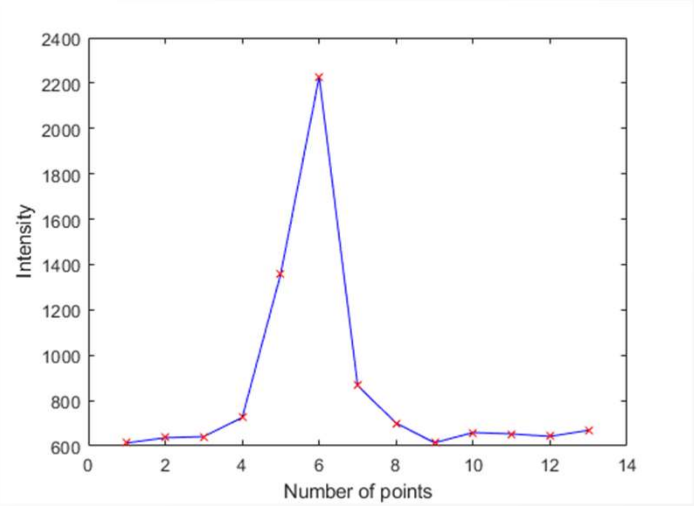


Example in PS10

Examples of curve-fitting applications

- Sub-pixel localization of particles – Gaussian function





The fitting function

```
F0 = fit(X, Y, fittype)
```

Inputs:

`X, Y` = **<u>COLUMN vectors</u>** containing data to fit to

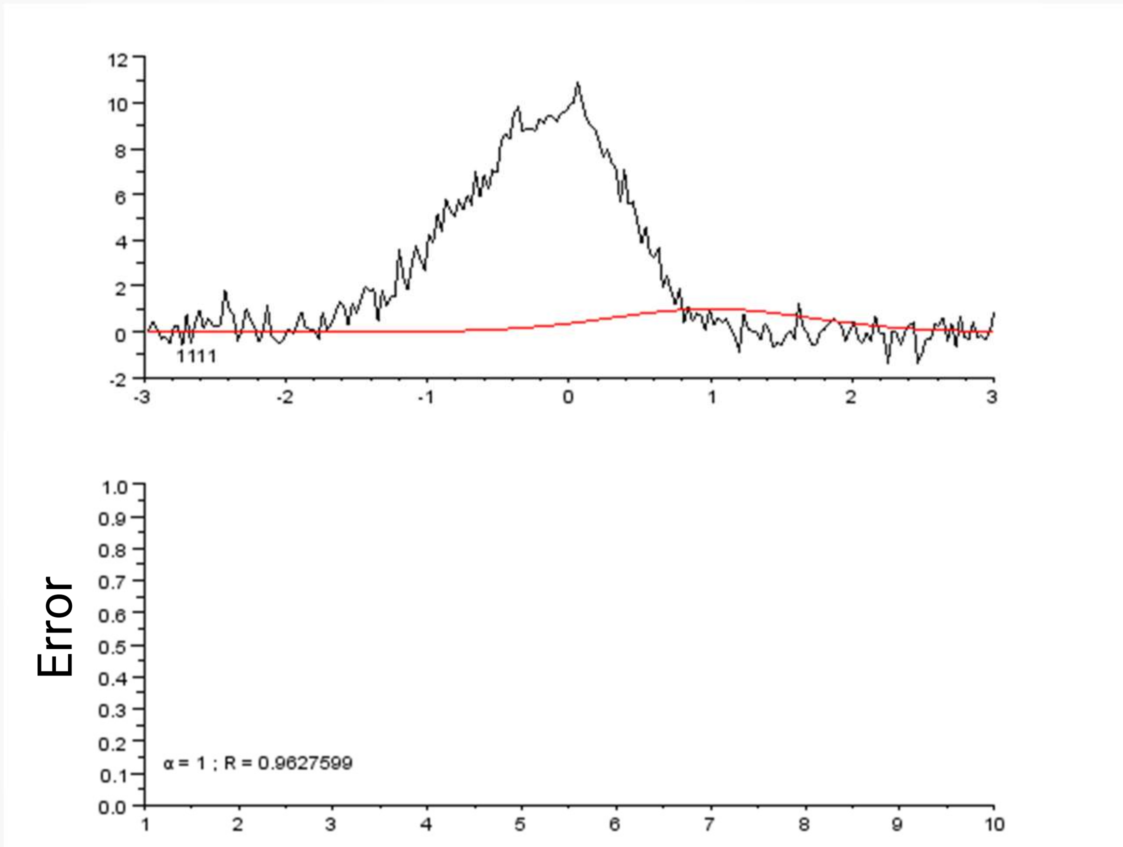
`fittype` = String describing model to fit

