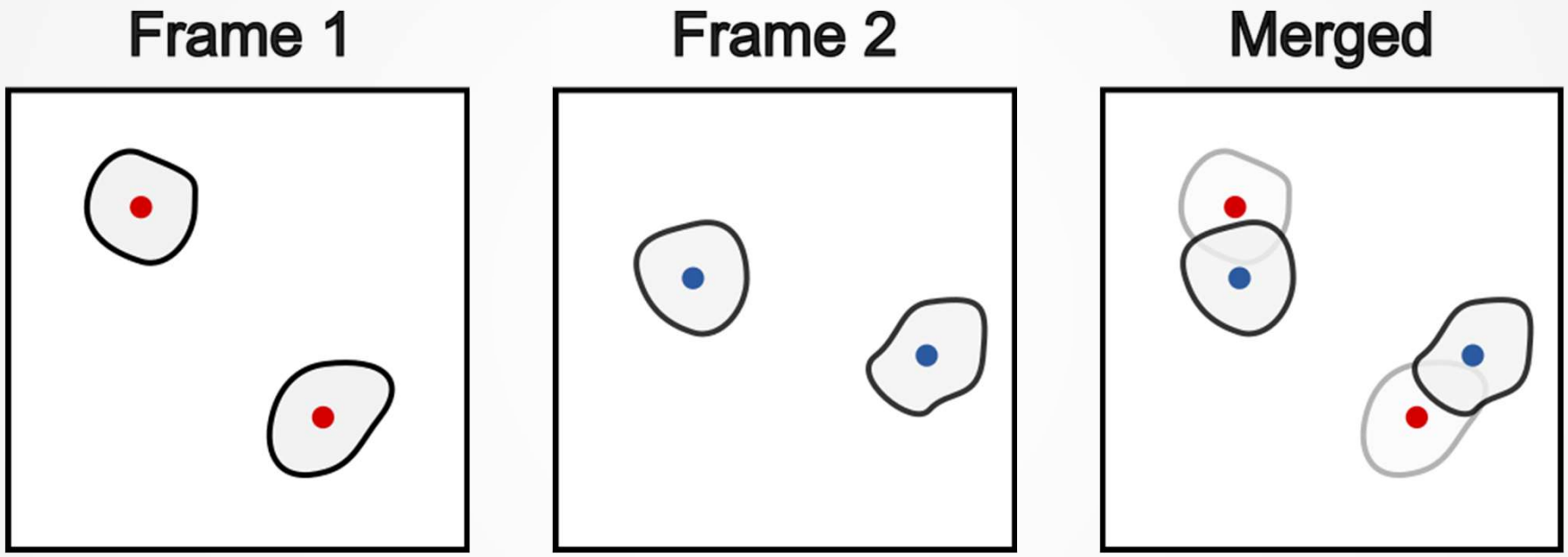
Lecture 32:
Nearest-neighbor Tracking (II)

University of Colorado Boulder

MCDB/BCHM 4312/5312
Fall 2020

The tracking problem



How do we link objects in frame 2 with objects in frame 1?

