Project 1 Required Components

- Read in labeled motion capture data
  - 3D points
  - Part labels for each point
- Tracking points through time to form trajectories, thus propagating part labels
  - Data association of trajectories up to time t-1 with points detected at time t
  - Do Kalman filter to update each trajectory with the point associated with it at time t
- Write out the corrected points / labels
- Quantitative evaluation

Project 1 Required Components

- Read in labeled motion capture data
  - 3D points
  - Part labels for each point
- Tracking points through time to form trajectories, thus propagating part labels
  - Data association of trajectories up to time t-1 with points detected at time t
  - Do Kalman filter to update each trajectory with the point associated with it at time t
- Write out the corrected points / labels
- Quantitative evaluation

Recall: Filtering Framework

Recall our discussion of state space filtering

We want to recursively estimate the current state at every time that a measurement is received.

Two step approach:

1) prediction: propagate state pdf forward in time, taking process/model noise into account (translate, deform, and spread the pdf)

2) update: use Bayes theorem to modify prediction pdf based on current noisy measurement

Recall: Kalman Filter

Kalman filtering is an example of Bayes filtering where:

- Motion model and measurement model are linear
- Noise is zero-mean Gaussian
- Prior distribution on state vector is Gaussian

1) Motion model

$$\mathbf{x}_k = F_k \mathbf{x}_{k-1} + \mathbf{v}_{k-1} \qquad p(v_k) = N(v_k \mid 0, Q_k)$$

2) Measurement model

$$\mathbf{z}_k = H_k \mathbf{x}_k + \mathbf{n}_k \qquad p(n_k) = N(n_k \mid 0, R_k)$$

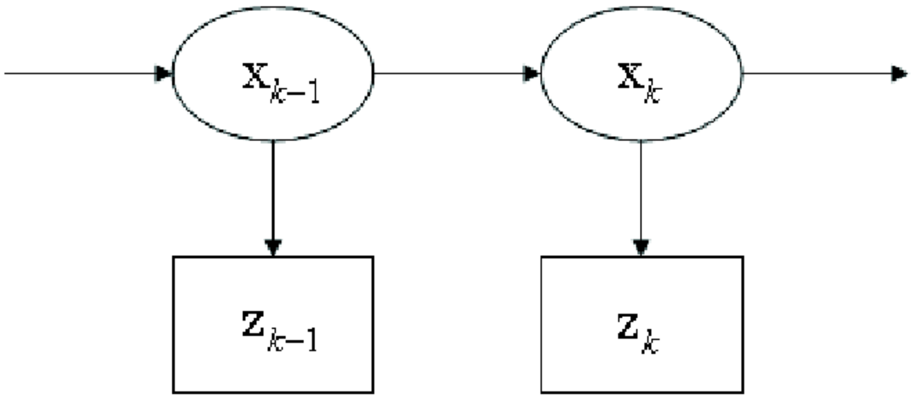
Kalman Filter

Under those conditions, all probability distributions remain Gaussian.

$$p(\mathbf{x}_k|\mathbf{x}_{k-1}) = N(\mathbf{x}_k|\,\mathbf{F}_k\mathbf{x}_{k-1}, \mathbf{Q}_k)$$

$$p(\mathbf{z}_k|\mathbf{x}_k) = N(\mathbf{z}_k|\,\mathbf{H}_k\mathbf{x}_k, \mathbf{R}_k)$$

$$p(\mathbf{x}_{k-1}|\mathbf{Z}_{k-1}) = N(\mathbf{x}_{k-1}|\,\hat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1})$$



Kalman Filter

Predict

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \quad \text{(predicted state)}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \quad \text{(predicted estimate covariance)}$$

Update

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \quad \text{(innovation or measurement residual)}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \quad \text{(innovation (or residual) covariance)}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1} \quad \text{(Kalman gain)}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \quad \text{(updated state estimate)}$$

$$\mathbf{P}_{k|k} = (I - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \quad \text{(updated estimate covariance)}$$

Problem

When tracking multiple objects, or when there are multiple detections, how do we know which detection to use when updating each state vector?

<u>Two step approach:</u>

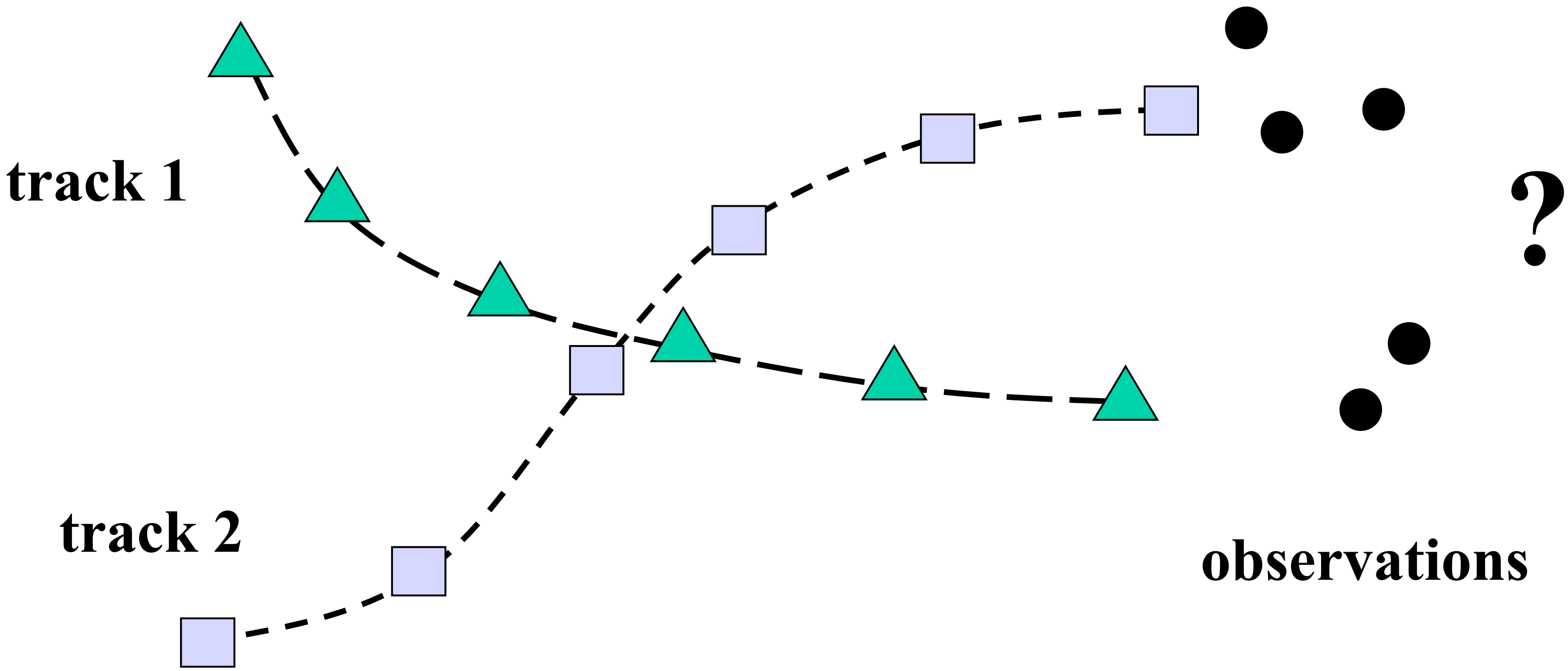
1) prediction: propagate state pdf forward in time, taking process noise into account (translate, deform, and spread the pdf)

2) update: use Bayes theorem to modify prediction pdf based on current measurement

But which observation should we update with?