Output:

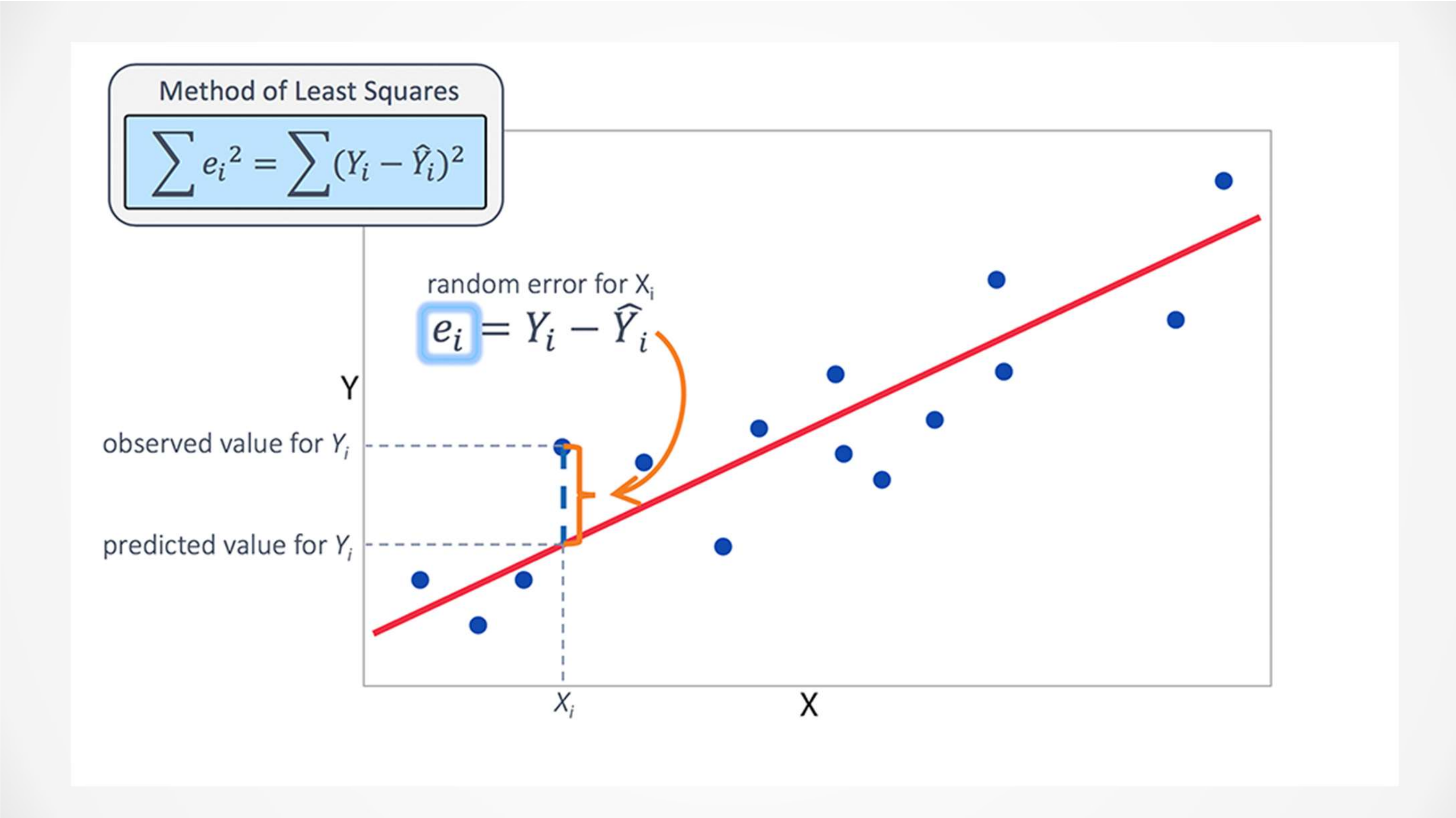
`F0` = Fit object

Curve-fitting works by changing variables to minimize error

Red line =
fitted model



Wikipedia


Method of Least Squares
$\sum e_i^2 = \sum (Y_i - \hat{Y}_i)^2$
random error for X_i
$e_i = Y_i - \hat{Y}_i$
Y
observed value for Y_i
predicted value for Y_i
X_i
X

Finding the list of fit types

Easiest way:

```
>> doc fit
```

- Scroll to the end, and select

"List of Library Models for Curve and Surface Fitting"