Nearest neighbor algorithm

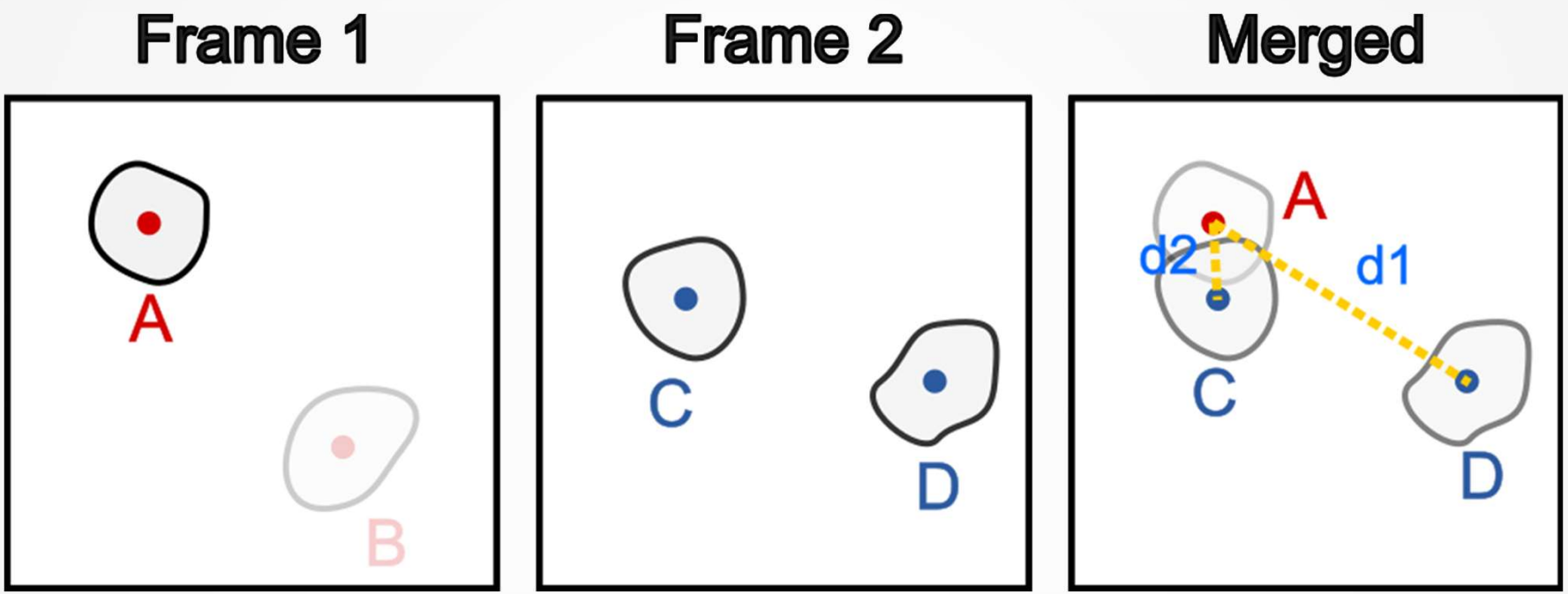
- Measure the distance from the last known (centroid) position of an object to the position of every detected object in the current frame

- Link objects with the shortest distance (i.e., the *nearest-neighbor*)

Recap

- Measured data from the two frames
- Computed the distance from objects in frame 1 to objects in frame 2
- Stored the distance in a vector

If you have lost your script, download the example from last week on Canvas



Since d2 < d1, link A and C

Outline

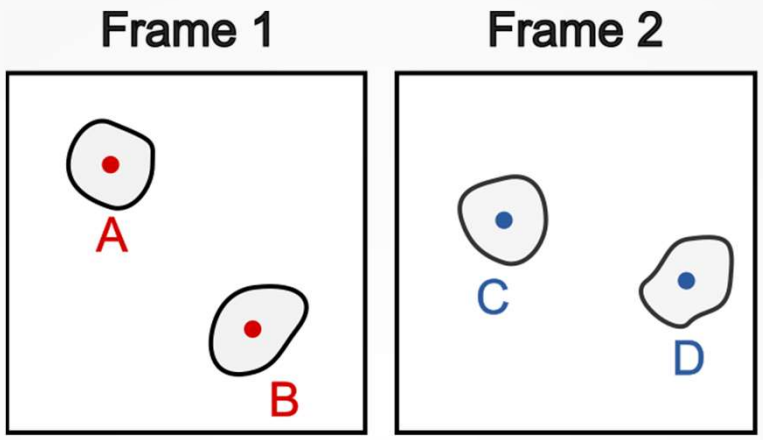
- Measured data from the two frames
- Computed the distance from objects in frame 1 to objects in frame 2
- Stored the distance in a vector
- Find the element of the nearest neighbor
- Create tracks (struct array)
- Completing the for loops

Task 5

- Find the nearest neighbor

```
[~, index] = min(distance);
```

Returns the index of the smallest value in the vector `distance`



Desired output:

$$\texttt{Track1.MeanIntensity} = [I_A, I_C]$$

$$\texttt{Track2.MeanIntensity} = [I_B, I_D]$$

Note: `MeanIntensity` is an example property

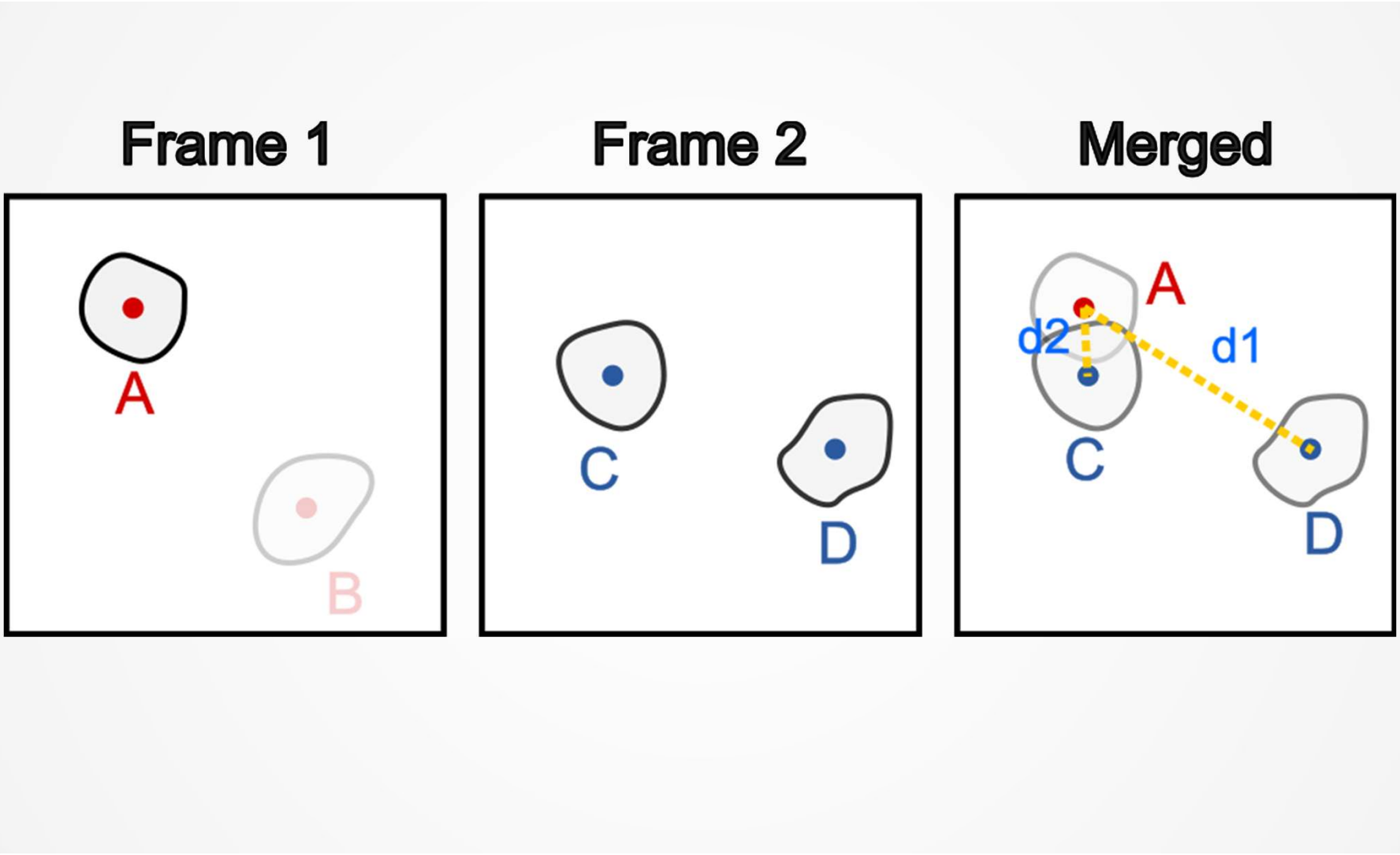
Task 6

- Initialize a struct to hold track data called `tracks`
- Populate the struct with data from Frame 1