Data Association

Multi-frame matching (matching observations in a new frame to a set of tracked trajectories)

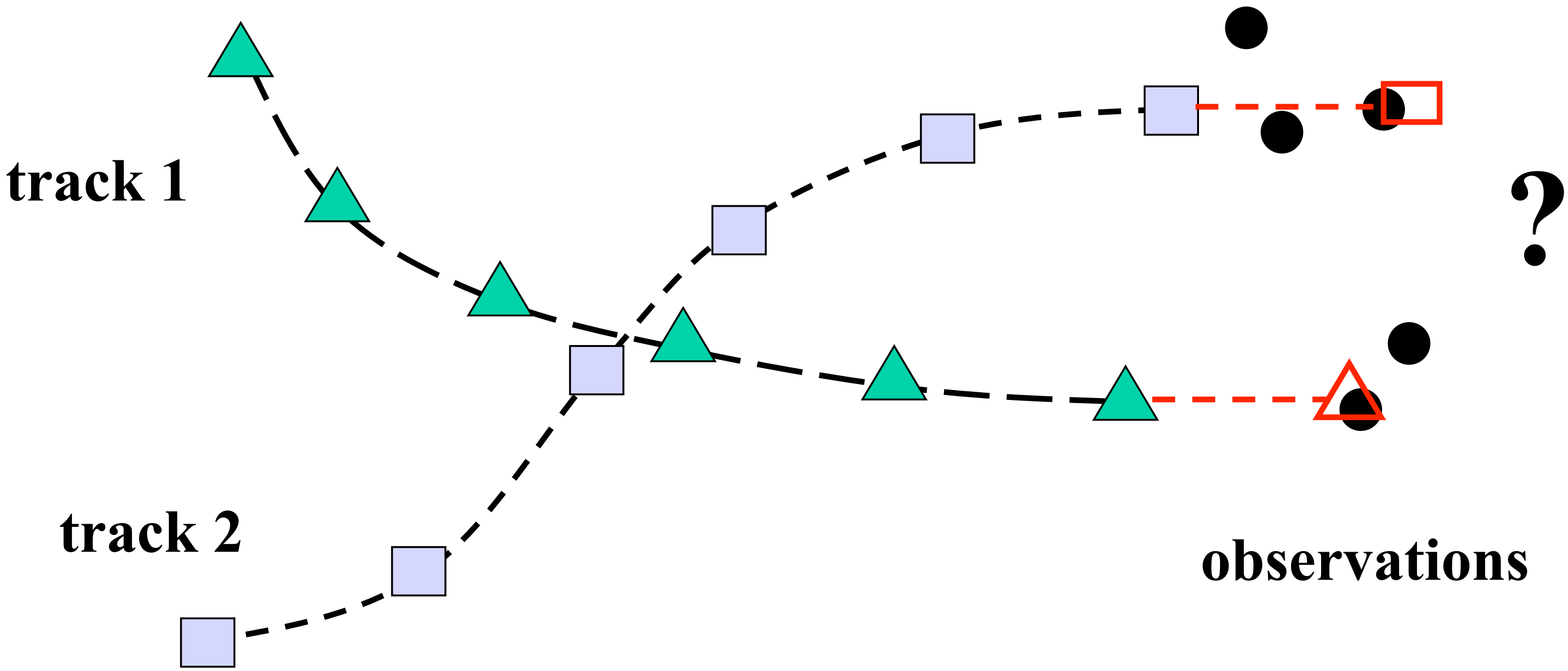


How to determine which observations to add to which track?

Robert Collins
CSE586

Data Association

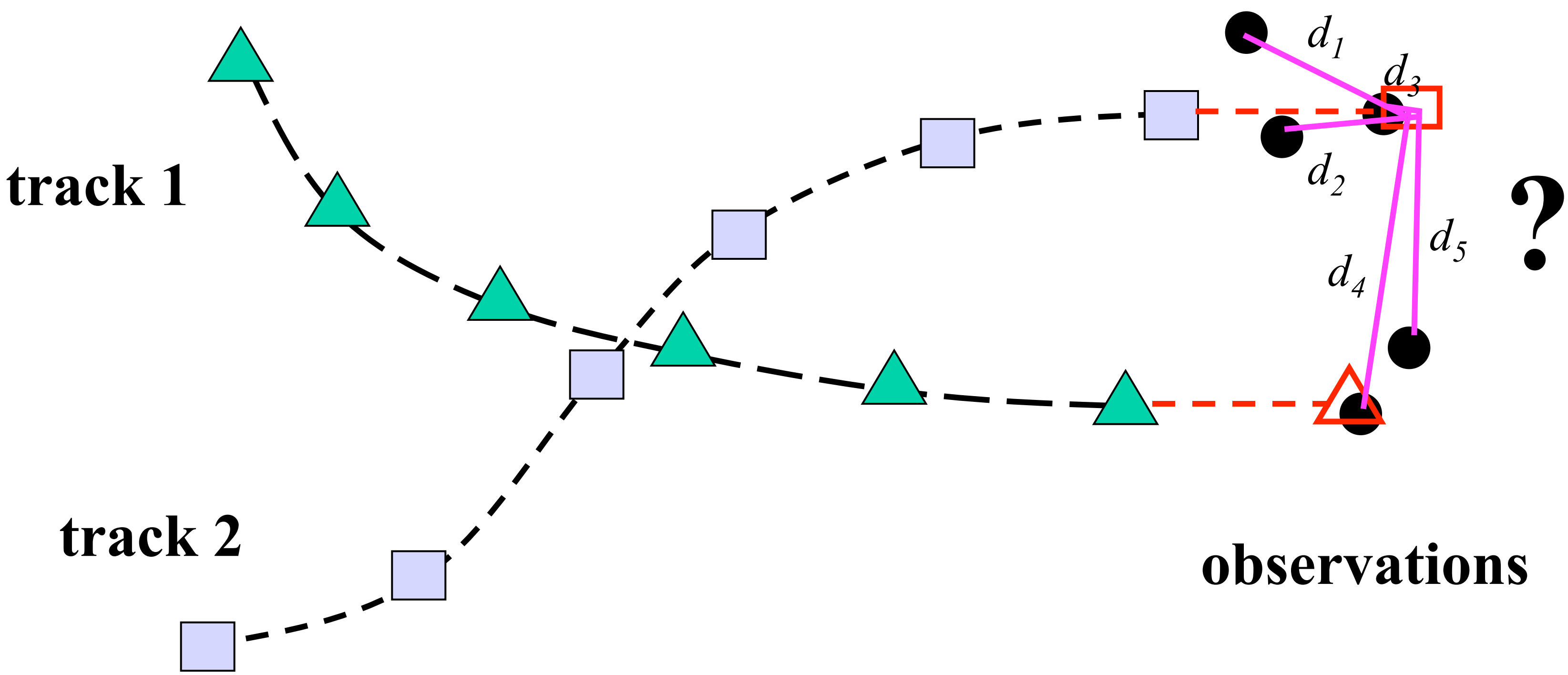
Intuition: predict next position along each track.



How to determine which observations to add to which track?