- Or visit the link below:

https://www.mathworks.com/help/curvefit/list-of-library-models-for-curve-and-surface-fitting.html

Task: Fitting data to a straight line

- Download file `lec34data.mat`
- Load the variables by double-clicking the file in the Current Folder panel
- Run the fitting algorithm

```
F0 = fit(X, Y, fittype)
```

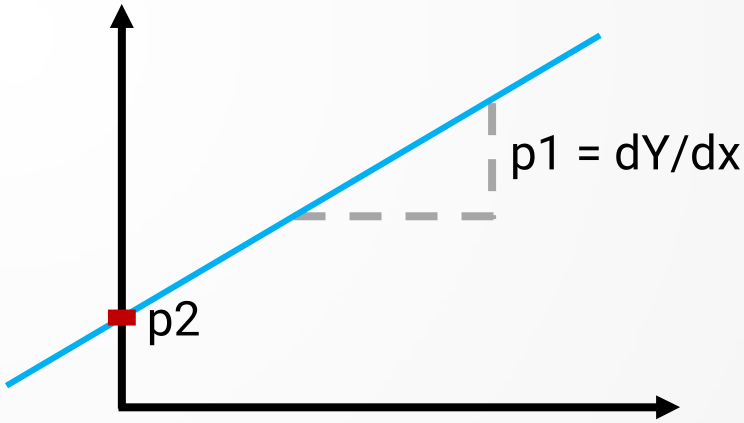
Find a suitable model:

https://www.mathworks.com/help/curvefit/list-of-library-models-for-curve-and-surface-fitting.html

Model for a straight line

- `poly1`
- Equation: `Y = p1*x + p2`

- `p1 = slope`
- `p2 = y-intercept`



Retrieving fitted variables

- To retrieve the value of fitted variables

  F0.(variableName)

- E.g. for poly1 model:
  - `slope = F0.p1`
  - `yInt = F0.p2`

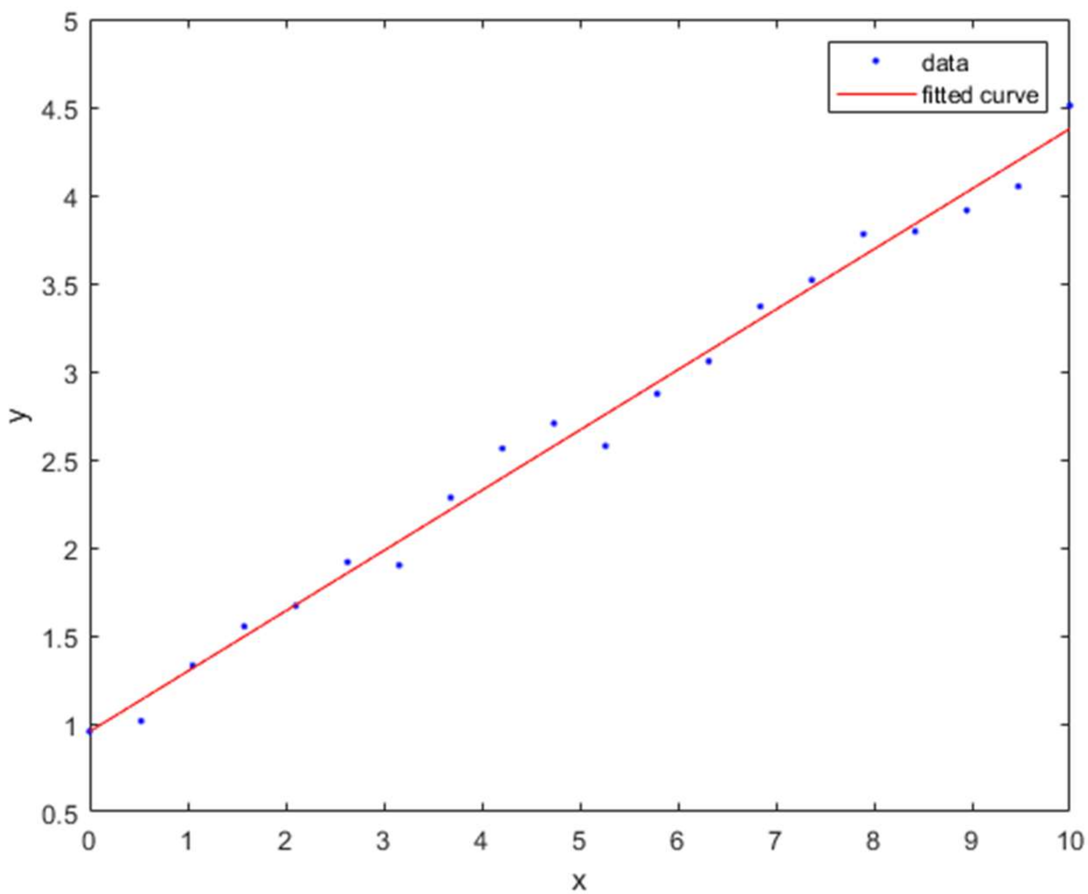
- The names of the fitted variables are different for each model, so make sure to read documentation or look at the fit object