Why? In Frame 1, we have no tracks, so we can just start new ones

Task 7

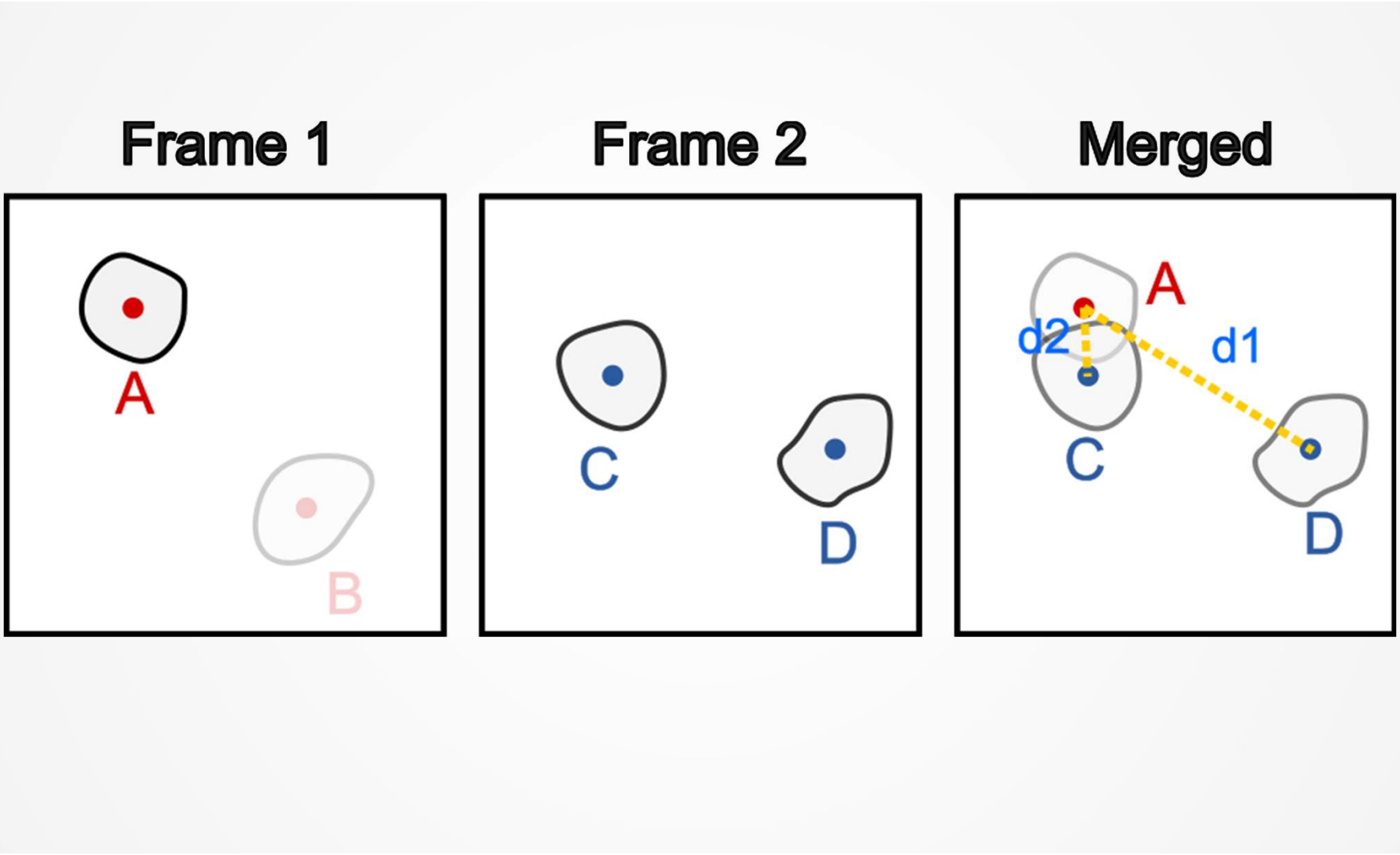
- Update the track with nearest-neighbor data from Frame 2