Data Association

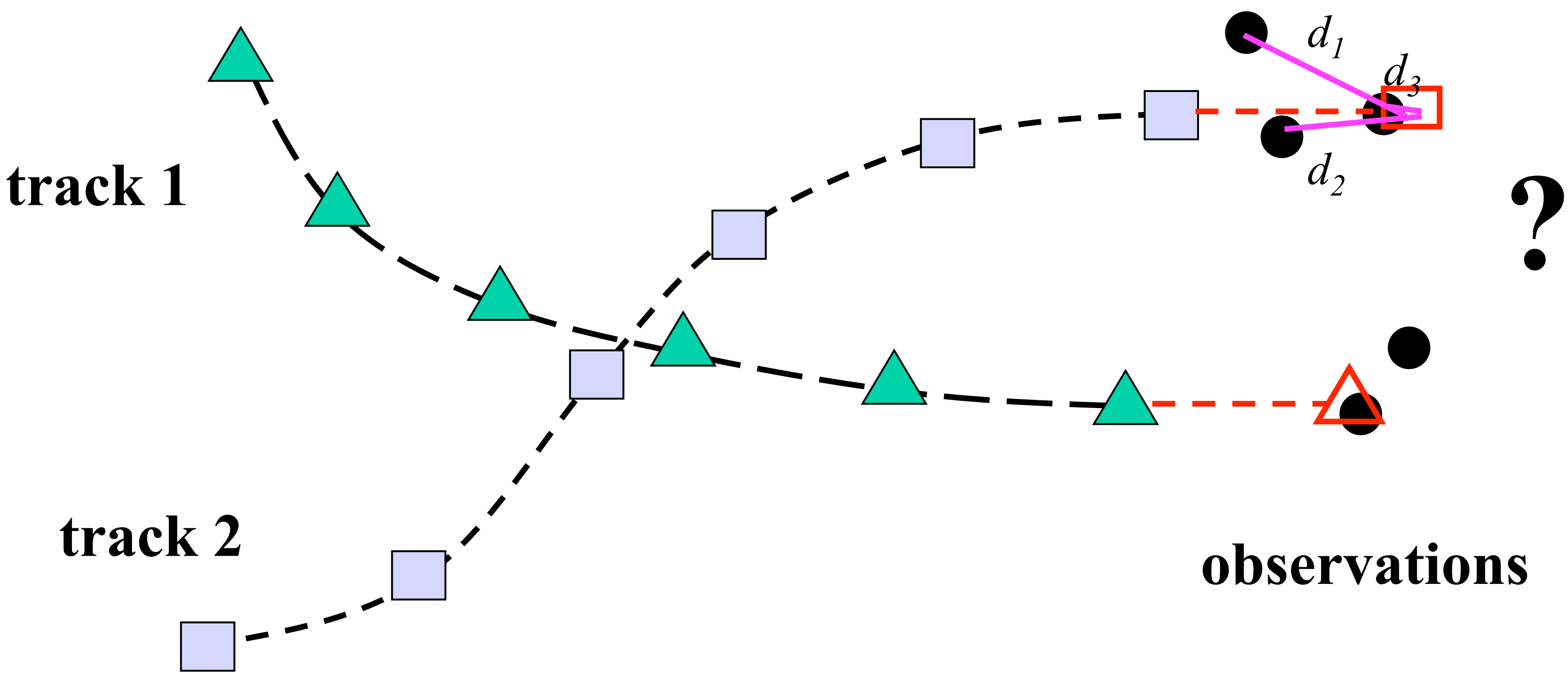
Intuition: predict next position along each track.
Intuition: match should be close to predicted position.



How to determine which observations to add to which track?

Data Association

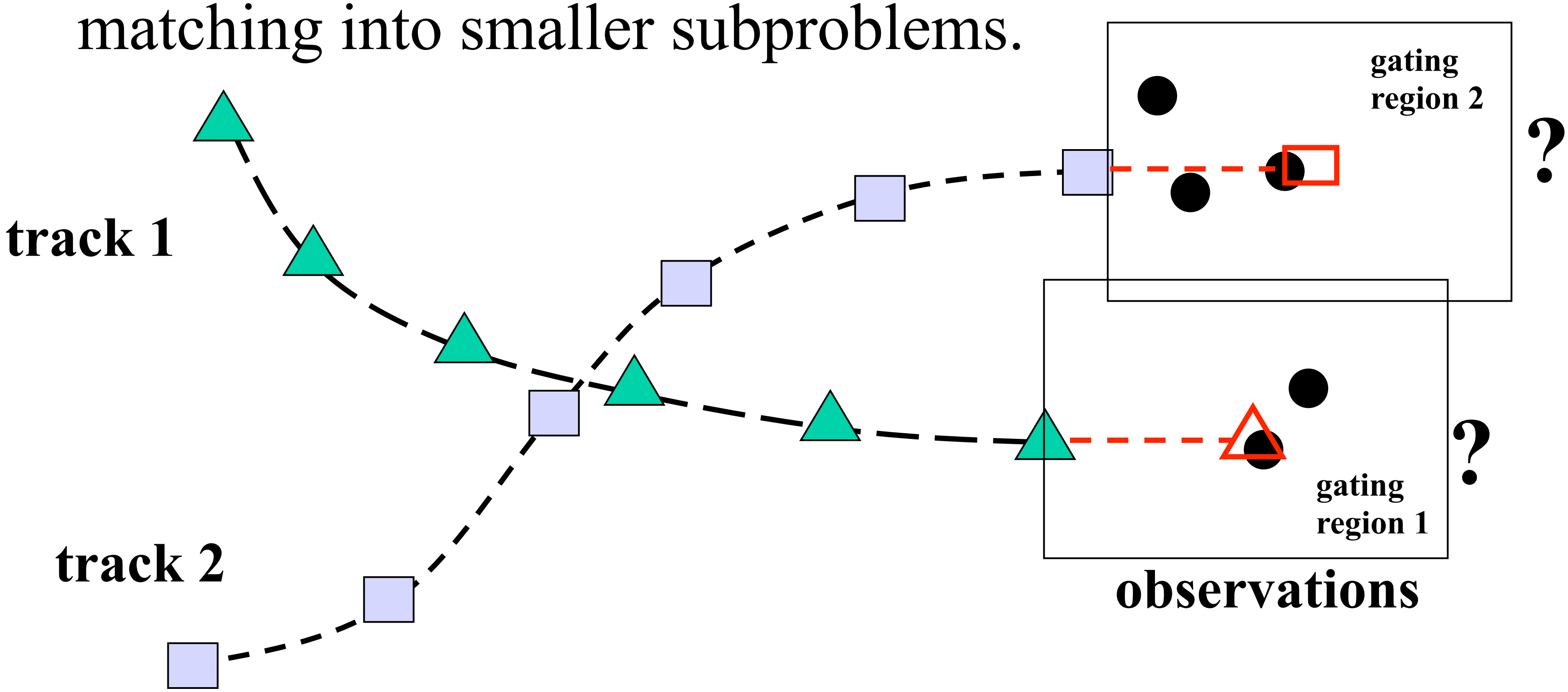
Intuition: predict next position along each track.
Intuition: match should be close to predicted position.
Intuition: some matches are highly unlikely.



How to determine which observations to add to which track?

Gating

A method for pruning matches that are geometrically unlikely from the start. Allows us to decompose matching into smaller subproblems.



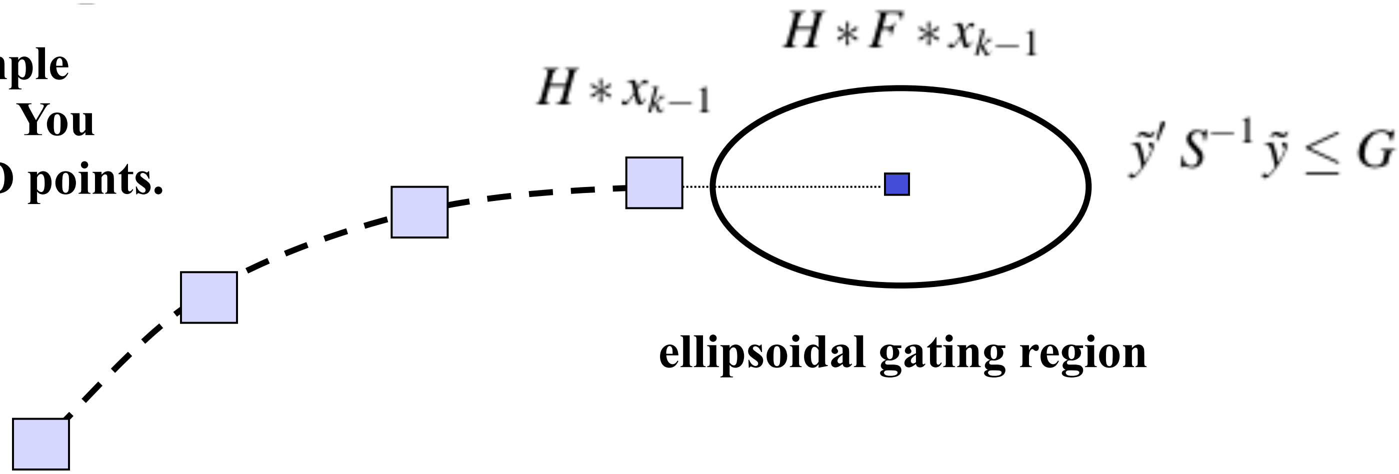
How to determine which observations to add to which track?