Plotting fit results

- Syntax:

```
plot(F0, X, Y)
```

- Note:
  - Using plot with the fit object results in a slightly different behavior
  - Data is plotted as circles, fitted line is shown in red



data
fitted curve
y
x
5
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
1
2
3
4
5
6
7
8
9
10

Evaluating the fit

- [FO, G] = fit(X, Y, ...) returns appropriate goodness-of-fit measures, for the given inputs, in the structure G.

R2

- R2 is the coefficient of determination
- R^2 typically has values between 0 and 1
  - $R^2 = 0$ – no data lies on the line described by the model ("bad fit")
  - $R^2 = 1$ – all the data lies on the line described by the model ("perfect fit")
- Typically want values of 0.98 and above
- https://en.wikipedia.org/wiki/Coefficient_of_determination

Discussion on nearest-neighbor tracking

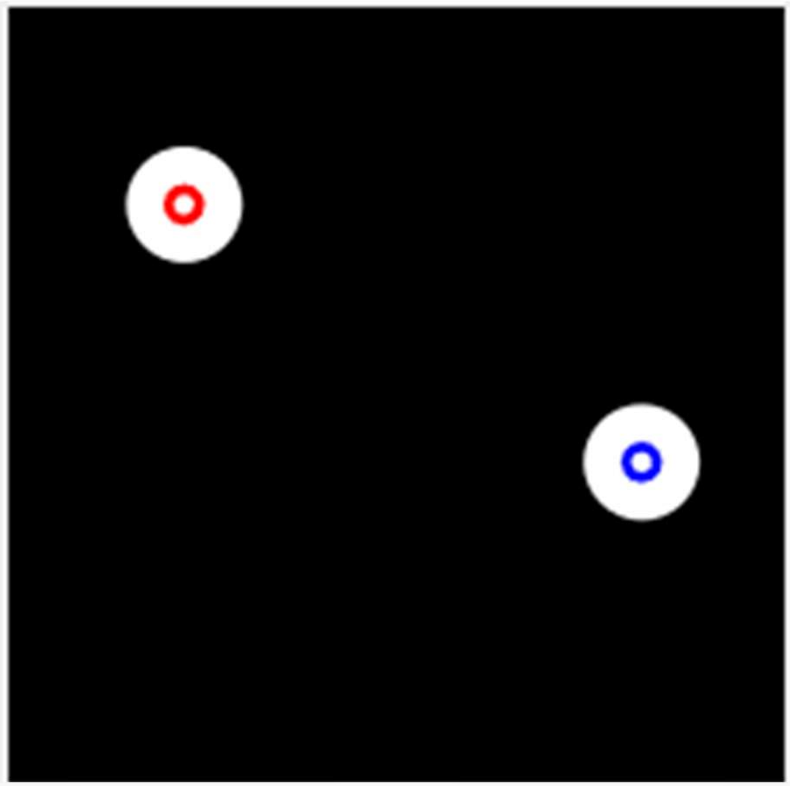
Flaws in our implementation of the algorithm

Linking the same object multiple times

- We don't keep track of linked objects so the same object could be linked multiple times