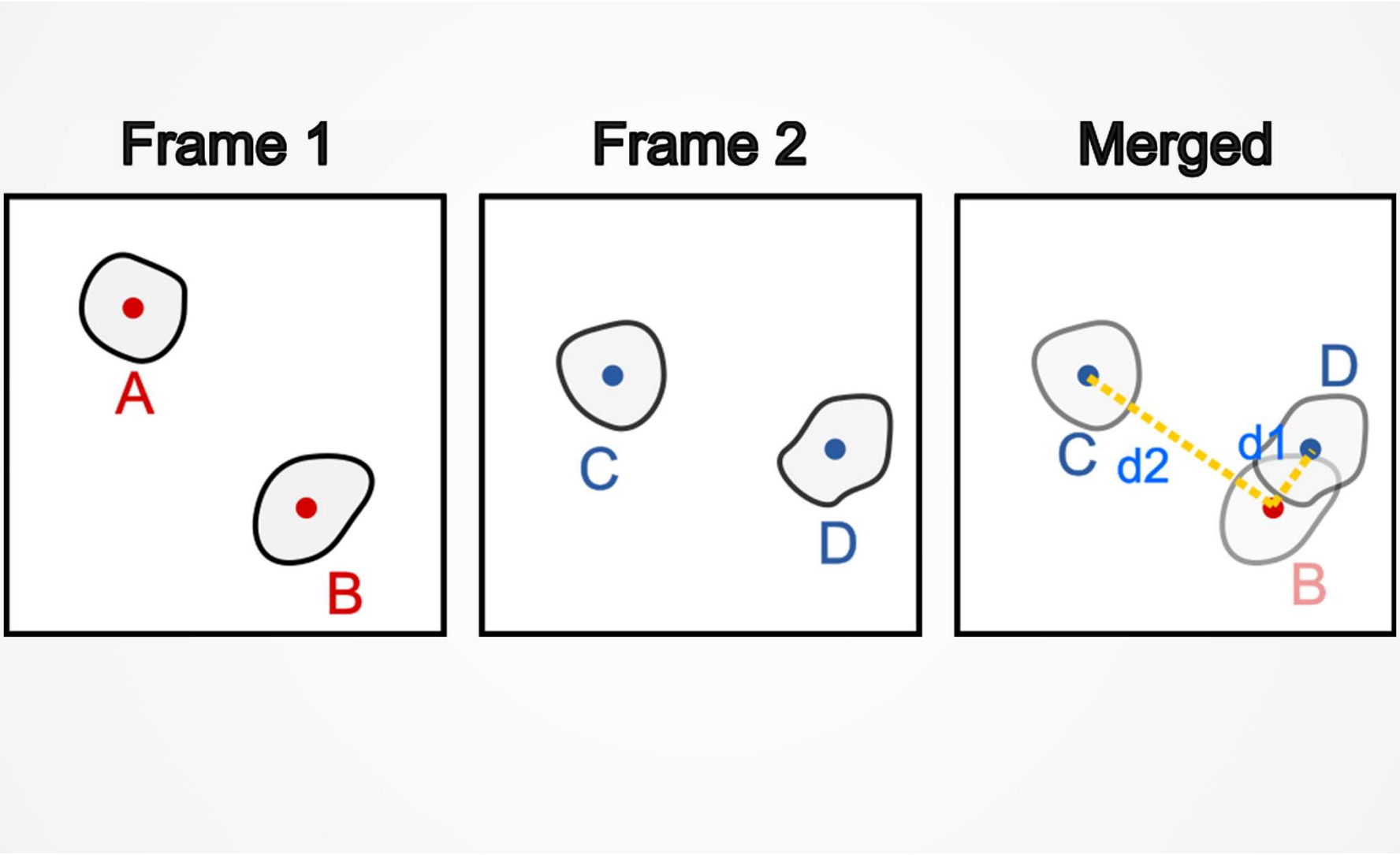

Frame 1
A
B
Frame 2
C
D
Merged
A
d2
d1
C
D

Expanding linking to multiple objects

Task 8

- Modify the for loop to use the last known object position


Frame 1
A
B
Frame 2
C
D
Merged
A
d2
d1
C
D


Frame 1
A
B
Frame 2
C
D
Merged
C
d2
d1
D
B

Task 9

- Expand the for loop to perform the linking step with every object in the `tracks` structure

Expanding to loop over each frame

Task 10

- Add (yet another) for loop to run the segmentation and tracking algorithm over every frame of the movie

Validating tracking results

Task 11

- Plot the position of the tracks over time to validate

Discussion on code performance

Extensions to the code

- Handling new objects (create new elements for unlinked detections)
- Handling loss of objects (i.e. objects drifting out of the field of view)
- Restricting distance to avoid linking objects over physically impossible distances

Limitations

- Objects cannot move "too fast" between frames
- Implementation is naïve, objects could cross paths and the tracks would swap between objects
  - Use a global minimization method (e.g. Linear Assignment)

Assumption

Images are acquired at a high enough frame rate that objects do not move "too much" between frames

How far can each object travel between frames?

- Depends on how far objects can travel in an image before crossing paths with another object
- Average distance between objects in image
- The more crowded an image, the stricter the requirement is (smaller step size is better)

Limitations

- Objects cannot move "too fast" between frames
- Implementation is naïve, objects could cross paths and the tracks would swap between objects
  - Use a global minimization method (e.g. Linear Assignment) – minimizes total distance of assignments in each frame

Practice

- Try to write a script from scratch to implement the nearest-neighbor algorithm
- Practice images:
  - `twonucl_2.tif` (easy)
  - `flowingbeads.tif` (harder – have to track particles entering and leaving field of view) – optional, not on final exam