Robert Collins
CSE586

Gating for Kalman Filters

$$x = \begin{bmatrix} x \\ y \\ u \\ v \end{bmatrix} \quad F = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

Note: this example uses 2D points. You will be using 3D points.



$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} \quad \text{(predicted state)}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \quad \text{(predicted estimate covariance)}$$

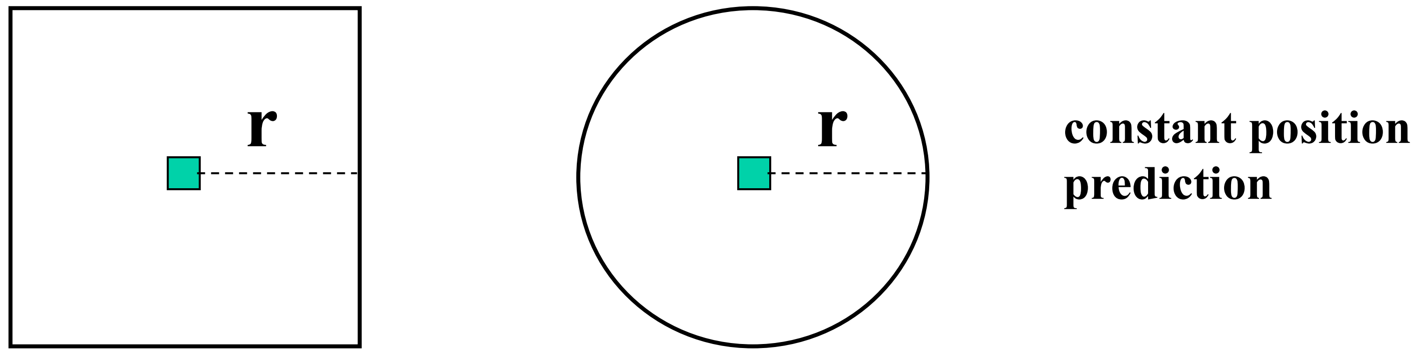
$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_{k|k-1} \quad \text{(innovation or measurement residual)}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \quad \text{(innovation (or residual) covariance)}$$

Robert Collins
CSE586

Simpler Prediction/Gating

Could replace ellipsoidal error region with box or sphere:



Note: the half-width or radius r should be computed somehow from the variances in D when KF covariance is decomposed as Cov => U D U^T

Alternatively, perhaps you could simplify the KF equations by assuming all covariance matrices are diagonal, with equal variances along the diagonal.

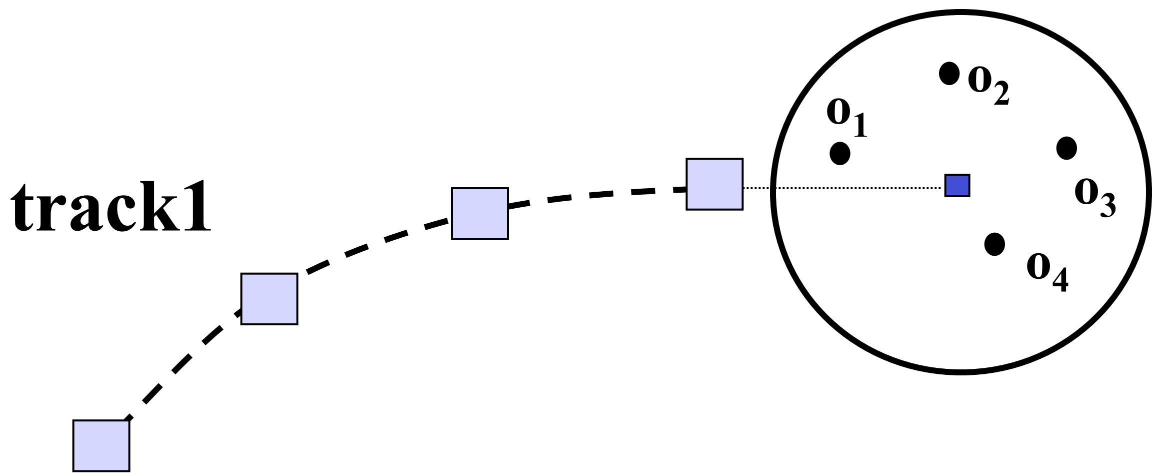
Filtering, Gating, Association

<u>Add Gating and Data Association</u>

1) prediction: propagate state pdf forward in time
2) Gating to determine possible matching observations
3) Data association to determine best match
4) update: use Bayes theorem to modify prediction pdf based on current noisy measurement

Global Nearest Neighbor (GNN)

Evaluate each observation in track gating region.
Choose "best" one to incorporate into track.

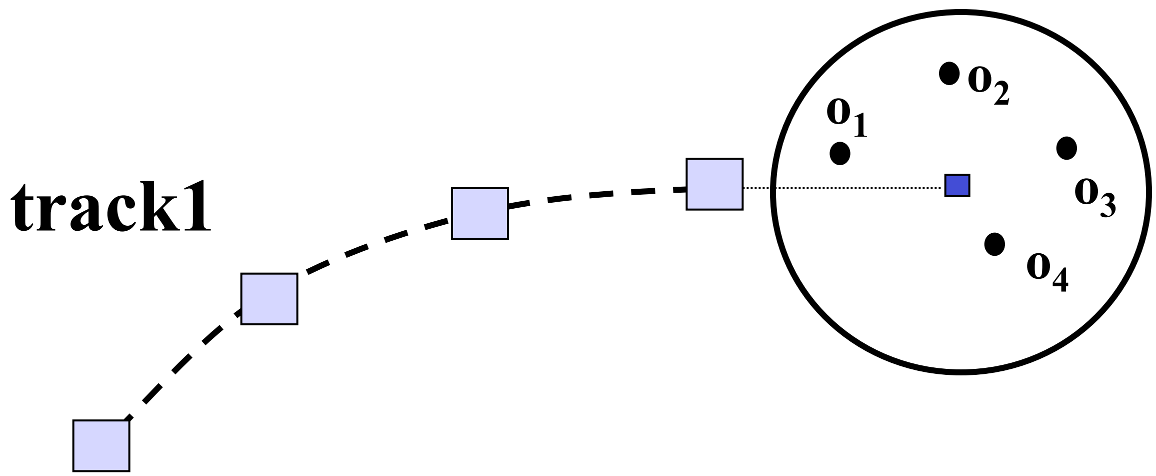


a_{1j} = score for matching observation j to track 1

Could be based on Euclidean or Mahalanobis distance to predicted location (e.g. $\exp\{-d^2\}$). Could be based on similarity of appearance (e.g. appearance template correlation score)

Global Nearest Neighbor (GNN)

Evaluate each observation in track gating region.
Choose "best" one to incorporate into track.