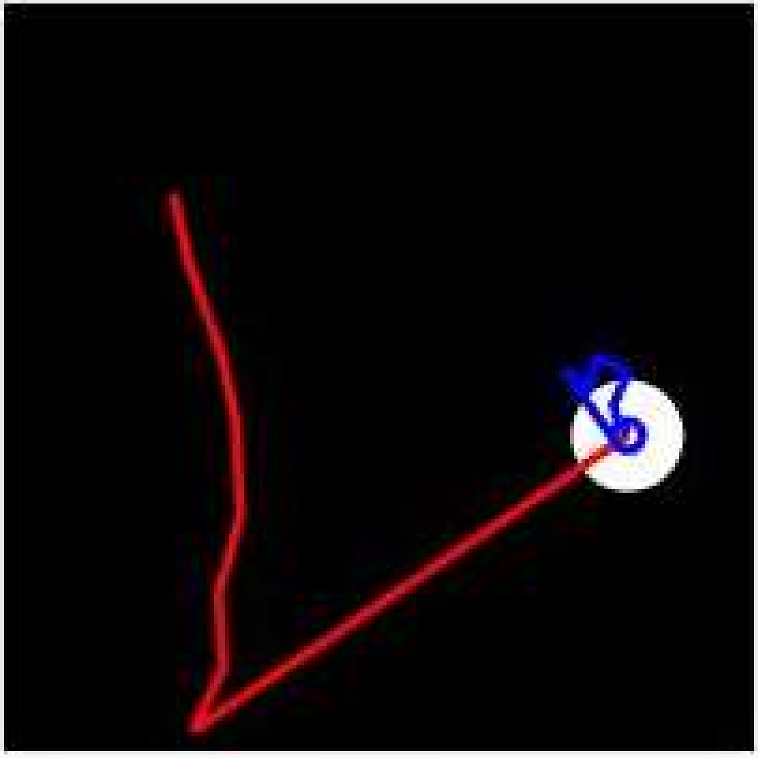
Linking the same object multiple times

- Example: A particle leaves the field of view



Linking the same object multiple times

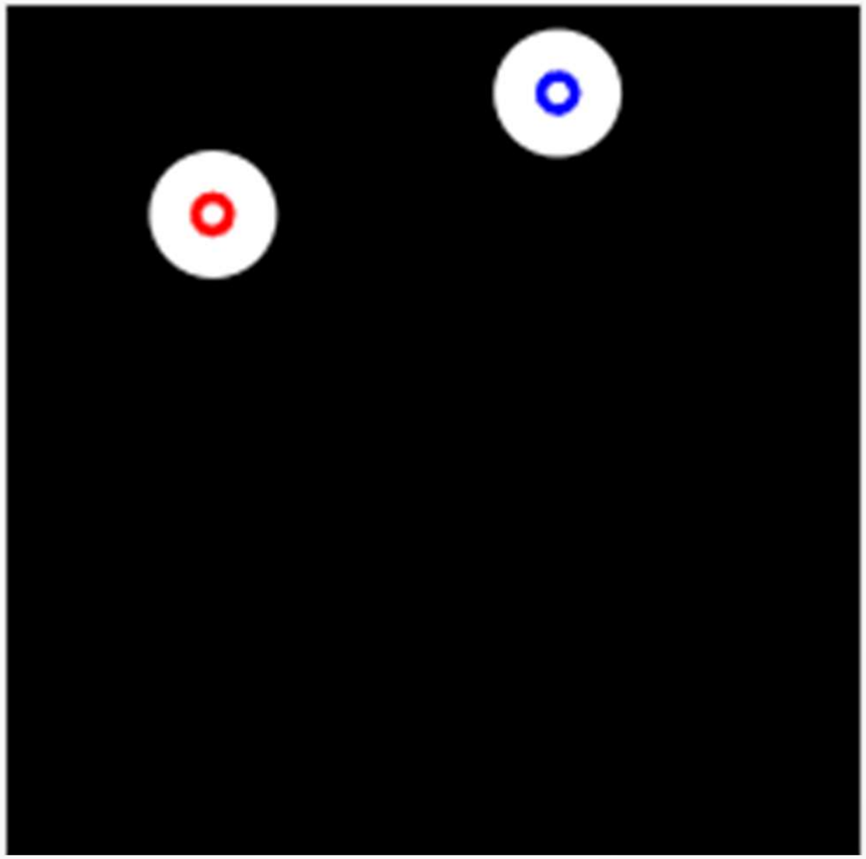
- Example: A particle leaves the field of view