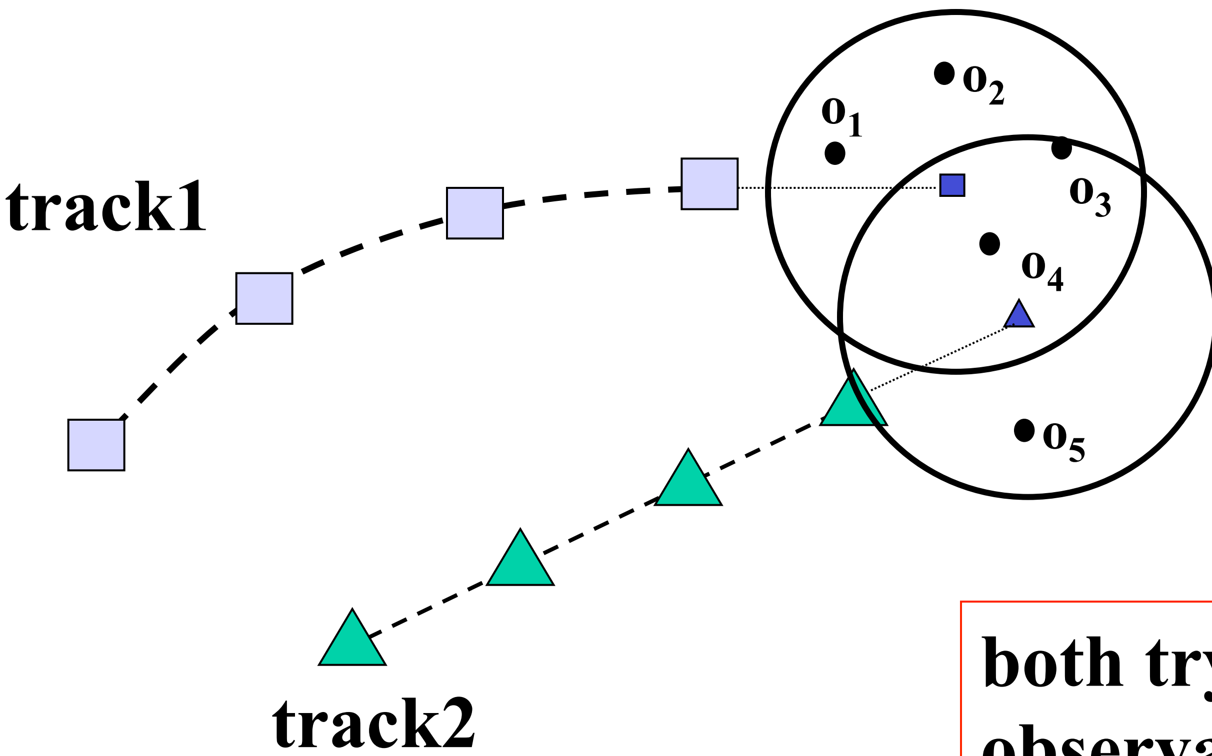
	ai1	
1	3.0	
2	5.0	
3	6.0	
4	$\boxed{9.0}$	max

a_{i1} = score for matching observation i to track 1

Choose best match $a_{m1} = \max\{a_{11}, a_{21}, a_{31}, a_{41}\}$

Global Nearest Neighbor (GNN)

Problem: if do independently for each track, could end up with contention for the same observations.



	ai1	ai2
1	3.0	
2	5.0	
3	6.0	1.0
4	$\boxed{9.0}$	$\boxed{8.0}$
5		3.0

both try to claim observation o_4

A Greedy (Best First) Strategy

Find the largest score.

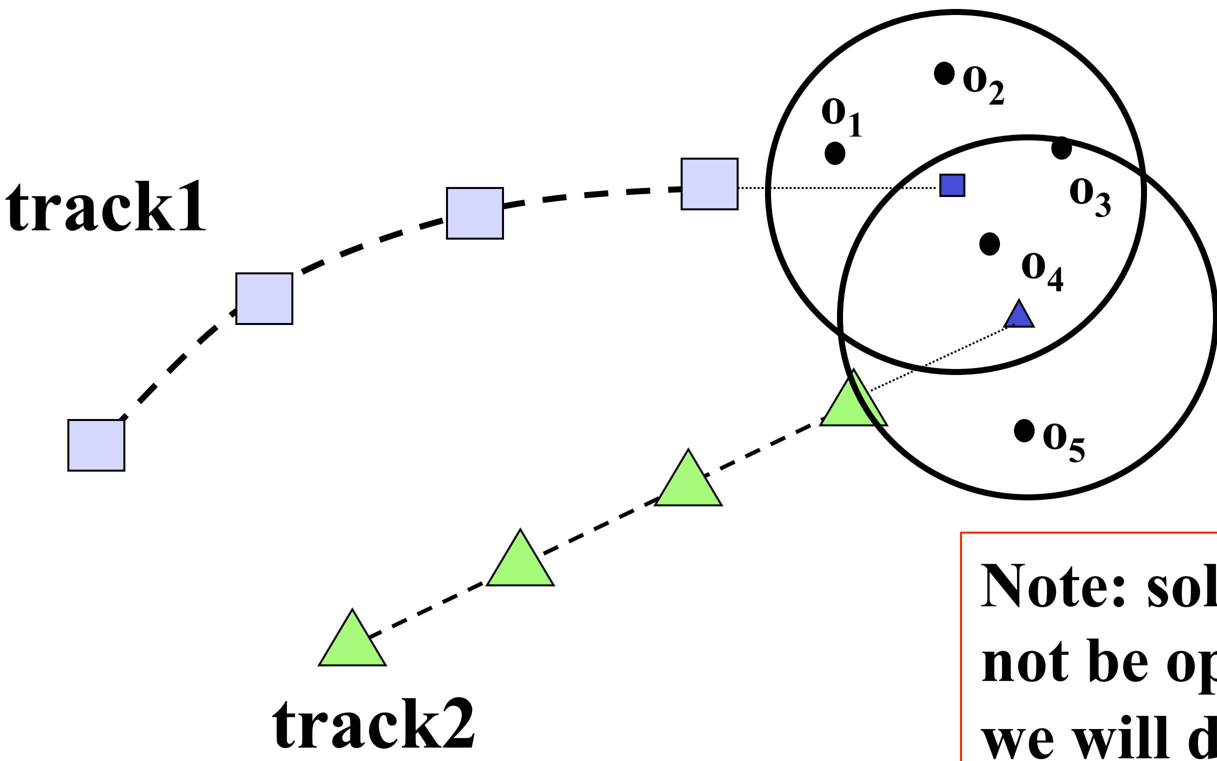
Remove scores in same row and column from consideration (that is, enforcing the 1-1 matching constraints)

Repeat

	1	**2**	**3**	**4**	**5**
1	$\boxed{0.95}$	~~0.76~~	~~0.62~~	~~0.41~~	~~0.06~~
2	~~0.23~~	~~0.46~~	~~0.79~~	$\boxed{0.94}$	~~0.35~~
3	~~0.61~~	~~0.02~~	$\boxed{0.92}$	~~0.92~~	~~0.81~~
4	~~0.49~~	$\boxed{0.82}$	~~0.74~~	~~0.41~~	~~0.01~~
5	~~0.89~~	~~0.44~~	~~0.18~~	~~0.89~~	$\boxed{0.14}$

Greedy (Best First) Strategy

Assign observations to trajectories in decreasing **order of goodness, making sure to not reuse an observation twice.**

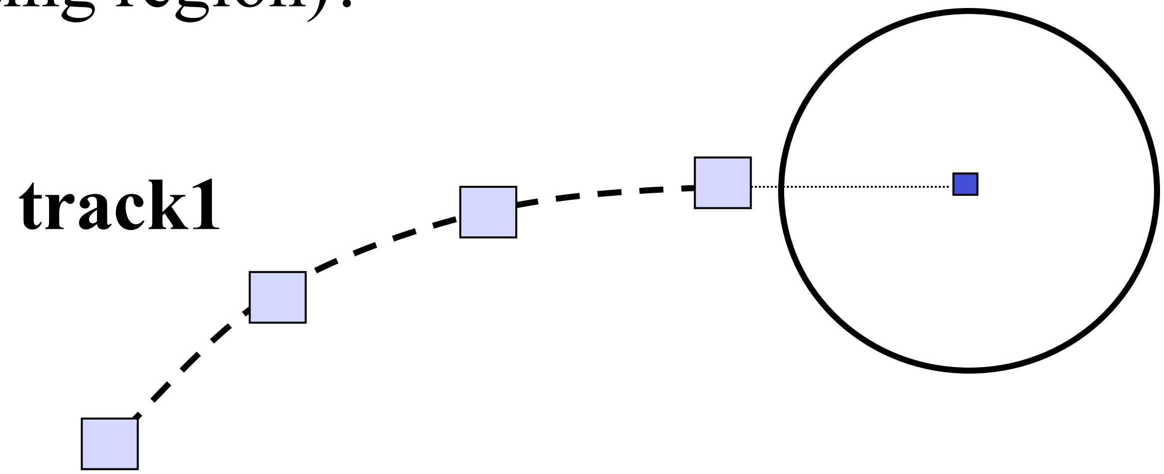


	ai1	ai2
1	3.0	
2	5.0	
3	6.0	1.0
4	$\boxed{9.0}$	8.0
5		$\boxed{3.0}$

Note: solution may not be optimal! Later we will discuss other methods that find the optimal solution.

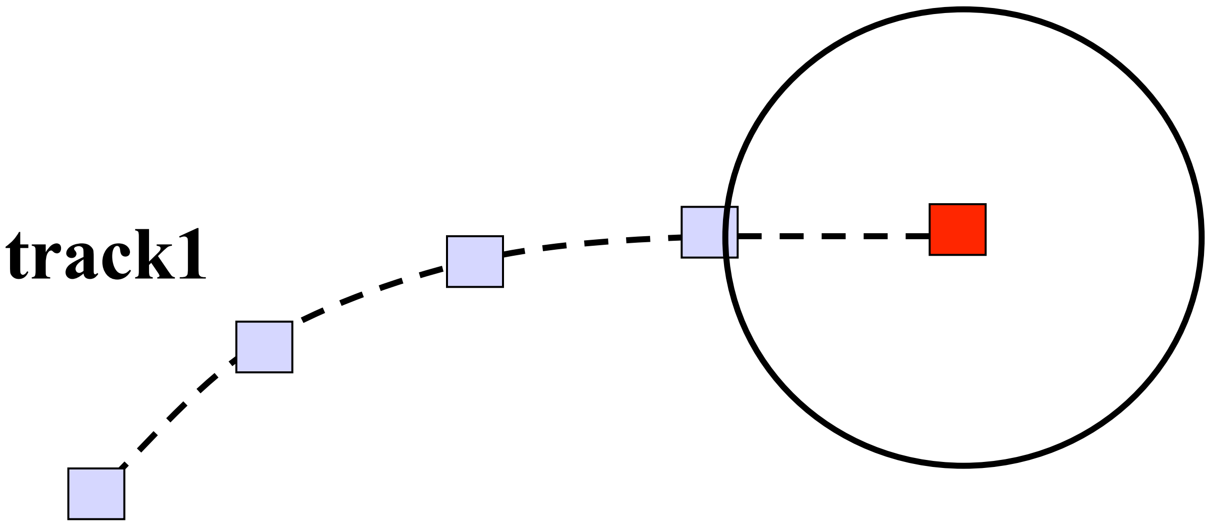
Missing Observations

What to do if there is no good observation to use for updating a trajectory (that is, no observation in the gating region)?



Missing Observations

Use predicted location as the updated location, but flag it as missing and increase the variance.

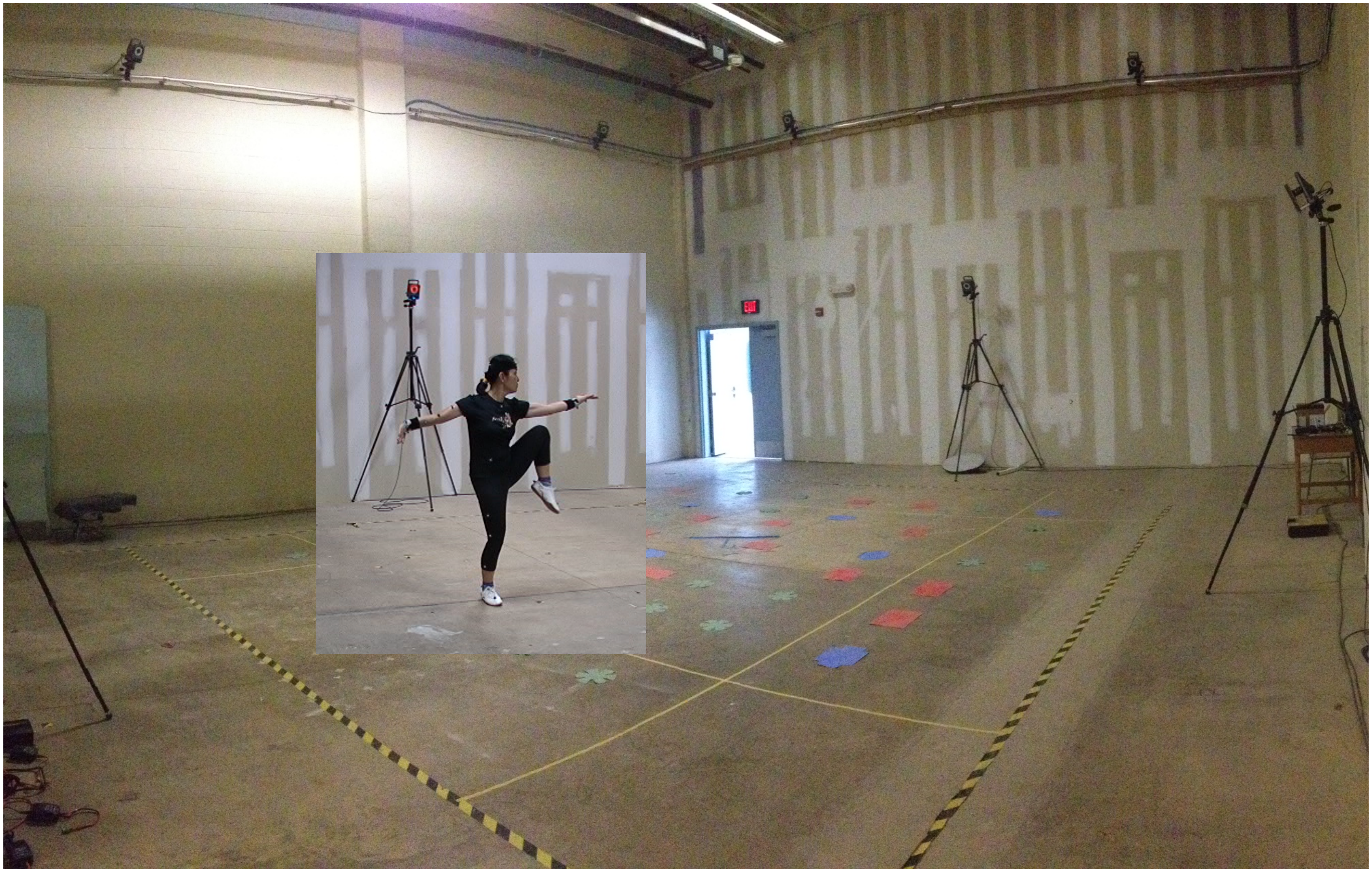


This way you can keep predicting forward in time, and hopefully regain tracking using detections in future frames.