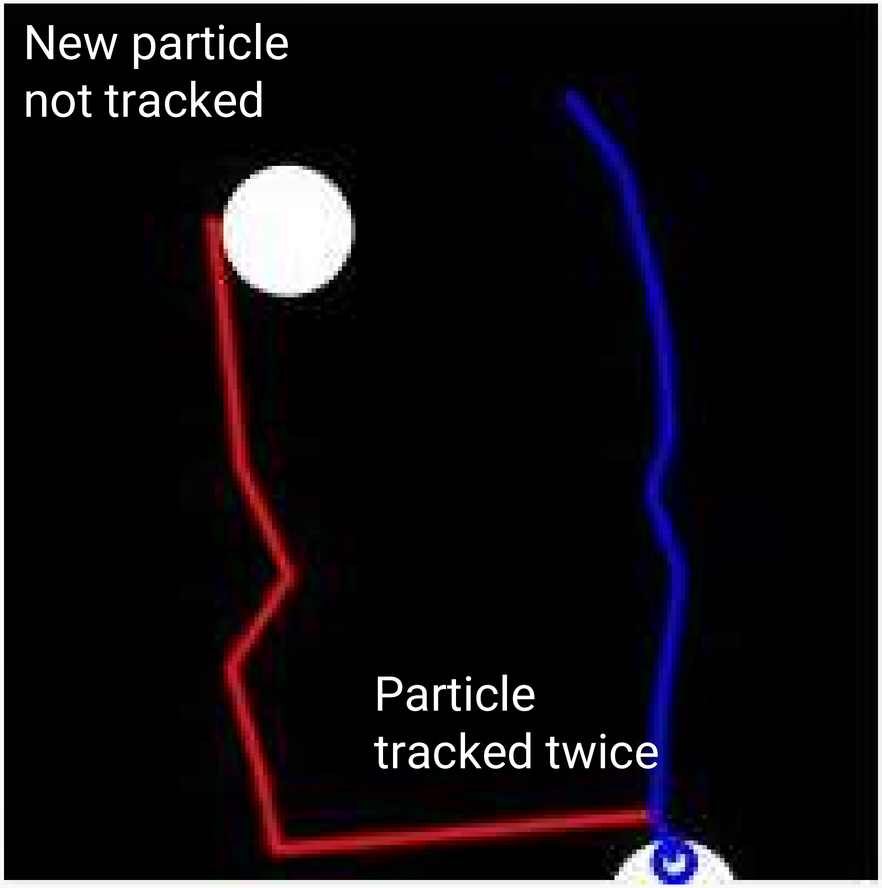
Linking the same object multiple times

- We don't keep track of linked objects so the same object could be linked multiple times
- Solution:
  - Keep track of linked particles
  - Do not let them be linked twice
  - If particle was not linked to a new object in current frame, stop tracking it