Project 1 Required Components

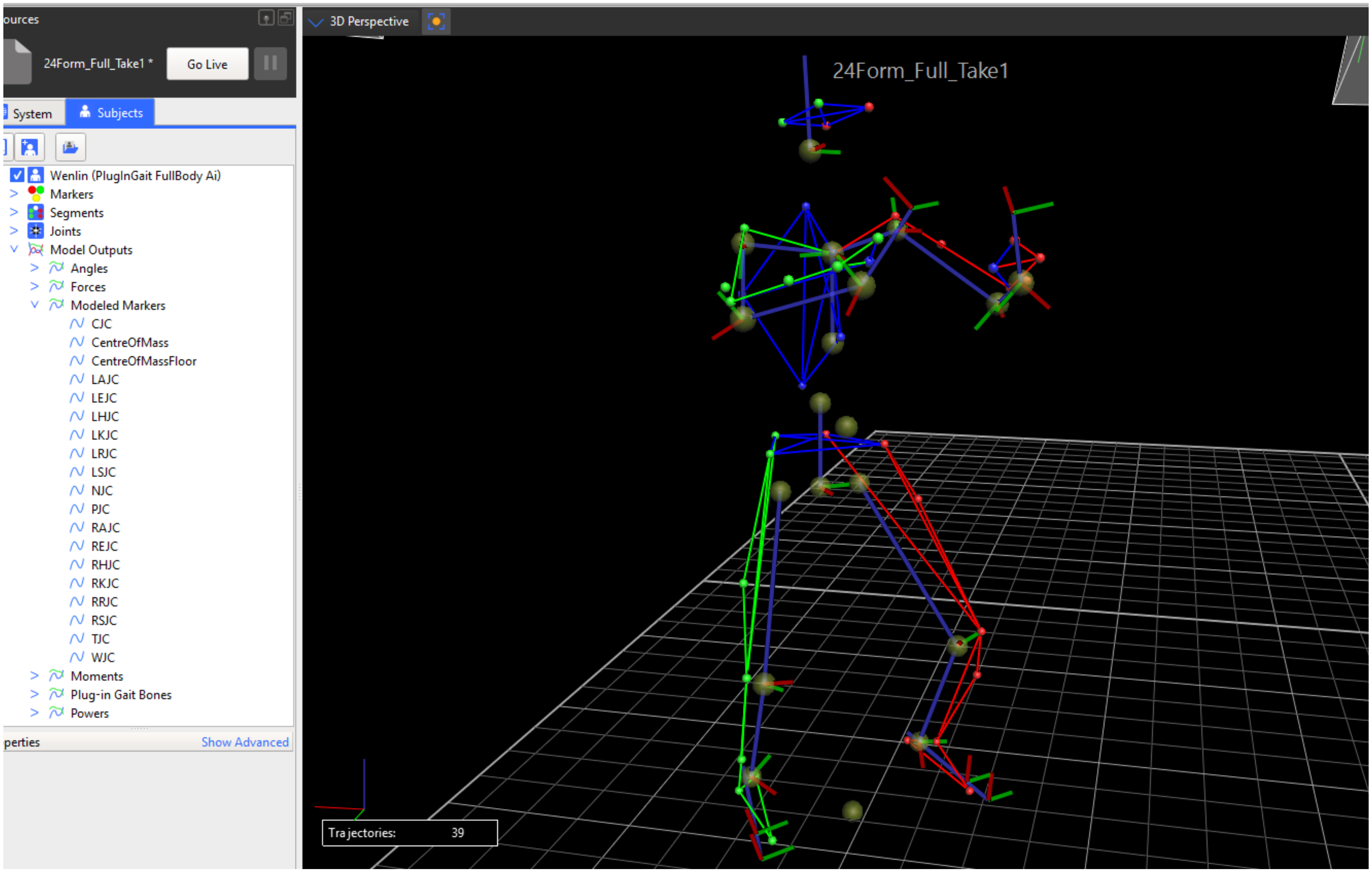
- Read in labeled motion capture data
  - 3D points
  - Part labels for each point
- Tracking points through time to form trajectories, thus propagating part labels
  - Data association of trajectories up to time t-1 with points detected at time t
  - Do Kalman filter to update each trajectory with the point associated with it at time t
- Write out the corrected points / labels
- Quantitative evaluation

Motion Capture Overview



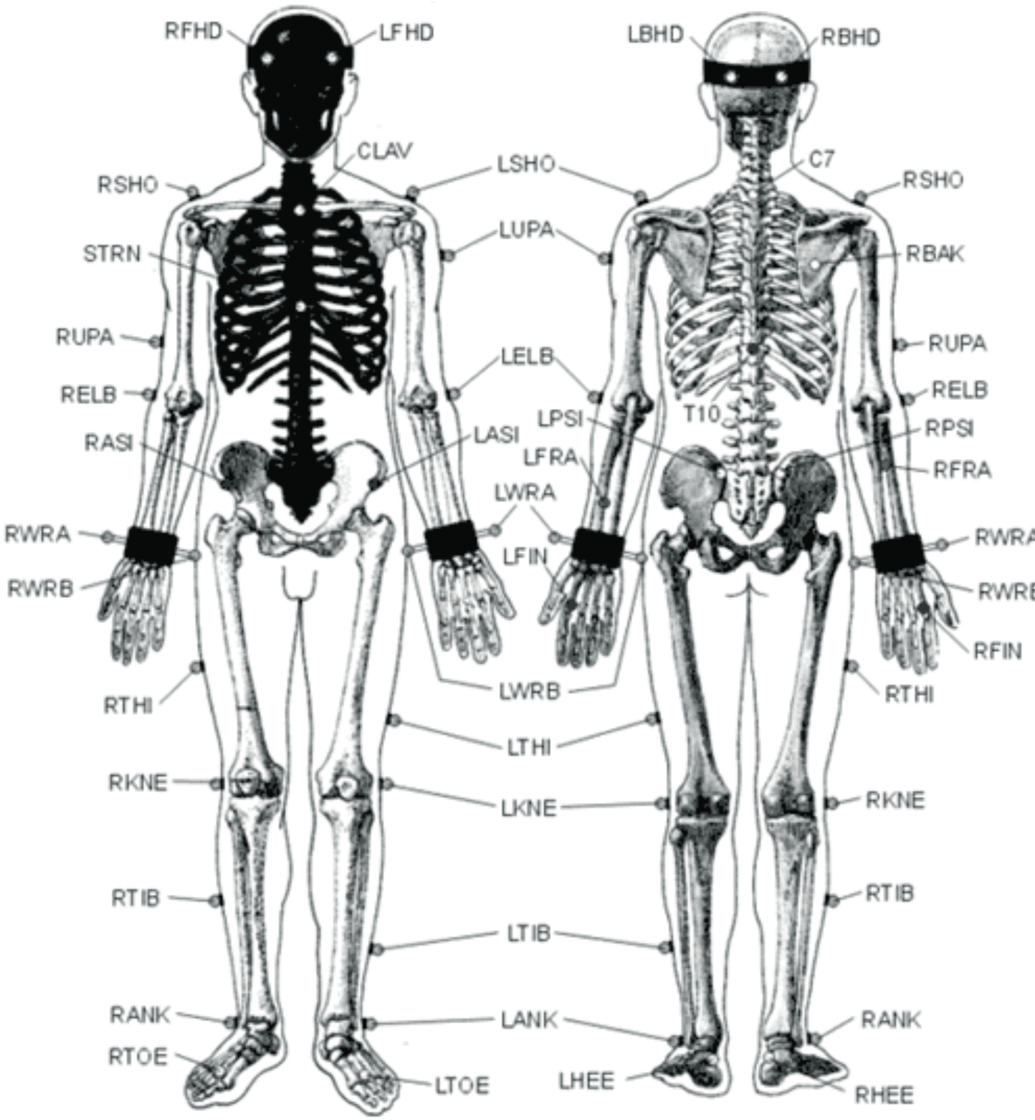
Robert Collins
CSE586

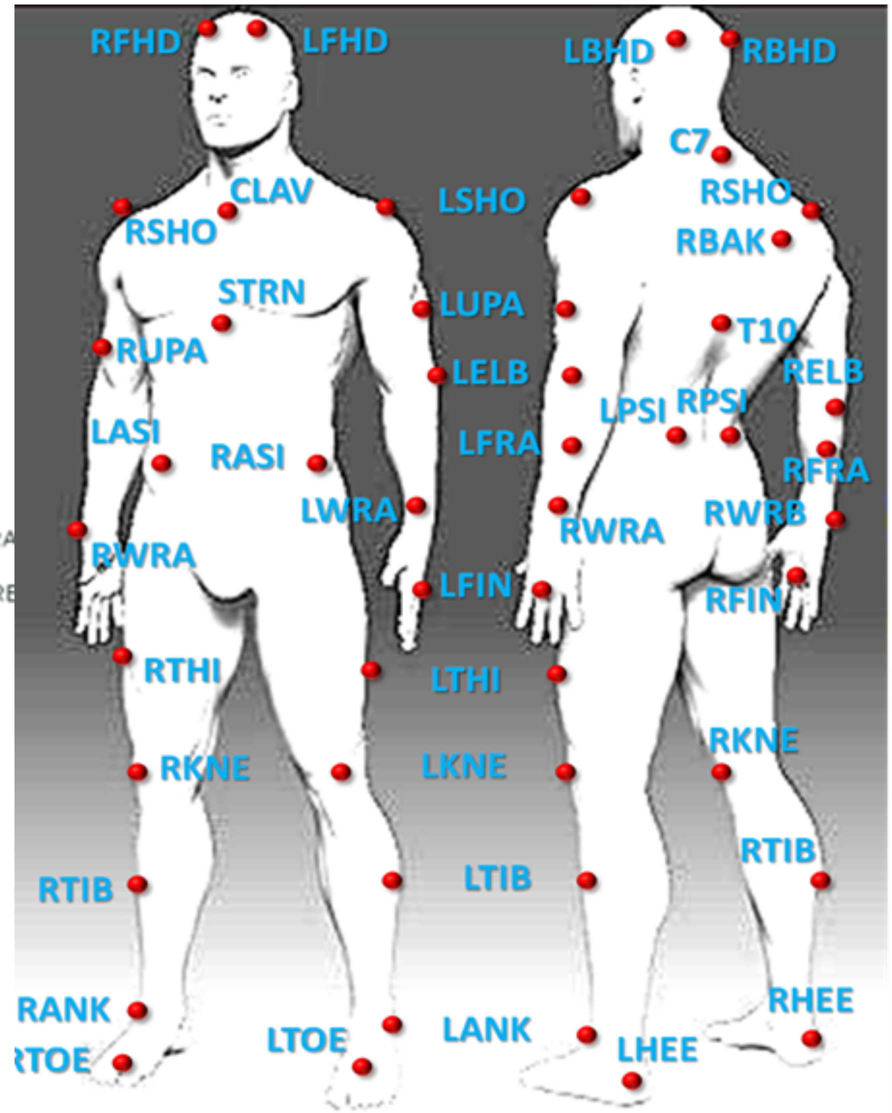
Motion Capture Overview



Robert Collins
CSE586

Marker Locations





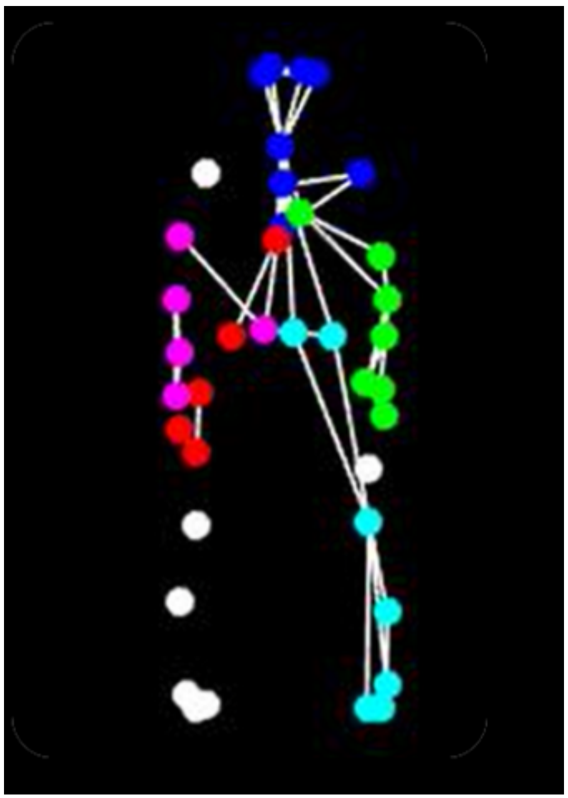
Motivation

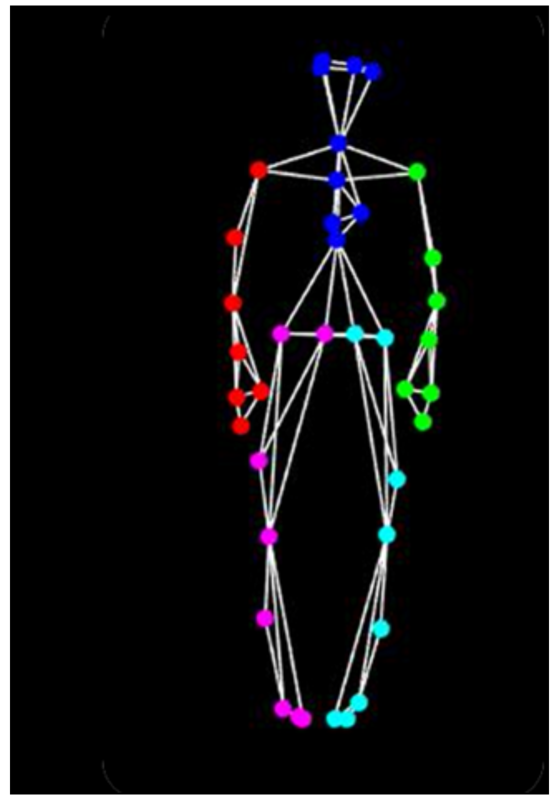
- The markers are identical 17mm diameter IR sphere reflectors.
- Because of this, when the VICON software finds the 3D positions of the markers there is no indication of which body part an individual marker corresponds to.
- VICON has an automatic labeling tool that tries to label the markers in the sequence, but...

Motivation

- The output of the VICON labeling tool has many errors including: unlabeled markers, mislabeled markers, extra markers, and missing markers.

VICON output





We want this!

Overall Hope

- After tracking, you will have 39 trajectories, each corresponding to one body marker.
- The ground truth label of a marker in one frame thus determines the correct label for all points on the same trajectory.

Getting Started

- Reference_Materials — **Background info + this ppt file**
- 08-21-15_Subject1 — **Data to use**
  - 24Form-Part1-Take1.frame_827.c3d — **One perfectly labeled frame**
  - 24Form-Part1-Take1.labeled_auto.c3d — **Sequence to be corrected**
- Filter_Code
  - SampleCode.m — **Start with this code...**
  - readMocapData39.m
  - visualizeSkeleton.m
  - MocapToolbox_v1.5
  - mcinitanimparSkeleton.m
  - orderData.m
  - orderNames.m

Getting Started

```
% SampleCode.m  - for Project 1 in Advanced Computer Vision, Spring 17
%
% The goal of the project is to implement a Kalman Filter that tracks and
% labels 39 motion capture markers automatically.
%
% Given:
%   initc3d - a single frame that has correctly labeled markers
%   datac3d - sequence of frames of data that needs to be corrected
%
% Desired:
%   outputc3d - this is a data structure with the corrected marker labeling
%
% This sample code shows how to read and visualize the motion capture data
% and gives a rough outline of the overall tracking code framework.
```

SampleCode.m is where you want to start

Extensions (do at least one in addition to the required components)

- Implement particle filtering, and compare with KF for tracking
- Implement JPDAF for “soft” data association
- Implement an optimal approach for solving “hard” data association (hard vs soft, not vs easy). Some choices are:
  - Kuhn-Munkres (Hungarian) algorithm
  - Linear programming
  - Min-cost network flow

When I say “implement” I mean implement the whole algorithm, not just call a library routine that does it.

Extensions (Continued)

- Use knowledge about neighboring markers to improve the data association / tracking
  - This is more open-ended... We don't necessarily know what to do here.
  - One idea: some pairs of marker points are constrained to be a certain distance away from each other, because they are both on a body part that is rigid (e.g. forearm; head; lower leg).
  - Another idea: points on the same rigid body should move similarly. If one is missing, perhaps you can use the velocity of the visible one.