Incorrect linking when objects leave the field of view

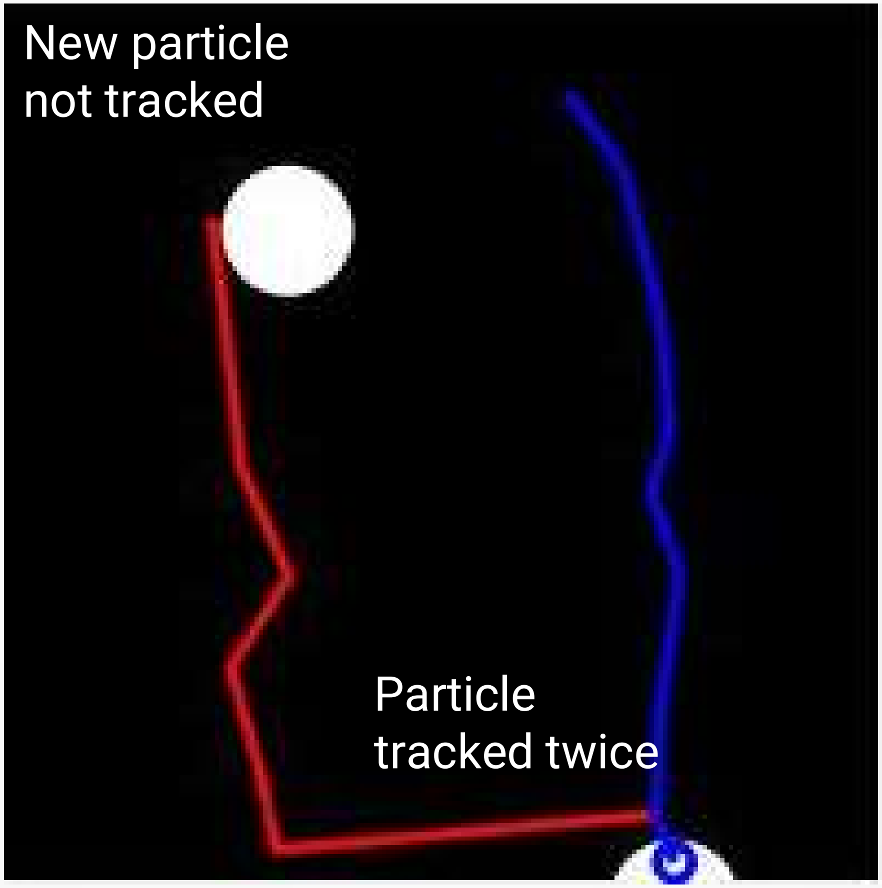


Incorrect linking when objects leave the field of view



Incorrect linking when objects leave the field of view

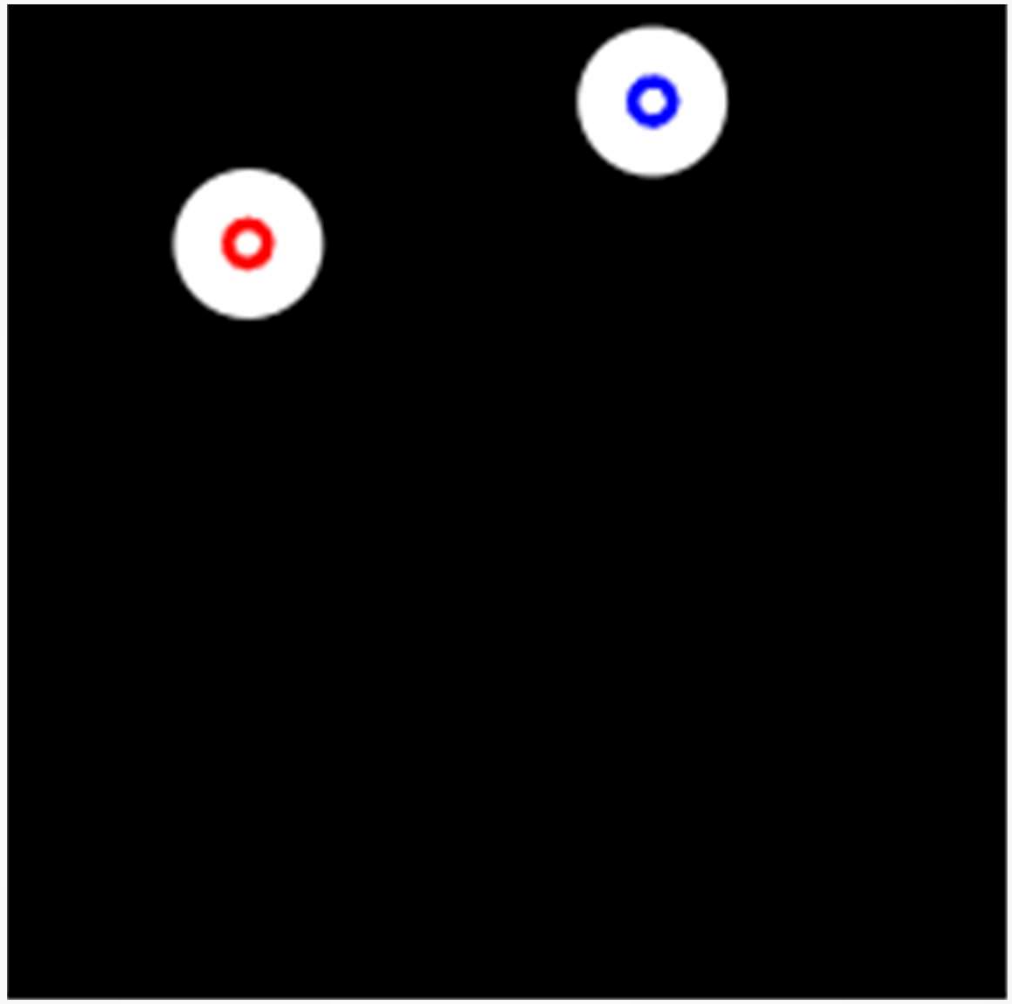
Identify when particles have no valid links – create a new track



Solution: restrict the maximum distance that objects can be linked to

Extensions to the code

- Create new tracks for unlinked detections (new objects)
- Only allow each detection to be linked ONCE
- Stop tracking tracks with no new detections in current frame (lost objects)
- Restricting distance to avoid linking objects over physically impossible distances
- I will upload an example tracking code that handles these issues on Canvas for you to look at (won't be on final exam)



General flaws with nearest-neighbor and tracking algorithms

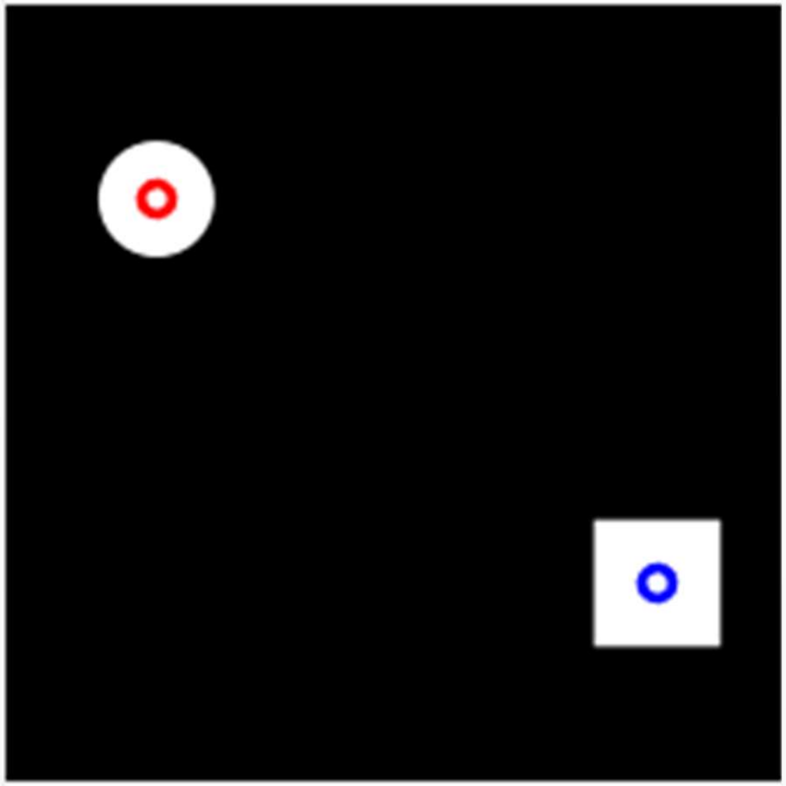
Tracking algorithms generally struggle

- When objects move **too fast** (so they cross paths)
- When the scene is **crowded**

- These limitations affect each other. The more crowded the scene, the more likely a fast moving particle is tracked incorrectly

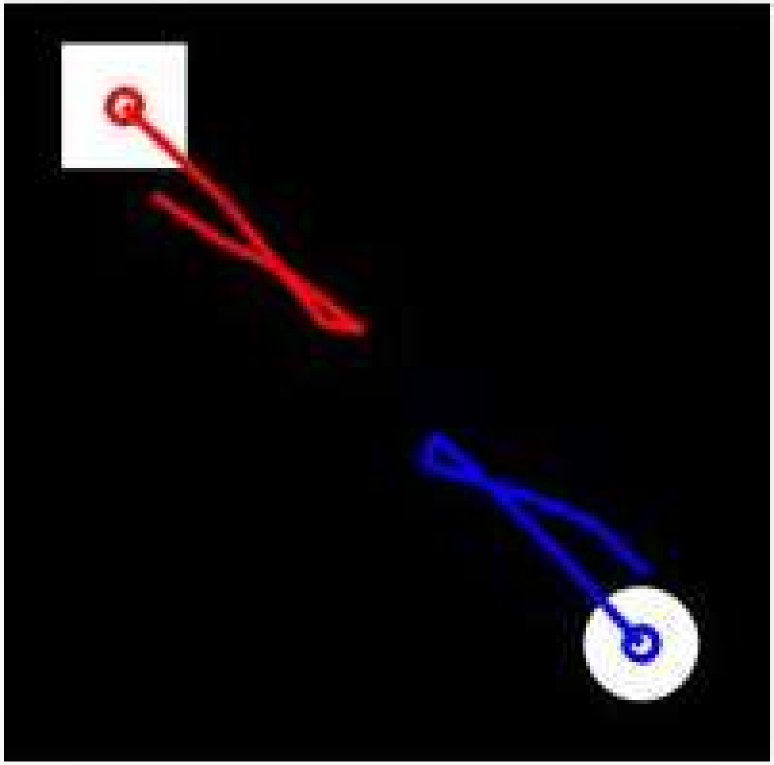
Example of a general problem with nearest-neighbor algorithms

- Sometimes objects "cross paths"



Example of a general problem with nearest-neighbor algorithms

- Sometimes objects "cross paths"



Further reading

- Jaqaman et al. Nat. methods 5, 695-702 (2008).
  - https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2747604/
- Ulman et al. Nat. methods 14, 1141-1152 (2017).
  - https://www.nature.com/articles/nmeth.4473

Not on final exam

Final word on tracking

There is no "perfect" tracking algorithm… at least right now

Announcement

- This is the last image analysis class with real content
- I encourage you to recreate the tracking code on your own – am happy to answer questions

- **Final image analysis class (Dec 2) will be a wrap-up/review**
  - Come with questions
  - Email me if you would like me to go over a concept